Appendix A
Practices and Solutions

Table of Contents

Practices for Lesson 1

Practice 1-1: Exploring the Database Architecture

Fill in the blanks with the correct answers:

1) The two main components of Oracle RDBMS are
   _________________________ and _______________________.

2) An instance consists of _____________________and _____________________ processes.

3) A session is a connection between the _______________ process and either the _______________ process or the ________________ process.

4) Name some components of the System Global Area (SGA).
   - ____________________________
   - ____________________________
   - ____________________________
   - ____________________________
   - ____________________________
   - ____________________________
   - _____________________________

5) List some background processes:
   - ____________________________
   - ____________________________
   - ____________________________
   - ____________________________
   - ____________________________
   - ____________________________

6) The _______________________ process writes the dirty buffers to the data files.

7) The _______________________ process writes the redo logs to the log files.

8) Name some files associated with an Oracle database.

Practice 1-1: Exploring the Database Architecture (continued)

- ____________________________
- ____________________________
- ____________________________
- ____________________________
- ____________________________
- ____________________________
- _____________________________

9) Some of the logical storage structures of an Oracle database are:

- ____________________________
- ____________________________
- ____________________________
- ____________________________
- ____________________________
- ____________________________
- _____________________________

10) The ___________________ process copies the redo log files to an archive destination.

11) The _____________________ contains data and control information for a server or a background process.

12) The logical tablespace structure is associated with the physical ____________________ files on disk.

13) State whether the following statements are true or false.

a) The SGA includes the Database buffer cache and the Redo log buffer. ____

b) Each server process and background process has its own Program Global Area (PGA). ____

c) User processes run the application or Oracle tool code. ____

d) Oracle Database processes include server processes and background processes. ____.

Practice 1-1: Exploring the Database Architecture (continued)

14) From a terminal session connected as the `oracle` user, execute the `processes.sh` script located in your `$HOME/solutions/Database_Architecture` directory. What does this script show you?

    a) It shows you all the database instance processes currently running on your machine. This includes both background processes and foreground processes.

```
[oracle@edrsr33p1-orcl Database_Architecture]$ ./processes.sh
oracle    9537     1  0 01:00 ?        00:00:00 ora_w000_orcl
oracle   12132 22002  0 Mar26 pts/4    00:01:08
/u01/app/oracle/product/11.1.0/db_1/jdk/bin/java -server -Xmx256M -
XX:MaxPermSize=200m -XX:MinHeapFreeRatio=20 -XX:MaxHeapFreeRatio=40
-DORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1 -
Doracle.home=/u01/app/oracle/product/11.1.0/db_1/oc4j -
Doracle.oc4j.localhome=/u01/app/oracle/product/11.1.0/db_1/edrsr33p1
.us.oracle.com_orcl/sysman -
DEMSTATE=/u01/app/oracle/product/11.1.0/db_1/edrsr33p1.us.oracle.com
_orcl -Doracle.j2ee.dont.use.memory.archive=true -
Djava.protocol.handler.pkgs=HTTPClient -
Doracle.security.jazn.config=/u01/app/oracle/product/11.1.0/db_1/oc4
j/j2ee/OC4J_DBConsole_edrsr33p1.us.oracle.com_orcl/config/jazn.xml -
Djava.security.policy=/u01/app/oracle/product/11.1.0/db_1/oc4j/j2ee/
OC4J_DBConsole_edrsr33p1.us.oracle.com_orcl/config/java2.policy -
Djavax.net.ssl.KeyStore=/u01/app/oracle/product/11.1.0/db_1/sysman/c
onfig/OCMTrustedCerts.txt-
Djava.security.properties=/u01/app/oracle/product/11.1.0/db_1/oc4j/j
2ee/home/config/jazn.security.props -
DEMDROOT=/u01/app/oracle/product/11.1.0/db_1/edrsr33p1.us.oracle.com
_orcl -Dsysman.md5password=true -
Drepapi.oracle.home=/u01/app/oracle/product/11.1.0/db_1 -
Ddisable.checkForUpdate=true -
Doracle.sysman.ccr.ocmSDK.websvc.keystore=/u01/app/oracle/product/11
.1.0/db_1/jlib/emocmclnt.ks -
Dice.pilots.html4.ignoreNonGenericFonts=true -
Djava.awt.headless=true -jar
/u01/app/oracle/product/11.1.0/db_1/oc4j/j2ee/home/oc4j.jar -config
/u01/app/oracle/product/11.1.0/db_1/oc4j/j2ee/OC4J_DBConsole_edrsr33
p1.us.oracle.com_orcl/config/server.xml
oracle   12225     1  0 Mar26 ?        00:00:00 ora_pmon_orcl
oracle   12227     1  0 Mar26 ?        00:00:00 ora_vktm_orcl
oracle   12231     1  0 Mar26 ?        00:00:00 ora_diag_orcl
oracle   12233     1  0 Mar26 ?        00:00:00 ora_dbrm_orcl
oracle   12235     1  0 Mar26 ?        00:00:00 ora_psp0_orcl
oracle   12239     1  0 Mar26 ?        00:00:00 ora_dia0_orcl
oracle   12241     1  0 Mar26 ?        00:00:00 ora_mman_orcl
oracle   12243     1  0 Mar26 ?        00:00:21 ora_dbw0_orcl
oracle   12245     1  0 Mar26 ?        00:00:48 ora_lgwr_orcl
oracle   12247     1  0 Mar26 ?        00:00:01 ora_ckpt_orcl
oracle   12249     1  0 Mar26 ?        00:00:04 ora_smon_orcl
oracle   12251     1  0 Mar26 ?        00:00:00 ora_reco_orcl
oracle   12253     1  0 Mar26 ?        00:00:05 ora_mmon_orcl
oracle   12255     1  0 Mar26 ?        00:00:01 ora_mmnl_orcl
oracle   12257     1  0 Mar26 ?        00:00:00 ora_d000_orcl
oracle   12259     1  0 Mar26 ?        00:00:00 ora_s000_orcl
oracle   12282     1  0 Mar26 ?        00:00:00 ora_fbda_orcl
```

```
oracle   12284     1  0 Mar26 ?        00:00:00 ora_smco_orcl
oracle   12290     1  0 Mar26 ?        00:00:00 ora_qmnc_orcl
oracle   12302     1  0 Mar26 ?        00:00:03 oracleorcl
(LOCAL=NO)
oracle   12325     1  0 Mar26 ?        00:00:00 ora_q000_orcl
oracle   12329     1  0 Mar26 ?        00:01:59 oracleorcl
(LOCAL=NO)
oracle   12331     1  0 Mar26 ?        00:00:12 oracleorcl
(LOCAL=NO)
oracle   12333     1  0 Mar26 ?        00:00:05 oracleorcl
(LOCAL=NO)
oracle   12335     1  0 Mar26 ?        00:00:37 oracleorcl
(LOCAL=NO)
oracle   12340     1  0 Mar26 ?        00:01:51 oracleorcl
(LOCAL=NO)
oracle   12346     1  0 Mar26 ?        00:00:01 oracleorcl
(LOCAL=NO)
oracle   12362     1  0 Mar26 ?        00:00:00 ora_q001_orcl
oracle   12570     1  0 Mar26 ?        00:00:01 ora_cjq0_orcl
oracle   20119     1  0 14:50 ?        00:00:00 oracleorcl
(LOCAL=NO)
oracle   20482 20480  0 14:56 pts/2    00:00:00 grep orcl
oracle   22002     1  0 Mar26 pts/4    00:00:08
/u01/app/oracle/product/11.1.0/db_1/perl/bin/perl
/u01/app/oracle/product/11.1.0/db_1/bin/emwd.pl dbconsole
/u01/app/oracle/product/11.1.0/db_1/edrsr33p1.us.oracle.com_orcl/sys
man/log/emdb.nohup
[oracle@edrsr33p1-orcl Database_Architecture]$

--------------------------------------------------------------

#!/bin/bash

ps -ef | grep orcl
```

15) From a terminal session connected as the `oracle` user, execute the `files.sh` script located in your `$HOME/solutions/Database_Architecture` directory. What does this script show you?

   a) This script shows you the location and names of all database files, the initialization file, the password file, and trace files.

```
[oracle@edrsr33p1-orcl Database_Architecture]$ ./files.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Thu Mar 27 17:58:56
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
```

Practice 1-1: Exploring the Database Architecture (continued)

```
SQL>
SQL> col name format a45
SQL>
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

SQL>
SQL>
SQL> col member format a45
SQL>
SQL> select group#,member from v$logfile;

    GROUP# MEMBER
---------- ---------------------------------------------
         3 /u01/app/oracle/oradata/orcl/redo03.log
         2 /u01/app/oracle/oradata/orcl/redo02.log
         1 /u01/app/oracle/oradata/orcl/redo01.log

SQL>
SQL>
SQL> col tablespace_name format a20
SQL> col file_name format a45
SQL>
SQL> select tablespace_name, file_name from dba_data_files;

TABLESPACE_NAME      FILE_NAME
-------------------- ---------------------------------------------
USERS                /u01/app/oracle/oradata/orcl/users01.dbf
UNDOTBS1             /u01/app/oracle/oradata/orcl/undotbs01.dbf
SYSAUX               /u01/app/oracle/oradata/orcl/sysaux01.dbf
SYSTEM               /u01/app/oracle/oradata/orcl/system01.dbf
EXAMPLE              /u01/app/oracle/oradata/orcl/example01.dbf

SQL>
SQL> select tablespace_name, file_name from dba_temp_files;

TABLESPACE_NAME      FILE_NAME
-------------------- ---------------------------------------------
TEMP                 /u01/app/oracle/oradata/orcl/temp01.dbf

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
-rw-rw----  1 oracle oinstall 1544 Aug 22  2007
/u01/app/oracle/product/11.1.0/db_1/dbs/hc_orcl.dat
-rw-r-----  1 oracle oinstall 1536 Mar 26 22:03
/u01/app/oracle/product/11.1.0/db_1/dbs/orapworcl
-rw-r-----  1 oracle oinstall 2560 Mar 27 03:13
/u01/app/oracle/product/11.1.0/db_1/dbs/spfileorcl.ora
```

```
alert  cdump  hm  incident  incpkg  ir  lck  metadata  stage  sweep
trace
-rw-r--r--  1 oracle oinstall 557386 Mar 27 13:00
/u01/app/oracle/diag/rdbms/orcl/orcl/trace/alert_orcl.log
[oracle@edrsr33p1-orcl Database_Architecture]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Database_Architecture

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin


sqlplus / as sysdba @files.sql

ls -l $ORACLE_HOME/dbs/*orcl*

ls /u01/app/oracle/diag/rdbms/orcl/orcl

ls -l /u01/app/oracle/diag/rdbms/orcl/orcl/trace/alert*

--------------------------------------------------------------

set echo on

col name format a45

select name from v$controlfile;


col member format a45

select group#,member from v$logfile;


col tablespace_name format a20
col file_name format a45

select tablespace_name, file_name from dba_data_files;

select tablespace_name, file_name from dba_temp_files;

exit;
```

16) From a terminal session connected as the `oracle` user, execute the `sga.sh` script located in your `$HOME/solutions/Database_Architecture` directory. What does this script show you?

Practice 1-1: Exploring the Database Architecture (continued)

a) This script prints the various pools held in your SGA.

```
[oracle@edrsr33p1-orcl Database_Architecture]$ ./sga.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Thu Mar 27 18:15:02
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> select * from v$sgainfo;

NAME                                  BYTES RES
-------------------------------- ---------- ---
Fixed SGA Size                      1303188 No
Redo Buffers                        5181440 No
Buffer Cache Size                 335544320 Yes
Shared Pool Size                  276824064 Yes
Large Pool Size                     4194304 Yes
Java Pool Size                     12582912 Yes
Streams Pool Size                         0 Yes
Shared IO Pool Size                       0 Yes
Granule Size                        4194304 No
Maximum SGA Size                  845348864 No
Startup overhead in Shared Pool    46137344 No

NAME                                  BYTES RES
-------------------------------- ---------- ---
Free SGA Memory Available         209715200

12 rows selected.

SQL>
SQL> col component format a30
SQL>
SQL> select component,current_size,min_size,max_size from
v$sga_dynamic_components;

COMPONENT                      CURRENT_SIZE   MIN_SIZE   MAX_SIZE
------------------------------ ------------ ---------- ----------
shared pool                       276824064  226492416  276824064
large pool                          4194304    4194304    4194304
java pool                          12582912   12582912   12582912
streams pool                              0          0          0
DEFAULT buffer cache              335544320  335544320  385875968
KEEP buffer cache                         0          0          0
RECYCLE buffer cache                      0          0          0
DEFAULT 2K buffer cache                   0          0          0
DEFAULT 4K buffer cache                   0          0          0
DEFAULT 8K buffer cache                   0          0          0
```

Practice 1-1: Exploring the Database Architecture (continued)

```
DEFAULT 16K buffer cache                        0          0          0

COMPONENT                      CURRENT_SIZE   MIN_SIZE   MAX_SIZE
------------------------------ ------------ ---------- ----------
DEFAULT 32K buffer cache                        0          0          0
Shared IO Pool                                  0          0          0
ASM Buffer Cache                                0          0          0

14 rows selected.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Database_Architecture]$


--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Database_Architecture

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin


sqlplus / as sysdba @sga.sql

--------------------------------------------------------------

set echo on

select * from v$sgainfo;

col component format a30

select component,current_size,min_size,max_size from
v$sga_dynamic_components;

exit;
```

Practices for Lesson 2

Practice 2-1: Avoiding Common Mistakes

In this practice, you examine some common mistakes in writing SQL statements. You have to find a workaround to enhance performance.

1) You analyze a correlated subquery first. Before executing the culprit statement, execute the `correlation_setup.sh` script to set up the environment for this example. Make sure you run the script from a terminal session connected as the `oracle` user. You can find the scripts for all the following cases in your `$HOME/solutions/Common_Mistakes` directory.

```
[oracle@edrsr33p1-orcl Common_Mistakes]$ ./correlation_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Mar 26 20:21:50
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL>
User altered.

SQL> SQL>
Grant succeeded.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Common_Mistakes]$

--------------------------------------------------------------
#!/bin/bash

cd /home/oracle/solutions/Common_Mistakes

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba <<EOF

alter user sh identified by sh account unlock;

grant dba to sh;
EOF
```

Practice 2-1: Avoiding Common Mistakes (continued)

2) Connected as the `SH` user from a SQL*Plus session (stay connected to that session until the end of this case), make sure you execute the following command:

```
set timing on
@flush
```

The goal of the first command is to tell you how long the next command takes to execute. The `flush.sql` script flushes both the shared pool and the buffer cache to avoid most caching effects so that you can have good comparisons between two executions. **Note:** You should *not* use commands found in this script on a production system.

```
[oracle@edrsr33p1-orcl Common_Mistakes]$ sqlplus sh/sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Mar 26 20:37:49
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> set timing on
SQL> @flush

System altered.

Elapsed: 00:00:00.21

System altered.

Elapsed: 00:00:00.49
SQL>

--------------------------------------------------------------

alter system flush shared_pool;
alter system flush buffer_cache;
```

3) From the same SQL*Plus session, execute the following statement and note the time it takes to execute (You can use the `correlation.sql` script.):

```
SELECT COUNT(*)
FROM   products p
WHERE  prod_list_price < 1.15 * (SELECT avg(unit_cost)
                                 FROM costs c
                                 WHERE c.prod_id = p.prod_id);
```

```
SQL> @correlation
SQL>
SQL> SELECT COUNT(*)
  2  FROM   products p
  3  WHERE  prod_list_price < 1.15 * (SELECT avg(unit_cost)
```

```
  4                                       FROM costs c
  5                                       WHERE c.prod_id = p.prod_id);

  COUNT(*)
----------
        46

Elapsed: 00:00:01.21
SQL>

--------------------------------------------------------------

set timing on
set echo on

SELECT COUNT(*)
FROM   products p
WHERE  prod_list_price < 1.15 * (SELECT avg(unit_cost)
                                 FROM costs c
                                 WHERE c.prod_id = p.prod_id);
```

4) Before trying to fix the previous statement, flush your environment again using the `flush.sql` script from the SQL*Plus session.

```
SQL> @flush
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.10
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.17
SQL>
```

5) How do you rewrite this statement to enhance performance? Test your solution. You should discuss this with your instructor.

```
SQL> @nocorrelation
SQL>
SQL> set timing on
SQL> set echo on
SQL>
SQL> SELECT COUNT(*)
  2  FROM   products p, (SELECT prod_id, AVG(unit_cost) ac FROM
costs GROUP BY prod_id) c
  3  WHERE  p.prod_id = c.prod_id AND
  4         p.prod_list_price < 1.15 * c.ac;

  COUNT(*)
----------
        46

Elapsed: 00:00:00.16
```

Practice 2-1: Avoiding Common Mistakes (continued)

```
SQL>

--------------------------------------------------------------

set timing on
set echo on

SELECT COUNT(*)
FROM   products p, (SELECT prod_id, AVG(unit_cost) ac FROM costs
GROUP BY prod_id) c
WHERE  p.prod_id = c.prod_id AND
       p.prod_list_price < 1.15 * c.ac;
```

6) Exit from your SQL*Plus session.

```
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Common_Mistakes]$
```

7) From your terminal session, execute the `correlation_cleanup.sh` script to clean up your environment.

```
[oracle@edrsr33p1-orcl Common_Mistakes]$ ./correlation_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Mar 26 20:49:22
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL>
Revoke succeeded.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Common_Mistakes]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Common_Mistakes

export ORACLE_SID=orcl
```

```
export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba <<EOF

revoke dba from sh;

EOF
```

8) Before continuing, execute the `setup_rest.sh` script to set up the environment for all the examples that follow. Make sure you run the script from a terminal session connected as the `oracle` user. You can find the scripts for all the following cases in your `$HOME/solutions/Common_Mistakes` directory.

```
[oracle@edrsr33p1-orcl Common_Mistakes]$ ./setup_rest.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Mar 26 20:59:53
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL> SQL> SQL> drop user jfv cascade
          *
ERROR at line 1:
ORA-01918: user 'JFV' does not exist


SQL> SQL>
User created.

SQL> SQL>
Grant succeeded.

SQL> SQL> Connected.
SQL> SQL> drop table orders purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL> SQL>
Table created.

SQL> SQL>   2    3    4    5    6    7
PL/SQL procedure successfully completed.
```

Practice 2-1: Avoiding Common Mistakes (continued)

```
SQL> SQL>   2    3    4    5    6    7
PL/SQL procedure successfully completed.

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


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


SQL> SQL>
Table created.

SQL> SQL>   2    3    4    5    6    7
PL/SQL procedure successfully completed.

SQL> SQL>
Table created.

SQL> SQL>   2    3    4    5    6    7
PL/SQL procedure successfully completed.

SQL> SQL>
Index created.

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


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


SQL> SQL>
Table created.

SQL>
Table created.

SQL> SQL>   2    3    4    5    6    7
PL/SQL procedure successfully completed.

SQL> SQL>   2    3    4    5    6    7
PL/SQL procedure successfully completed.

SQL> SQL> SQL> Disconnected from Oracle Database 11g Enterprise
Edition Release 11.1.0.6.0 - Production
```

Practice 2-1: Avoiding Common Mistakes (continued)

```
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Common_Mistakes]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Common_Mistakes

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba <<EOF

set echo on

drop user jfv cascade;

create user jfv identified by jfv default tablespace users temporary
tablespace temp;

grant connect, resource, dba to jfv;

connect jfv/jfv

drop table orders purge;

create table orders (order_id_char varchar2(50) primary key,
order_total number, customer_name varchar2(300));

begin
for i in 1..500000 loop
insert into orders
values(i,100,'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaa');
end loop;
commit;
end;
/

begin
for i in 1..500000 loop
insert into orders
values(500000+i,100,'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa');
end loop;
commit;
```

Practice 2-1: Avoiding Common Mistakes (continued)

```
end;
/

drop table employees purge;
drop table job_history purge;

create table employees (employee_id number primary key, name
varchar2(500));

begin
for i in 1..500000 loop
insert into employees
values(i,'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaa');
end loop;
commit;
end;
/

create table job_history (employee_id number, job varchar2(500));

begin
for i in 1..500000 loop
insert into job_history
values(mod(i,1000),'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaa');
end loop;
commit;
end;
/

create index job_history_empid_indx on job_history(employee_id);


drop table old purge;
drop table new purge;

create table old(name varchar2(10), other varchar2(500));
create table new(name varchar2(10), other varchar2(500));

begin
for i in 1..500000 loop
insert into old
values(i,'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaa');
end loop;
commit;
end;
/

begin
```

```
for i in 1..500000 loop
insert into new
values(500000+i,'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaa');
end loop;
commit;
end;
/


EOF
```

9) Connect as the `JFV` user from a SQL*Plus session and stay connected in that session until further notice. In the session, set SQL*Plus timings on and flush your environment again before starting the second case. You can use the `set timing on` command and the `flush.sql` script for this.

```
[oracle@edrsr33p1-orcl Common_Mistakes]$ sqlplus jfv/jfv

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Mar 26 21:06:33
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> set timing on
SQL> @flush

System altered.

Elapsed: 00:00:00.33

System altered.

Elapsed: 00:00:07.12
SQL>

--------------------------------------------------------------

alter system flush shared_pool;
alter system flush buffer_cache;
```

Practice 2-1: Avoiding Common Mistakes (continued)

10) The second case you analyze is a join case. From your SQL*Plus session, execute the following query and note the time it takes to complete:

```
SELECT count(*)
FROM   job_history jh, employees e
WHERE
substr(to_char(e.employee_id),1)=substr(to_char(jh.employee_id),1);
```

```
SQL> @join
SQL>
SQL> SELECT count(*)
  2  FROM   job_history jh, employees e
  3  WHERE  substr(to_char(e.employee_id),1) =
substr(to_char(jh.employee_id),1);

  COUNT(*)
----------
    499500

Elapsed: 00:00:03.03
SQL>

--------------------------------------------------------------

set timing on
set echo on

SELECT count(*)
FROM   job_history jh, employees e
WHERE  substr(to_char(e.employee_id),1) =
substr(to_char(jh.employee_id),1);
```

11) Before trying to fix the previous statement, flush your environment again using the `flush.sql` script from your SQL*Plus session.

```
SQL> @flush
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.12
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.72
SQL>

--------------------------------------------------------------

alter system flush shared_pool;
alter system flush buffer_cache;
```

12) How would you rewrite the previous query for better performance? Test your solution. You should discuss this with your instructor.

Practice 2-1: Avoiding Common Mistakes (continued)

```
SQL> @better_join
SQL>
SQL> set timing on
SQL> set echo on
SQL>
SQL> SELECT count(*)
  2  FROM   job_history jh, employees e
  3  WHERE  e.employee_id = jh.employee_id;

  COUNT(*)
----------
    499500

Elapsed: 00:00:00.70
SQL>

--------------------------------------------------------------

set timing on
set echo on

SELECT count(*)
FROM   job_history jh, employees e
WHERE  e.employee_id = jh.employee_id;
```

13) Before analyzing the third case, flush your environment again using the `flush.sql` script from your SQL*Plus session.

```
SQL> @flush
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.11
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.23
SQL>

--------------------------------------------------------------

alter system flush shared_pool;
alter system flush buffer_cache;
```

14) The third case you analyze is a simple predicate case. Still connected as the `JFV` user from your SQL*Plus session, execute the following query and note the time it takes to complete:
`SELECT * FROM orders WHERE order_id_char = 1205;`

```
SQL> @simple_predicate
SQL>
SQL> set timing on
SQL> set echo on
```

Practice 2-1: Avoiding Common Mistakes (continued)

```
SQL>
SQL> SELECT * FROM orders WHERE order_id_char = 1205;

ORDER_ID_CHAR                                          ORDER_TOTAL
------------------------------------------------------ -----------
CUSTOMER_NAME
--------------------------------------------------------------------
------------
1205                                                           100
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa


Elapsed: 00:00:07.62
SQL>

--------------------------------------------------------------

set timing on
set echo on

SELECT * FROM orders WHERE order_id_char = 1205;
```

15) Before trying to fix the SELECT statement in step 14, flush your environment again using the flush.sql script.

```
SQL> @flush
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.13
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.20
SQL>
```

16) How would you rewrite the previous statement for better performance? Test your solution. You should discuss this with your instructor.

```
SQL> @better_predicate
SQL>
SQL> set timing on
SQL> set echo on
SQL>
SQL> SELECT * FROM orders WHERE order_id_char = '1205';

ORDER_ID_CHAR                                          ORDER_TOTAL
------------------------------------------------------ -----------
CUSTOMER_NAME
```

Practice 2-1: Avoiding Common Mistakes (continued)

```
---------------------------------------------------------------------
------------
1205                                                       100
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa


Elapsed: 00:00:00.04
SQL>

--------------------------------------------------------------

set timing on
set echo on

SELECT * FROM orders WHERE order_id_char = '1205';
```

17) Before proceeding with the next analysis, flush your environment again using the `flush.sql` script.

```
SQL> @flush
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.11
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.11
SQL>

--------------------------------------------------------------

alter system flush shared_pool;
alter system flush buffer_cache;
```

18) The fourth case is a `UNION` case. Execute the following query and note the time it takes to complete:
```
select count(*)
from (select name from old union select name from new);
```

```
SQL> @union
SQL>
SQL> set timing on
SQL> set echo on
SQL>
SQL> select count(*)
  2  from (select name from old union select name from new);

  COUNT(*)
```

Practice 2-1: Avoiding Common Mistakes (continued)

```
----------
   1000000

Elapsed: 00:00:06.39
SQL>
```

19) Before investigating a better solution, flush your environment again using the `flush.sql` script.

```
SQL> @flush
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.10
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.09
SQL>
```

20) How would you rewrite the previous statement for better performance? Test your solution. You should discuss this with your instructor.

```
SQL> @better_union
SQL>
SQL> set timing on
SQL> set echo on
SQL>
SQL> select count(*)
  2  from (select name from old union all select name from new);

  COUNT(*)
----------
   1000000

Elapsed: 00:00:00.42
SQL>
```

21) Execute the `multiple_setup.sql` script to set up the environment for this case.

```
SQL> @multiple_setup
SQL> create table myemp as select * from hr.employees;

Table created.

SQL> insert into myemp select * from myemp;
/
/
/
/
/
/
/
/
/
/
```

Practice 2-1: Avoiding Common Mistakes (continued)

```
/
/
/
/
/
commit;

107 rows created.

SQL>
214 rows created.

SQL>
428 rows created.

SQL>
856 rows created.

SQL>
1712 rows created.

SQL>
3424 rows created.

SQL>
6848 rows created.

SQL>
13696 rows created.

SQL>
27392 rows created.

SQL>
54784 rows created.

SQL>
109568 rows created.

SQL>
219136 rows created.

SQL>
438272 rows created.

SQL>
876544 rows created.

SQL>
1753088 rows created.

SQL>

3506176 rows created.

SQL>
Commit complete.
```

Practice 2-1: Avoiding Common Mistakes (continued)

```
SQL> insert into myemp select * from myemp;
commit;

7012352 rows created.

SQL>
Commit complete.

SQL>
```

22) Execute the `multiple1.sql` script and note the total time it takes to execute.

```
SQL> @multiple1
SQL> set timing on

SQL>
SQL>
SQL> SELECT COUNT (*)
FROM myemp
WHERE salary < 2000;
  2    3



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

Elapsed: 00:00:15.49
SQL> SELECT COUNT (*)
FROM myemp
WHERE salary BETWEEN 2000 AND 4000;
  2    3
  COUNT(*)
----------
   5636096

Elapsed: 00:00:17.14
SQL> SELECT COUNT (*)
FROM myemp
WHERE salary>4000;
  2    3
  COUNT(*)
----------
   8388608

Elapsed: 00:00:18.12
SQL>
```

23) How would you rewrite the statements found in `multiple1.sql` script for better performance?

```
SQL> @multiple2
SQL> SELECT COUNT (CASE WHEN salary < 2000
  2                      THEN 1 ELSE null END) count1,
```

Practice 2-1: Avoiding Common Mistakes (continued)

```
  3         COUNT (CASE WHEN salary BETWEEN 2001 AND 4000
  4                      THEN 1 ELSE null END) count2,
  5         COUNT (CASE WHEN salary > 4000
  6                      THEN 1 ELSE null END) count3
  7  FROM myemp;

    COUNT1     COUNT2     COUNT3
---------- ---------- ----------
         0    5636096    8388608

Elapsed: 00:00:18.19
SQL>
```

24) Exit from your SQL*Plus session.

```
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Common_Mistakes]$
```

25) You now analyze a fifth case that deals with database connections. Execute the `bad_connect.sh` script from your terminal window connected as the `oracle` user. Note the time it takes to complete.

```
[oracle@edrsr33p1-orcl Common_Mistakes]$ ./bad_connect.sh
Wed Mar 26 21:56:42 GMT-7 2008
Wed Mar 26 21:57:07 GMT-7 2008
[oracle@edrsr33p1-orcl Common_Mistakes]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Common_Mistakes

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin


STREAM_NUM=0
MAX_STREAM=500

date

while [ $STREAM_NUM -lt $MAX_STREAM ]; do

  # one more
  let STREAM_NUM="STREAM_NUM+1"

  # start one more stream
```

Practice 2-1: Avoiding Common Mistakes (continued)

```
  sqlplus -s jfv/jfv @select.sql >> /tmp/bad_connect.log 2>&1

done

date

--------------------------------------------------------------


select count(*) from dba_users;
exit;
```

26) Analyze the `bad_connect.sh` script and try to find a better solution to enhance the performance of that application. Test your solution. You should discuss this with your instructor.

```
[oracle@edrsr33p1-orcl Common_Mistakes]$ ./better_connect.sh
Wed Mar 26 22:00:48 GMT-7 2008
Wed Mar 26 22:00:50 GMT-7 2008
[oracle@edrsr33p1-orcl Common_Mistakes]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Common_Mistakes

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin


date

sqlplus -s jfv/jfv @select2.sql >> /tmp/better_connect.log 2>&1

date

--------------------------------------------------------------

declare
 c number;
begin
 for i in 1..500 loop
   select count(*) into c from dba_users;
 end loop;
end;
/
exit;
```

Practice 2-1: Avoiding Common Mistakes (continued)

27) Clean up your environment by executing the `cleanup_rest.sh` script from your terminal session.

```
[oracle@edrsr33p1-orcl Common_Mistakes]$ ./cleanup_rest.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Mar 26 22:03:10
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
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 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Common_Mistakes]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Common_Mistakes

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba <<EOF

set echo on

drop user jfv cascade;

EOF
```

Practices for Lesson 3

Practice 3-1: Understanding Optimizer Decisions

In this practice, you try to understand optimizer decisions relating to which execution plan to use. All the scripts needed for this practice can be found in your `$HOME/solutions/Trace_Event` directory.

1) Execute the `te_setup.sh` script. This script executes the following query and generates a trace file that contains all optimizer decisions.

```
SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
SUM(s.amount_sold) sales_amount
FROM   sh.sales s,sh.times t,sh.customers c,sh.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-Q1','1999-Q2')
GROUP BY ch.channel_class, c.cust_city, t.calendar_quarter_desc;
```

```
[oracle@edrsr33p1-orcl Trace_Event]$ ./te_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Apr 9 22:16:53 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> alter system flush shared_pool;

System altered.

SQL>
SQL> alter user sh identified by sh account unlock;

User altered.

SQL>
SQL> connect sh/sh
Connected.
SQL>
SQL> ALTER SESSION SET TRACEFILE_IDENTIFIER = 'MYOPTIMIZER';

Session altered.

SQL>
SQL> alter session set events '10053 trace name context forever,
level 1';

Session altered.

SQL>
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
SQL> SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
SUM(s.amount_sold) sales_amount
  2  FROM   sh.sales s,sh.times t,sh.customers c,sh.channels ch
  3  WHERE  s.time_id = t.time_id AND
  4         s.cust_id = c.cust_id AND
  5         s.channel_id = ch.channel_id AND
  6         c.cust_state_province = 'CA' AND
  7         ch.channel_desc IN ('Internet','Catalog') AND
  8         t.calendar_quarter_desc IN ('1999-Q1','1999-Q2')
  9  GROUP BY ch.channel_class, c.cust_city,
t.calendar_quarter_desc;

no rows selected

SQL>
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
orcl_ora_1801_MYOPTIMIZER.trc
orcl_ora_1801_MYOPTIMIZER.trm
[oracle@edrsr33p1-orcl Trace_Event]$
```

2) With the help of your instructor, see the generated trace and interpret the important parts of the trace file. **Note:** This lab is only for demonstration purposes. Do *not* use it on your production system unless explicitly asked by Oracle Support Services.

   a) The 10053 trace output is broken down into a number of sections that broadly reflect the stages that the optimizer goes through in evaluating a plan. These stages are as follows: query, parameters used by the optimizer, base statistical information, base table access cost, join order and method computations, recosting for special features, such as query transformations.

```
[oracle@edrsr33p1-orcl Trace_Event]$ cat myoptimizer.trc

Trace file
/u01/app/oracle/diag/rdbms/orcl/orcl/trace/orcl_ora_32643_MYOPTIMIZE
R.trc
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
ORACLE_HOME = /u01/app/oracle/product/11.1.0/db_1
System name:    Linux
Node name:      edrsr33p1.us.oracle.com
Release:        2.6.9-55.0.0.0.2.ELsmp
Version:        #1 SMP Wed May 2 14:59:56 PDT 2007
Machine:        i686
Instance name: orcl
Redo thread mounted by this instance: 1
Oracle process number: 18
Unix process pid: 32643, image: oracle@edrsr33p1.us.oracle.com (TNS
V1-V3)
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
*** 2008-04-09 21:46:51.950
*** SESSION ID:(134.35423) 2008-04-09 21:46:51.950
*** CLIENT ID:() 2008-04-09 21:46:51.950
*** SERVICE NAME:(SYS$USERS) 2008-04-09 21:46:51.950
*** MODULE NAME:(SQL*Plus) 2008-04-09 21:46:51.950
*** ACTION NAME:() 2008-04-09 21:46:51.950

Registered qb: SEL$1 0xb7e0905c (PARSER)
---------------------
QUERY BLOCK SIGNATURE
---------------------
  signature (): qb_name=SEL$1 nbfros=4 flg=0
    fro(0): flg=4 objn=72254 hint_alias="C"@"SEL$1"
    fro(1): flg=4 objn=72252 hint_alias="CH"@"SEL$1"
    fro(2): flg=4 objn=72192 hint_alias="S"@"SEL$1"
    fro(3): flg=4 objn=72250 hint_alias="T"@"SEL$1"

SPM: statement not found in SMB
DOP: Automatic degree of parallelism is disabled: Parameter.
PM: Considering predicate move-around in query block SEL$1 (#0)
**************************
Predicate Move-Around (PM)
**************************
OPTIMIZER INFORMATION

******************************************
----- Current SQL Statement for this session (sql_id=70fqjd9u1zk7c)
-----
SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
SUM(s.amount_sold) sales_amount
FROM   sh.sales s,sh.times t,sh.customers c,sh.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-Q1','1999-Q2')
GROUP BY ch.channel_class, c.cust_city, t.calendar_quarter_desc
*******************************************
Legend
The following abbreviations are used by optimizer trace.
CBQT - cost-based query transformation
JPPD - join predicate push-down
OJPPD - old-style (non-cost-based) JPPD
FPD - filter push-down
PM - predicate move-around
CVM - complex view merging
SPJ - select-project-join
SJC - set join conversion
SU - subquery unnesting
OBYE - order by elimination
ST - star transformation
CNT - count(col) to count(*) transformation
JE - Join Elimination
qb - query block
LB - leaf blocks
DK - distinct keys
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
LB/K - average number of leaf blocks per key
DB/K - average number of data blocks per key
CLUF - clustering factor
NDV - number of distinct values
Resp - response cost
Card - cardinality
Resc - resource cost
NL - nested loops (join)
SM - sort merge (join)
HA - hash (join)
CPUSPEED - CPU Speed
IOTFRSPEED - I/O transfer speed
IOSEEKTIM - I/O seek time
SREADTIM - average single block read time
MREADTIM - average multiblock read time
MBRC - average multiblock read count
MAXTHR - maximum I/O system throughput
SLAVETHR - average slave I/O throughput
dmeth - distribution method
  1: no partitioning required
  2: value partitioned
  4: right is random (round-robin)
  128: left is random (round-robin)
  8: broadcast right and partition left
  16: broadcast left and partition right
  32: partition left using partitioning of right
  64: partition right using partitioning of left
  256: run the join in serial
  0: invalid distribution method
sel - selectivity
ptn - partition
***************************************
PARAMETERS USED BY THE OPTIMIZER
********************************
  *************************************
  PARAMETERS WITH ALTERED VALUES
  ******************************
Compilation Environment Dump
_smm_min_size                      = 204 KB
_smm_max_size                      = 40960 KB
_smm_px_max_size                   = 102400 KB
Bug Fix Control Environment


  *************************************
  PARAMETERS WITH DEFAULT VALUES
  ******************************
Compilation Environment Dump
optimizer_mode_hinted              = false
optimizer_features_hinted          = 0.0.0
parallel_execution_enabled         = true
parallel_query_forced_dop          = 0
parallel_dml_forced_dop            = 0
parallel_ddl_forced_degree         = 0
parallel_ddl_forced_instances      = 0
_query_rewrite_fudge               = 90
optimizer_features_enable          = 11.1.0.6
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
_optimizer_search_limit            = 5

…

optimizer_use_invisible_indexes     = false
flashback_data_archive_internal_cursor = 0
_optimizer_extended_stats_usage_control = 240
Bug Fix Control Environment
    fix  3834770 = 1
    fix  3746511 = enabled
    fix  4519016 = enabled
    fix  3118776 = enabled
    fix  4488689 = enabled
    fix  2194204 = disabled
…
    fix  6133948 = enabled
    fix  6239909 = enabled


  ***************************************
  PARAMETERS IN OPT_PARAM HINT
  ****************************
***************************************
Column Usage Monitoring is ON: tracking level = 1
***************************************


Considering Query Transformations on query block SEL$1 (#0)
**************************
Query transformations (QT)
**************************
CBQT: Validity checks passed for 70fqjd9u1zk7c.
CSE: Considering common sub-expression elimination in query block
SEL$1 (#0)
*************************
Common Subexpression elimination (CSE)
*************************
CSE:     CSE not performed on query block SEL$1 (#0).
OBYE:   Considering Order-by Elimination from view SEL$1 (#0)
***************************
Order-by elimination (OBYE)
***************************
OBYE:     OBYE bypassed: no order by to eliminate.
JE:   Considering Join Elimination on query block SEL$1 (#0)
*************************
Join Elimination (JE)
*************************
SQL:******* UNPARSED QUERY IS *******
SELECT "CH"."CHANNEL_CLASS" "CHANNEL_CLASS","C"."CUST_CITY"
"CUST_CITY","T"."CALENDAR_QUARTER_DESC"
"CALENDAR_QUARTER_DESC",SUM("S"."AMOUNT_SOLD") "SALES_AMOUNT" FROM
"SH"."SALES" "S","SH"."TIMES" "T","SH"."CUSTOMERS"
"C","SH"."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"='Internet' OR "CH"."CHANNEL_DESC"='Catalog')
AND ("T"."CALENDAR_QUARTER_DESC"='1999-Q1' OR
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
"T"."CALENDAR_QUARTER_DESC"='1999-Q2') GROUP BY
"CH"."CHANNEL_CLASS","C"."CUST_CITY","T"."CALENDAR_QUARTER_DESC"
Query block SEL$1 (#0) unchanged
CNT:   Considering count(col) to count(*) on query block SEL$1 (#0)
*************************
Count(col) to Count(*) (CNT)
*************************
CNT:     COUNT() to COUNT(*) not done.
JE:   Considering Join Elimination on query block SEL$1 (#0)
*************************
Join Elimination (JE)
*************************
SQL:******* UNPARSED QUERY IS *******
SELECT "CH"."CHANNEL_CLASS" "CHANNEL_CLASS","C"."CUST_CITY"
"CUST_CITY","T"."CALENDAR_QUARTER_DESC"
"CALENDAR_QUARTER_DESC",SUM("S"."AMOUNT_SOLD") "SALES_AMOUNT" FROM
"SH"."SALES" "S","SH"."TIMES" "T","SH"."CUSTOMERS"
"C","SH"."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"='Internet' OR "CH"."CHANNEL_DESC"='Catalog')
AND ("T"."CALENDAR_QUARTER_DESC"='1999-Q1' OR
"T"."CALENDAR_QUARTER_DESC"='1999-Q2') GROUP BY
"CH"."CHANNEL_CLASS","C"."CUST_CITY","T"."CALENDAR_QUARTER_DESC"
Query block SEL$1 (#0) unchanged
query block SEL$1 (#0) unchanged
Considering Query Transformations on query block SEL$1 (#0)
**************************
Query transformations (QT)
**************************
CSE: Considering common sub-expression elimination in query block
SEL$1 (#0)
*************************
Common Subexpression elimination (CSE)
*************************
CSE:     CSE not performed on query block SEL$1 (#0).
query block SEL$1 (#0) unchanged
apadrv-start sqlid=8087006336042125548
  :
    call(in-use=62016, alloc=81864), compile(in-use=64064,
alloc=66868), execution(in-use=3376, alloc=4060)

*******************************************
Peeked values of the binds in SQL statement
*******************************************

CBQT: Considering cost-based transformation on query block SEL$1
(#0)
********************************
COST-BASED QUERY TRANSFORMATIONS
********************************
FPD: Considering simple filter push (pre rewrite) in query block
SEL$1 (#0)
FPD:  Current where clause predicates "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"='Internet'
```

```
OR "CH"."CHANNEL_DESC"='Catalog') AND
("T"."CALENDAR_QUARTER_DESC"='1999-Q1' OR "T"."C

OBYE:   Considering Order-by Elimination from view SEL$1 (#0)
***************************
Order-by elimination (OBYE)
***************************
OBYE:     OBYE bypassed: no order by to eliminate.
Considering Query Transformations on query block SEL$1 (#0)
**************************
Query transformations (QT)
**************************
CSE: Considering common sub-expression elimination in query block
SEL$1 (#0)
*************************
Common Subexpression elimination (CSE)
*************************
CSE:     CSE not performed on query block SEL$1 (#0).
kkqctdrvTD-start on query block SEL$1 (#0)
kkqctdrvTD-start: :
    call(in-use=62016, alloc=81864), compile(in-use=105980,
alloc=109196), execution(in-use=3376, alloc=4060)

kkqctdrvTD-cleanup: transform(in-use=0, alloc=0) :
    call(in-use=62016, alloc=81864), compile(in-use=106488,
alloc=109196), execution(in-use=3376, alloc=4060)

kkqctdrvTD-end:
    call(in-use=62016, alloc=81864), compile(in-use=106808,
alloc=109196), execution(in-use=3376, alloc=4060)

SJC: Considering set-join conversion in query block SEL$1 (#1)
*************************
Set-Join Conversion (SJC)
*************************
SJC: not performed
CNT:   Considering count(col) to count(*) on query block SEL$1 (#1)
*************************
Count(col) to Count(*) (CNT)
*************************
CNT:     COUNT() to COUNT(*) not done.
JE:   Considering Join Elimination on query block SEL$1 (#1)
*************************
Join Elimination (JE)
*************************
SQL:******* UNPARSED QUERY IS *******
SELECT "CH"."CHANNEL_CLASS" "CHANNEL_CLASS","C"."CUST_CITY"
"CUST_CITY","T"."CALENDAR_QUARTER_DESC"
"CALENDAR_QUARTER_DESC",SUM("S"."AMOUNT_SOLD") "SALES_AMOUNT" FROM
"SH"."SALES" "S","SH"."TIMES" "T","SH"."CUSTOMERS"
"C","SH"."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"='Internet' OR "CH"."CHANNEL_DESC"='Catalog')
AND ("T"."CALENDAR_QUARTER_DESC"='1999-Q1' OR
"T"."CALENDAR_QUARTER_DESC"='1999-Q2') GROUP BY
"CH"."CHANNEL_CLASS","C"."CUST_CITY","T"."CALENDAR_QUARTER_DESC"
```

```
Query block SEL$1 (#1) unchanged
PM: Considering predicate move-around in query block SEL$1 (#1)
**************************
Predicate Move-Around (PM)
**************************
PM:     PM bypassed: Outer query contains no views.
PM:     PM bypassed: Outer query contains no views.
kkqctdrvTD-start on query block SEL$1 (#1)
kkqctdrvTD-start: :
    call(in-use=81660, alloc=98240), compile(in-use=109508,
alloc=113320), execution(in-use=3376, alloc=4060)

kkqctdrvTD-cleanup: transform(in-use=0, alloc=0) :
    call(in-use=81660, alloc=98240), compile(in-use=109984,
alloc=113320), execution(in-use=3376, alloc=4060)

kkqctdrvTD-end:
    call(in-use=81660, alloc=98240), compile(in-use=110304,
alloc=113320), execution(in-use=3376, alloc=4060)

kkqctdrvTD-start on query block SEL$1 (#1)
kkqctdrvTD-start: :
    call(in-use=81660, alloc=98240), compile(in-use=110304,
alloc=113320), execution(in-use=3376, alloc=4060)

Registered qb: SEL$1 0xb7e01c6c (COPY SEL$1)
---------------------
QUERY BLOCK SIGNATURE
---------------------
  signature(): NULL
***********************************
Cost-Based Group By Placement
***********************************
GBP: Checking validity of GBP for query block SEL$1 (#1)
GBP: Checking validity of group-by placement for query block SEL$1
(#1)
GBP: Bypassed: QB has disjunction.
kkqctdrvTD-cleanup: transform(in-use=9008, alloc=9460) :
    call(in-use=81684, alloc=98240), compile(in-use=129100,
alloc=132448), execution(in-use=3376, alloc=4060)

kkqctdrvTD-end:
    call(in-use=81684, alloc=98240), compile(in-use=119884,
alloc=132448), execution(in-use=3376, alloc=4060)

GBP: Applying transformation directives
GBP: Checking validity of group-by placement for query block SEL$1
(#1)
GBP: Bypassed: QB has disjunction.
JPPD:  Considering Cost-based predicate pushdown from query block
SEL$1 (#1)
************************************
Cost-based predicate pushdown (JPPD)
************************************
kkqctdrvTD-start on query block SEL$1 (#1)
kkqctdrvTD-start: :
```

```
    call(in-use=81708, alloc=98240), compile(in-use=122632,
alloc=132448), execution(in-use=3376, alloc=4060)

kkqctdrvTD-cleanup: transform(in-use=0, alloc=0) :
    call(in-use=81708, alloc=98240), compile(in-use=123108,
alloc=132448), execution(in-use=3376, alloc=4060)

kkqctdrvTD-end:
    call(in-use=81708, alloc=98240), compile(in-use=123428,
alloc=132448), execution(in-use=3376, alloc=4060)

JPPD: Applying transformation directives
query block SEL$1 (#1) unchanged
FPD: Considering simple filter push in query block SEL$1 (#1)
"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"='Internet'
OR "CH"."CHANNEL_DESC"='Catalog') AND
("T"."CALENDAR_QUARTER_DESC"='1999-Q1' OR "T"."C
try to generate transitive predicate from check constraints for
query block SEL$1 (#1)
finally: "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"='Internet'
OR "CH"."CHANNEL_DESC"='Catalog') AND
("T"."CALENDAR_QUARTER_DESC"='1999-Q1' OR "T"."C

kkoqbc: optimizing query block SEL$1 (#1)

        :
    call(in-use=81716, alloc=98240), compile(in-use=124640,
alloc=132448), execution(in-use=3376, alloc=4060)

kkoqbc-subheap (create addr=0xb7d47960)
****************
QUERY BLOCK TEXT
****************
SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
SUM(s.amount_sold) sales_amount
FROM   sh.sales s,sh.times t,sh.customers c,sh.channels ch
WHERE  s.time_id = t.time_id AND
       s.cust_id = c.cust_id AND
       s.channel_id = ch.channel_id
---------------------
QUERY BLOCK SIGNATURE
---------------------
signature (optimizer): qb_name=SEL$1 nbfros=4 flg=0
  fro(0): flg=0 objn=72254 hint_alias="C"@"SEL$1"
  fro(1): flg=0 objn=72252 hint_alias="CH"@"SEL$1"
  fro(2): flg=0 objn=72192 hint_alias="S"@"SEL$1"
  fro(3): flg=0 objn=72250 hint_alias="T"@"SEL$1"

-----------------------------
SYSTEM STATISTICS INFORMATION
-----------------------------
  Using NOWORKLOAD Stats
  CPUSPEED: 1263 millions instructions/sec
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
  IOTFRSPEED: 4096 bytes per millisecond (default is 4096)
  IOSEEKTIM: 10 milliseconds (default is 10)

***************************************
BASE STATISTICAL INFORMATION
***********************
Table Stats::
  Table: CHANNELS  Alias: CH
    #Rows: 5  #Blks:  4  AvgRowLen:  40.00
Index Stats::
  Index: CHANNELS_PK  Col#: 1
    LVLS: 0  #LB: 1  #DK: 5  LB/K: 1.00  DB/K: 1.00  CLUF: 1.00
***********************
Table Stats::
  Table: CUSTOMERS  Alias:  C
    #Rows: 55500  #Blks:  1486  AvgRowLen:  180.00
Index Stats::
  Index: CUSTOMERS_GENDER_BIX  Col#: 4
    LVLS: 1  #LB: 3  #DK: 2  LB/K: 1.00  DB/K: 2.00  CLUF: 5.00
  Index: CUSTOMERS_MARITAL_BIX  Col#: 6
    LVLS: 1  #LB: 5  #DK: 11  LB/K: 1.00  DB/K: 1.00  CLUF: 18.00
  Index: CUSTOMERS_PK  Col#: 1
    LVLS: 1  #LB: 115  #DK: 55500  LB/K: 1.00  DB/K: 1.00  CLUF:
54405.00
  Index: CUSTOMERS_YOB_BIX  Col#: 5
    LVLS: 1  #LB: 19  #DK: 75  LB/K: 1.00  DB/K: 1.00  CLUF: 75.00
***********************
Table Stats::
  Table: TIMES  Alias:  T
    #Rows: 1826  #Blks:  59  AvgRowLen:  197.00
Index Stats::
  Index: TIMES_PK  Col#: 1
    LVLS: 1  #LB: 5  #DK: 1826  LB/K: 1.00  DB/K: 1.00  CLUF: 53.00
***********************
Table Stats::
  Table: SALES  Alias:  S  (Using composite stats)
    #Rows: 918843  #Blks:  1769  AvgRowLen:  29.00
Index Stats::
  Index: SALES_CHANNEL_BIX  Col#: 4
    USING COMPOSITE STATS
    LVLS: 1  #LB: 47  #DK: 4  LB/K: 11.00  DB/K: 23.00  CLUF: 92.00
  Index: SALES_CUST_BIX  Col#: 2
    USING COMPOSITE STATS
    LVLS: 1  #LB: 475  #DK: 7059  LB/K: 1.00  DB/K: 5.00  CLUF:
35808.00
  Index: SALES_PROD_BIX  Col#: 1
    USING COMPOSITE STATS
    LVLS: 1  #LB: 32  #DK: 72  LB/K: 1.00  DB/K: 14.00  CLUF:
1074.00
  Index: SALES_PROMO_BIX  Col#: 5
    USING COMPOSITE STATS
    LVLS: 1  #LB: 30  #DK: 4  LB/K: 7.00  DB/K: 13.00  CLUF: 54.00
  Index: SALES_TIME_BIX  Col#: 3
    USING COMPOSITE STATS
    LVLS: 1  #LB: 59  #DK: 1460  LB/K: 1.00  DB/K: 1.00  CLUF:
1460.00
Access path analysis for SALES
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
*************************************** 
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for SALES[S]
  Table: SALES  Alias: S
    Card: Original: 918843.000000  Rounded: 918843  Computed:
918843.00  Non Adjusted: 918843.00
  Access Path: TableScan
    Cost:  498.20  Resp: 498.20  Degree: 0
      Cost_io: 481.00  Cost_cpu: 260685437
      Resp_io: 481.00  Resp_cpu: 260685437
  ****** trying bitmap/domain indexes ******
  Access Path: index (FullScan)
    Index: SALES_CHANNEL_BIX
    resc_io: 75.00  resc_cpu: 552508
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 75.04  Resp: 75.04  Degree: 0
  Access Path: index (FullScan)
    Index: SALES_CUST_BIX
    resc_io: 503.00  resc_cpu: 10743684
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 503.71  Resp: 503.71  Degree: 0
  Access Path: index (FullScan)
    Index: SALES_PROD_BIX
    resc_io: 60.00  resc_cpu: 642086
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 60.04  Resp: 60.04  Degree: 0
  Access Path: index (FullScan)
    Index: SALES_PROMO_BIX
    resc_io: 58.00  resc_cpu: 423844
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 58.03  Resp: 58.03  Degree: 0
  Access Path: index (FullScan)
    Index: SALES_TIME_BIX
    resc_io: 60.00  resc_cpu: 719286
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 60.05  Resp: 60.05  Degree: 0
  Access Path: index (FullScan)
    Index: SALES_PROMO_BIX
    resc_io: 58.00  resc_cpu: 423844
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 58.03

Access path: Bitmap index - accepted
    Cost: 2895.306181 Cost_io: 2869.400000 Cost_cpu:
392576474.936000 Sel: 1.000000
    Not Believed to be index-only
  ****** finished trying bitmap/domain indexes ******
  Best:: AccessPath: TableScan
         Cost: 498.20  Degree: 1  Resp: 498.20  Card: 918843.00
Bytes: 0

Access path analysis for TIMES
***************************************
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for TIMES[T]
  Table: TIMES  Alias: T
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
    Card: Original: 1826.000000  Rounded: 183  Computed: 182.60  Non
Adjusted: 182.60
  Access Path: TableScan
    Cost:  18.15  Resp: 18.15  Degree: 0
      Cost_io: 18.00  Cost_cpu: 2314640
      Resp_io: 18.00  Resp_cpu: 2314640
  ****** trying bitmap/domain indexes ******
  Access Path: index (FullScan)
    Index: TIMES_PK
    resc_io: 6.00  resc_cpu: 407929
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 6.03  Resp: 6.03  Degree: 0
  ****** finished trying bitmap/domain indexes ******
  Best:: AccessPath: TableScan
         Cost: 18.15  Degree: 1  Resp: 18.15  Card: 182.60  Bytes: 0

Access path analysis for CUSTOMERS
***************************************
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for CUSTOMERS[C]
  Table: CUSTOMERS  Alias: C
    Card: Original: 55500.000000  Rounded: 383  Computed: 382.76
Non Adjusted: 382.76
  Access Path: TableScan
    Cost:  406.16  Resp: 406.16  Degree: 0
      Cost_io: 404.00  Cost_cpu: 32782460
      Resp_io: 404.00  Resp_cpu: 32782460
  ****** trying bitmap/domain indexes ******
  Access Path: index (FullScan)
    Index: CUSTOMERS_GENDER_BIX
    resc_io: 4.00  resc_cpu: 29486
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 4.00  Resp: 4.00  Degree: 0
  Access Path: index (FullScan)
    Index: CUSTOMERS_MARITAL_BIX
    resc_io: 6.00  resc_cpu: 46329
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 6.00  Resp: 6.00  Degree: 0
  Access Path: index (FullScan)
    Index: CUSTOMERS_PK
    resc_io: 116.00  resc_cpu: 11926087
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 116.79  Resp: 116.79  Degree: 0
  Access Path: index (FullScan)
    Index: CUSTOMERS_YOB_BIX
    resc_io: 20.00  resc_cpu: 157429
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 20.01  Resp: 20.01  Degree: 0
  Access Path: index (FullScan)
    Index: CUSTOMERS_GENDER_BIX
    resc_io: 4.00  resc_cpu: 29486
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 4.00  Resp: 4.00  Degree: 0
  Access path: Bitmap index - accepted
    Cost: 2367.447569 Cost_io: 2364.560000 Cost_cpu: 43757572.166400
Sel: 1.000000
    Not Believed to be index-only
```

```
  ****** finished trying bitmap/domain indexes ******
  Best:: AccessPath: TableScan
         Cost: 406.16  Degree: 1  Resp: 406.16  Card: 382.76  Bytes:
0

Access path analysis for CHANNELS
***************************************
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for CHANNELS[CH]
  Table: CHANNELS  Alias: CH
    Card: Original: 5.000000  Rounded: 2  Computed: 2.00  Non
Adjusted: 2.00
  Access Path: TableScan
    Cost:  3.00  Resp: 3.00  Degree: 0
      Cost_io: 3.00  Cost_cpu: 29826
      Resp_io: 3.00  Resp_cpu: 29826
  ****** trying bitmap/domain indexes ******
  Access Path: index (FullScan)
    Index: CHANNELS_PK
    resc_io: 1.00  resc_cpu: 8121
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 1.00  Resp: 1.00  Degree: 0
  ****** finished trying bitmap/domain indexes ******
  Best:: AccessPath: TableScan
         Cost: 3.00  Degree: 1  Resp: 3.00  Card: 2.00  Bytes: 0

Grouping column cardinality [CHANNEL_CL]    2
Grouping column cardinality [ CUST_CITY]    286
Grouping column cardinality [CALENDAR_Q]    2
***************************************


OPTIMIZER STATISTICS AND COMPUTATIONS
***************************************
GENERAL PLANS
***************************************
Considering cardinality-based initial join order.
Permutations for Starting Table :0
Join order[1]:  CHANNELS[CH]#0  TIMES[T]#1  CUSTOMERS[C]#2
SALES[S]#3

***************
Now joining: TIMES[T]#1
***************
NL Join
  Outer table: Card: 2.00  Cost: 3.00  Resp: 3.00  Degree: 1  Bytes:
21
Access path analysis for TIMES
  Inner table: TIMES  Alias: T
  Access Path: TableScan
    NL Join:  Cost: 37.31  Resp: 37.31  Degree: 1
      Cost_io: 37.00  Cost_cpu: 4659106
      Resp_io: 37.00  Resp_cpu: 4659106
  ****** trying bitmap/domain indexes ******
  Access Path: index (FullScan)
    Index: TIMES_PK
    resc_io: 6.00  resc_cpu: 407929
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 6.03  Resp: 6.03  Degree: 0
  ****** finished trying bitmap/domain indexes ******

  Best NL cost: 37.31
          resc: 37.31  resc_io: 37.00  resc_cpu: 4659106
          resp: 37.31  resp_io: 37.00  resc_cpu: 4659106
Join Card:  365.200000 = = outer (2.000000) * inner (182.600000) *
sel (1.000000)
Join Card - Rounded: 365 Computed: 365.20
Grouping column cardinality [CHANNEL_CL]    2
Grouping column cardinality [ CUST_CITY]    286
Grouping column cardinality [CALENDAR_Q]    2
Best:: JoinMethod: NestedLoop
       Cost: 37.31  Degree: 1  Resp: 37.31  Card: 365.20 Bytes: 37

***************
Now joining: CUSTOMERS[C]#2
***************
NL Join
  Outer table: Card: 365.20  Cost: 37.31  Resp: 37.31  Degree: 1
Bytes: 37
Access path analysis for CUSTOMERS
  Inner table: CUSTOMERS  Alias: C
  Access Path: TableScan
    NL Join:  Cost: 147725.92  Resp: 147725.92  Degree: 1
      Cost_io: 146936.00  Cost_cpu: 11970256947
      Resp_io: 146936.00  Resp_cpu: 11970256947
  ****** trying bitmap/domain indexes ******
  Access Path: index (FullScan)
    Index: CUSTOMERS_GENDER_BIX
    resc_io: 4.00  resc_cpu: 29486
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 4.00  Resp: 4.00  Degree: 0
  Access Path: index (FullScan)
    Index: CUSTOMERS_MARITAL_BIX
    resc_io: 6.00  resc_cpu: 46329
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 6.00  Resp: 6.00  Degree: 0
  Access Path: index (FullScan)
    Index: CUSTOMERS_PK
    resc_io: 116.00  resc_cpu: 11926087
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 116.79  Resp: 116.79  Degree: 0
  Access Path: index (FullScan)
    Index: CUSTOMERS_YOB_BIX
    resc_io: 20.00  resc_cpu: 157429
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    Cost: 20.01  Resp: 20.01  Degree: 0
  Access Path: index (FullScan)
    Index: CUSTOMERS_GENDER_BIX
    resc_io: 4.00  resc_cpu: 29486
    ix_sel: 1.000000  ix_sel_with_filters: 1.000000
    NL Join : Cost: 1498.02  Resp: 1498.02  Degree: 1
      Cost_io: 1497.00  Cost_cpu: 15421408
      Resp_io: 1497.00  Resp_cpu: 15421408
  Access path: Bitmap index - accepted
```

```
    Cost: 864192.977522 Cost_io: 863138.400000 Cost_cpu:
15980832052.096001 Sel: 1.000000
    Not Believed to be index-only
  ****** finished trying bitmap/domain indexes ******

  Best NL cost: 147725.92
          resc: 147725.92  resc_io: 146936.00  resc_cpu: 11970256947
          resp: 147725.92  resp_io: 146936.00  resc_cpu: 11970256947
Join Card:  139783.448276 = = outer (365.200000) * inner
(382.758621) * sel (1.000000)
Join Card - Rounded: 139783 Computed: 139783.45
Grouping column cardinality [CHANNEL_CL]    2
Grouping column cardinality [ CUST_CITY]    286
Grouping column cardinality [CALENDAR_Q]    2
Best:: JoinMethod: NestedLoop
       Cost: 147725.92  Degree: 1  Resp: 147725.92  Card: 139783.45
Bytes: 63

***************
Now joining: SALES[S]#3
***************
NL Join
  Outer table: Card: 139783.45  Cost: 147725.92  Resp: 147725.92
Degree: 1  Bytes: 63
Access path analysis for SALES
  Inner table: SALES  Alias: S
  Access Path: TableScan
    NL Join:  Cost: 2625413.78  Resp: 2625413.78  Degree: 1
      Cost_io: 2538743.82  Cost_cpu: 1313377131608
      Resp_io: 2538743.82  Resp_cpu: 1313377131608
  ****** trying bitmap/domain indexes ******
  Access Path: index (AllEqJoinGuess)
    Index: SALES_CHANNEL_BIX
    resc_io: 11.00  resc_cpu: 83786
    ix_sel: 0.250000  ix_sel_with_filters: 0.250000
    NL Join : Cost: 1686111.78  Resp: 1686111.78  Degree: 1
      Cost_io: 1684549.00  Cost_cpu: 23682093020
      Resp_io: 1684549.00  Resp_cpu: 23682093020
  Access Path: index (AllEqJoinGuess)
    Index: SALES_CUST_BIX
    resc_io: 1.00  resc_cpu: 9171
    ix_sel: 0.000142  ix_sel_with_filters: 0.000142
    NL Join : Cost: 287593.52  Resp: 287593.52  Degree: 1
      Cost_io: 286719.00  Cost_cpu: 13252268345
      Resp_io: 286719.00  Resp_cpu: 13252268345
  Access Path: index (AllEqJoinGuess)
    Index: SALES_TIME_BIX
    resc_io: 1.00  resc_cpu: 8171
    ix_sel: 0.000685  ix_sel_with_filters: 0.000685
    NL Join : Cost: 287584.29  Resp: 287584.29  Degree: 1
      Cost_io: 286719.00  Cost_cpu: 13112485345
      Resp_io: 286719.00  Resp_cpu: 13112485345
  Access path: Bitmap index - accepted
    Cost: 723020.120275 Cost_io: 720497.059076 Cost_cpu:
38233905490.990532 Sel: 0.000000
    Not Believed to be index-only
  ****** finished trying bitmap/domain indexes ******
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
  Best NL cost: 723020.12
          resc: 723020.12  resc_io: 720497.06  resc_cpu: 38233905491
          resp: 723020.12  resp_io: 720497.06  resc_cpu: 38233905491
Join Card:  3115.595241 = = outer (139783.448276) * inner
(918843.000000) * sel (0.000000)
Join Card - Rounded: 3116 Computed: 3115.60
Grouping column cardinality [CHANNEL_CL]    2
Grouping column cardinality [ CUST_CITY]    286
Grouping column cardinality [CALENDAR_Q]    2
  Outer table:  CUSTOMERS  Alias: C
    resc: 147725.92  card 139783.45  bytes: 63  deg: 1  resp:
147725.92
  Inner table:  SALES  Alias: S
    resc: 498.20  card: 918843.00  bytes: 21  deg: 1  resp: 498.20
    using dmeth: 2  #groups: 1
    SORT ressource         Sort statistics
      Sort width:         238 Area size:      208896 Max Area size:
41943040
      Degree:               1
      Blocks to Sort: 1370 Row size:     80 Total Rows:
139783
      Initial runs:   2 Merge passes:  1 IO Cost / pass:        744
      Total IO sort cost: 2114      Total CPU sort cost: 156539686
      Total Temp space used: 21423000
    SORT ressource         Sort statistics
      Sort width:         238 Area size:      208896 Max Area size:
41943040
      Degree:               1
      Blocks to Sort: 3825 Row size:     34 Total Rows:
918843
      Initial runs:   2 Merge passes:  1 IO Cost / pass:       2074
      Total IO sort cost: 5899      Total CPU sort cost: 929421655
      Total Temp space used: 66626000
  SM join: Resc: 156308.78  Resp: 156308.78  [multiMatchCost=0.00]
SM Join
  SM cost: 156308.78
     resc: 156308.78 resc_io: 155430.00 resc_cpu: 13316903726
     resp: 156308.78 resp_io: 155430.00 resp_cpu: 13316903726
  Outer table:  CUSTOMERS  Alias: C
    resc: 147725.92  card 139783.45  bytes: 63  deg: 1  resp:
147725.92
  Inner table:  SALES  Alias: S
    resc: 498.20  card: 918843.00  bytes: 21  deg: 1  resp: 498.20
    using dmeth: 2  #groups: 1
    Cost per ptn: 1944.36  #ptns: 1
    hash_area: 124 (max=10240) buildfrag: 1280  probefrag: 3702
ppasses: 1
  Hash join: Resc: 150168.48  Resp: 150168.48  [multiMatchCost=0.00]
HA Join
  HA cost: 150168.48
     resc: 150168.48 resc_io: 149346.00 resc_cpu: 12463693737
     resp: 150168.48 resp_io: 149346.00 resp_cpu: 12463693737
GROUP BY sort
GROUP BY adjustment factor: 0.500000
GROUP BY cardinality:  572.000000, TABLE cardinality:  3116.000000
    SORT ressource         Sort statistics
```

```
      Sort width:         238 Area size:      208896 Max Area size:
41943040
      Degree:               1
      Blocks to Sort: 40 Row size:     103 Total Rows:
3116
      Initial runs:   1 Merge passes:  0 IO Cost / pass:          0
      Total IO sort cost: 0      Total CPU sort cost: 16783070
      Total Temp space used: 0
Best:: JoinMethod: Hash
       Cost: 150169.59  Degree: 1  Resp: 150169.59  Card: 3115.60
Bytes: 84
***********************
Best so far:  Table#: 0  cost: 3.0020  card: 2.0000  bytes: 42
              Table#: 1  cost: 37.3075  card: 365.2000  bytes: 13505
              Table#: 2  cost: 147725.9191  card: 139783.4483
bytes: 8806329
              Table#: 3  cost: 150169.5886  card: 3115.5952  bytes:
261744
***********************
Join order[2]:  CHANNELS[CH]#0  TIMES[T]#1  SALES[S]#3
CUSTOMERS[C]#2

***************
Now joining: SALES[S]#3
***************
NL Join
  Outer table: Card: 365.20  Cost: 37.31  Resp: 37.31  Degree: 1
Bytes: 37
Access path analysis for SALES
  Inner table: SALES  Alias: S
  Access Path: TableScan
    NL Join:  Cost: 6507.09  Resp: 6507.09  Degree: 1

…

***********************
Join order[22]:  SALES[S]#3  CUSTOMERS[C]#2  TIMES[T]#1
CHANNELS[CH]#0

***************
Now joining: TIMES[T]#1
***************
NL Join
  Outer table: Card: 49822.22  Cost: 910.93  Resp: 910.93  Degree: 1
Bytes: 47
Access path analysis for TIMES
  Inner table: TIMES  Alias: T
  Access Path: TableScan
    NL Join:  Cost: 804636.92  Resp: 804636.92  Degree: 1
      Cost_io: 797001.00  Cost_cpu: 115712978623
      Resp_io: 797001.00  Resp_cpu: 115712978623
  Access Path: index (UniqueScan)
    Index: TIMES_PK
    resc_io: 1.00  resc_cpu: 10059
    ix_sel: 0.000548  ix_sel_with_filters: 0.000548
    NL Join : Cost: 50766.00  Resp: 50766.00  Degree: 1
      Cost_io: 50707.00  Cost_cpu: 894143044
```

```
      Resp_io: 50707.00  Resp_cpu: 894143044
  Access Path: index (AllEqUnique)
    Index: TIMES_PK
    resc_io: 1.00  resc_cpu: 10059
    ix_sel: 0.000548  ix_sel_with_filters: 0.000548
    NL Join : Cost: 50766.00  Resp: 50766.00  Degree: 1
      Cost_io: 50707.00  Cost_cpu: 894143044
      Resp_io: 50707.00  Resp_cpu: 894143044
  ****** trying bitmap/domain indexes ******
  ****** finished trying bitmap/domain indexes ******

  Best NL cost: 50766.00
          resc: 50766.00  resc_io: 50707.00  resc_cpu: 894143044
          resp: 50766.00  resp_io: 50707.00  resc_cpu: 894143044
Join Card:  6231.190483 = = outer (49822.224013) * inner
(182.600000) * sel (0.000685)
Join Card - Rounded: 6231 Computed: 6231.19
Grouping column cardinality [CHANNEL_CL]    2
Grouping column cardinality [ CUST_CITY]    286
Grouping column cardinality [CALENDAR_Q]    2
  Outer table:  CUSTOMERS  Alias: C
    resc: 910.93  card 49822.22  bytes: 47  deg: 1  resp: 910.93
  Inner table:  TIMES  Alias: T
    resc: 18.15  card: 182.60  bytes: 16  deg: 1  resp: 18.15
    using dmeth: 2  #groups: 1
    SORT ressource         Sort statistics
      Sort width:         238 Area size:      208896 Max Area size:
41943040
      Degree:               1
      Blocks to Sort: 379 Row size:     62 Total Rows:
49822
      Initial runs:   2 Merge passes:  1 IO Cost / pass:        206
      Total IO sort cost: 585      Total CPU sort cost: 59514576
      Total Temp space used: 6022000
    SORT ressource         Sort statistics
      Sort width:         238 Area size:      208896 Max Area size:
41943040
      Degree:               1
      Blocks to Sort: 1 Row size:     28 Total Rows:            183
      Initial runs:   1 Merge passes:  0 IO Cost / pass:          0
      Total IO sort cost: 0      Total CPU sort cost: 15215743
      Total Temp space used: 0
  SM join: Resc: 1519.02  Resp: 1519.02  [multiMatchCost=0.00]
SM Join
  SM cost: 1519.02
     resc: 1519.02 resc_io: 1488.00 resc_cpu: 470031494
     resp: 1519.02 resp_io: 1488.00 resp_cpu: 470031494
  Outer table:  CUSTOMERS  Alias: C
    resc: 910.93  card 49822.22  bytes: 47  deg: 1  resp: 910.93
  Inner table:  TIMES  Alias: T
    resc: 18.15  card: 182.60  bytes: 16  deg: 1  resp: 18.15
    using dmeth: 2  #groups: 1
    Cost per ptn: 141.09  #ptns: 1
    hash_area: 124 (max=10240) buildfrag: 359  probefrag: 1
ppasses: 1
  Hash join: Resc: 1070.22  Resp: 1070.22  [multiMatchCost=0.04]
  Outer table:  TIMES  Alias: T
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
    resc: 18.15  card 182.60  bytes: 16  deg: 1  resp: 18.15
  Inner table:  CUSTOMERS  Alias: C
    resc: 910.93  card: 49822.22  bytes: 47  deg: 1  resp: 910.93
    using dmeth: 2  #groups: 1
    Cost per ptn: 0.83  #ptns: 1
    hash_area: 124 (max=10240) buildfrag: 1  probefrag: 359
ppasses: 1
  Hash join: Resc: 929.92  Resp: 929.92  [multiMatchCost=0.00]
HA Join
  HA cost: 929.92 swapped
     resc: 929.92 resc_io: 903.00 resc_cpu: 407887714
     resp: 929.92 resp_io: 903.00 resp_cpu: 407887714
Best:: JoinMethod: Hash
       Cost: 929.92  Degree: 1  Resp: 929.92  Card: 6231.19 Bytes:
63

***************
Now joining: CHANNELS[CH]#0
***************
NL Join
  Outer table: Card: 6231.19  Cost: 929.92  Resp: 929.92  Degree: 1
Bytes: 63
Access path analysis for CHANNELS
  Inner table: CHANNELS  Alias: CH
  Access Path: TableScan
    NL Join:  Cost: 7694.18  Resp: 7694.18  Degree: 1
      Cost_io: 7655.00  Cost_cpu: 593732024
      Resp_io: 7655.00  Resp_cpu: 593732024
  Access Path: index (UniqueScan)
    Index: CHANNELS_PK
    resc_io: 1.00  resc_cpu: 8451
    ix_sel: 0.200000  ix_sel_with_filters: 0.200000
    NL Join : Cost: 7164.39  Resp: 7164.39  Degree: 1
      Cost_io: 7134.00  Cost_cpu: 460548636
      Resp_io: 7134.00  Resp_cpu: 460548636
  Access Path: index (AllEqUnique)
    Index: CHANNELS_PK
    resc_io: 1.00  resc_cpu: 8451
    ix_sel: 0.200000  ix_sel_with_filters: 0.200000
    NL Join : Cost: 7164.39  Resp: 7164.39  Degree: 1
      Cost_io: 7134.00  Cost_cpu: 460548636
      Resp_io: 7134.00  Resp_cpu: 460548636
  ****** trying bitmap/domain indexes ******
  ****** finished trying bitmap/domain indexes ******

  Best NL cost: 7164.39
          resc: 7164.39  resc_io: 7134.00  resc_cpu: 460548636
          resp: 7164.39  resp_io: 7134.00  resc_cpu: 460548636
Join Card:  3115.595241 = = outer (6231.190483) * inner (2.000000) *
sel (0.250000)
Join Card - Rounded: 3116 Computed: 3115.60
Grouping column cardinality [CHANNEL_CL]    2
Grouping column cardinality [ CUST_CITY]    286
Grouping column cardinality [CALENDAR_Q]    2
  Outer table:  TIMES  Alias: T
    resc: 929.92  card 6231.19  bytes: 63  deg: 1  resp: 929.92
  Inner table:  CHANNELS  Alias: CH
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
    resc: 3.00  card: 2.00  bytes: 21  deg: 1  resp: 3.00
    using dmeth: 2  #groups: 1
    SORT ressource         Sort statistics
      Sort width:         238 Area size:      208896 Max Area size:
41943040
      Degree:               1
      Blocks to Sort: 62 Row size:     80 Total Rows:           6231
      Initial runs:   2 Merge passes:  1 IO Cost / pass:         36
      Total IO sort cost: 98      Total CPU sort cost: 20219323
      Total Temp space used: 926000
    SORT ressource         Sort statistics
      Sort width:         238 Area size:      208896 Max Area size:
41943040
      Degree:               1
      Blocks to Sort: 1 Row size:     34 Total Rows:              2
      Initial runs:   1 Merge passes:  0 IO Cost / pass:          0
      Total IO sort cost: 0      Total CPU sort cost: 15153867
      Total Temp space used: 0
  SM join: Resc: 1033.25  Resp: 1033.25  [multiMatchCost=0.00]
SM Join
  SM cost: 1033.25
     resc: 1033.25 resc_io: 1004.00 resc_cpu: 443290729
     resp: 1033.25 resp_io: 1004.00 resp_cpu: 443290729
  Outer table:  TIMES  Alias: T
    resc: 929.92  card 6231.19  bytes: 63  deg: 1  resp: 929.92
  Inner table:  CHANNELS  Alias: CH
    resc: 3.00  card: 2.00  bytes: 21  deg: 1  resp: 3.00
    using dmeth: 2  #groups: 1
    Cost per ptn: 0.56  #ptns: 1
    hash_area: 124 (max=10240) buildfrag: 58  probefrag: 1  ppasses:
1
  Hash join: Resc: 933.50  Resp: 933.50  [multiMatchCost=0.02]
  Outer table:  CHANNELS  Alias: CH
    resc: 3.00  card 2.00  bytes: 21  deg: 1  resp: 3.00
  Inner table:  TIMES  Alias: T
    resc: 929.92  card: 6231.19  bytes: 63  deg: 1  resp: 929.92
    using dmeth: 2  #groups: 1
    Cost per ptn: 0.54  #ptns: 1
    hash_area: 124 (max=10240) buildfrag: 1  probefrag: 58  ppasses:
1
  Hash join: Resc: 933.46  Resp: 933.46  [multiMatchCost=0.00]
HA Join
  HA cost: 933.46 swapped
     resc: 933.46 resc_io: 906.00 resc_cpu: 416117828
     resp: 933.46 resp_io: 906.00 resp_cpu: 416117828
GROUP BY sort
GROUP BY adjustment factor: 0.500000
GROUP BY cardinality:  572.000000, TABLE cardinality:  3116.000000
    SORT ressource         Sort statistics
      Sort width:         238 Area size:      208896 Max Area size:
41943040
      Degree:               1
      Blocks to Sort: 40 Row size:     103 Total Rows:
3116
      Initial runs:   1 Merge passes:  0 IO Cost / pass:          0
      Total IO sort cost: 0      Total CPU sort cost: 16783070
      Total Temp space used: 0
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
Join order aborted: cost > best plan cost
***********************
(newjo-stop-1) k:0, spcnt:0, perm:22, maxperm:2000

*********************************
Number of join permutations tried: 22
*********************************
(newjo-save)    [1 3 2 0 ]
GROUP BY adjustment factor: 0.500000
GROUP BY cardinality:  572.000000, TABLE cardinality:  3116.000000
    SORT ressource         Sort statistics
      Sort width:         238 Area size:      208896 Max Area size:
41943040
      Degree:               1
      Blocks to Sort: 40 Row size:     103 Total Rows:
3116
      Initial runs:   1 Merge passes:  0 IO Cost / pass:          0
      Total IO sort cost: 0      Total CPU sort cost: 16783070
      Total Temp space used: 0
Trying or-Expansion on query block SEL$1 (#1)
Transfer Optimizer annotations for query block SEL$1 (#1)
id=0 frofand predicate="C"."CUST_STATE_PROVINCE"='CA'
id=0 frofkksm[i] (sort-merge/hash)
predicate="S"."CUST_ID"="C"."CUST_ID"
id=0 frosand (sort-merge/hash) predicate="S"."CUST_ID"="C"."CUST_ID"
id=0 frofkksm[i] (sort-merge/hash)
predicate="S"."TIME_ID"="T"."TIME_ID"
id=0 frosand (sort-merge/hash) predicate="S"."TIME_ID"="T"."TIME_ID"
id=0 frofand predicate="T"."CALENDAR_QUARTER_DESC"='1999-Q1' OR
"T"."CALENDAR_QUARTER_DESC"='1999-Q2'
id=0 frofkksm[i] (sort-merge/hash)
predicate="S"."CHANNEL_ID"="CH"."CHANNEL_ID"
id=0 frosand (sort-merge/hash)
predicate="S"."CHANNEL_ID"="CH"."CHANNEL_ID"
id=0 frofand predicate="CH"."CHANNEL_DESC"='Catalog' OR
"CH"."CHANNEL_DESC"='Internet'
GROUP BY adjustment factor: 1.000000
Final cost for query block SEL$1 (#1) - All Rows Plan:
  Best join order: 16
  Cost: 934.5672  Degree: 1  Card: 3116.0000  Bytes: 261744
  Resc: 934.5672  Resc_io: 906.0000  Resc_cpu: 432900898
  Resp: 934.5672  Resp_io: 906.0000  Resc_cpu: 432900898
kkoqbc-subheap (delete addr=0xb7d47960, in-use=105564, alloc=107812)
kkoqbc-end:
        :
    call(in-use=124380, alloc=246544), compile(in-use=132500,
alloc=136572), execution(in-use=3376, alloc=4060)

kkoqbc: finish optimizing query block SEL$1 (#1)
apadrv-end
          :
    call(in-use=124380, alloc=246544), compile(in-use=133196,
alloc=136572), execution(in-use=3376, alloc=4060)


Starting SQL statement dump
```

```
user_id=92 user_name=SH module=SQL*Plus action=
sql_id=70fqjd9u1zk7c plan_hash_value=1647000731 problem_type=3
----- Current SQL Statement for this session (sql_id=70fqjd9u1zk7c)
-----
SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
SUM(s.amount_sold) sales_amount
FROM   sh.sales s,sh.times t,sh.customers c,sh.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-Q1','1999-Q2')
GROUP BY ch.channel_class, c.cust_city, t.calendar_quarter_desc
sql_text_length=473
sql=SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
SUM(s.amount_sold) sales_amount
FROM   sh.sales s,sh.times t,sh.customers c,sh.channels ch
WHERE  s.time_id = t.time_id AND
       s.cust_id = c.cust_id AND
       s.channel_id = ch.channel_id
sql=AND
       c.cust_state_province = 'CA' AND
       ch.channel_desc IN ('Internet','Catalog') AND
       t.calendar_quarter_desc IN ('1999-Q1','1999-Q2')
GROUP BY ch.channel_class, c.cust_city, t.calendar_quarter_desc
----- Explain Plan Dump -----
----- Plan Table -----

============
Plan Table
============
-----------------------------------------------------+---------------
--------------------+---------------+
| Id  | Operation                        | Name      | Rows   | Bytes
| Cost  | Time       | Pstart| Pstop |
-----------------------------------------------------+---------------
--------------------+---------------+
| 0   | SELECT STATEMENT                 |           |        |
|   935 |            |       |       |
| 1   |  HASH GROUP BY                   |           |   572 |   47K
|   935 |  00:00:12 |       |       |
| 2   |   HASH JOIN                      |           |  3116 |  256K
|   933 |  00:00:12 |       |       |
| 3   |    TABLE ACCESS FULL             | CHANNELS  |      2 |    42
|     3 |  00:00:01 |       |       |
| 4   |    HASH JOIN                     |           |  6231 |  383K
|   930 |  00:00:12 |       |       |
| 5   |     PART JOIN FILTER CREATE      | :BF0000   |   183 |  2928
|    18 |  00:00:01 |       |       |
| 6   |      TABLE ACCESS FULL           | TIMES     |   183 |  2928
|    18 |  00:00:01 |       |       |
| 7   |     HASH JOIN                    |           |   49K | 2287K
|   911 |  00:00:11 |       |       |
| 8   |      TABLE ACCESS FULL           | CUSTOMERS|   383 |  9958
|   406 |  00:00:05 |       |       |
```

```
| 9   |      PARTITION RANGE JOIN-FILTER |          |  897K |   18M
|   498 |  00:00:06 | :BF0000| :BF0000|
| 10  |       TABLE ACCESS FULL          | SALES    |  897K |   18M
|   498 |  00:00:06 | :BF0000| :BF0000|
-----------------------------------------------------+--------------
-------------------+---------------+
Predicate Information:
----------------------
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-Q1' OR
"T"."CALENDAR_QUARTER_DESC"='1999-Q2'))
7 - access("S"."CUST_ID"="C"."CUST_ID")
8 - filter("C"."CUST_STATE_PROVINCE"='CA')

Content of other_xml column
===========================
  db_version     : 11.1.0.6
  parse_schema   : SH
  plan_hash      : 1647000731
  plan_hash_2    : 94088042
  Outline Data:
  /*+
    BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('11.1.0.6')
      DB_VERSION('11.1.0.6')
      ALL_ROWS
      OUTLINE_LEAF(@"SEL$1")
      FULL(@"SEL$1" "C"@"SEL$1")
      FULL(@"SEL$1" "S"@"SEL$1")
      FULL(@"SEL$1" "T"@"SEL$1")
      FULL(@"SEL$1" "CH"@"SEL$1")
      LEADING(@"SEL$1" "C"@"SEL$1" "S"@"SEL$1" "T"@"SEL$1"
"CH"@"SEL$1")
      USE_HASH(@"SEL$1" "S"@"SEL$1")
      USE_HASH(@"SEL$1" "T"@"SEL$1")
      USE_HASH(@"SEL$1" "CH"@"SEL$1")
      SWAP_JOIN_INPUTS(@"SEL$1" "T"@"SEL$1")
      SWAP_JOIN_INPUTS(@"SEL$1" "CH"@"SEL$1")
      USE_HASH_AGGREGATION(@"SEL$1")
    END_OUTLINE_DATA
  */

Optimizer state dump:
Compilation Environment Dump
optimizer_mode_hinted               = false
optimizer_features_hinted           = 0.0.0
parallel_execution_enabled          = true
parallel_query_forced_dop           = 0
parallel_dml_forced_dop             = 0
parallel_ddl_forced_degree          = 0
parallel_ddl_forced_instances       = 0
_query_rewrite_fudge                = 90
optimizer_features_enable           = 11.1.0.6
```

Practice 3-1: Understanding Optimizer Decisions (continued)

```
…

Bug Fix Control Environment
    fix  3834770 = 1
    fix  3746511 = enabled
    fix  4519016 = enabled
    fix  3118776 = enabled
    fix  4488689 = enabled
    fix  2194204 = disabled
    fix  2660592 = enabled



…


Query Block Registry:
SEL$1 0xb7e0905c (PARSER) [FINAL]

:
    call(in-use=143844, alloc=246544), compile(in-use=166732,
alloc=224788), execution(in-use=13796, alloc=16288)

End of Optimizer State Dump
===================== END SQL Statement Dump ======================
[oracle@edrsr33p1-orcl Trace_Event]$
```

3) Execute the `te_cleanup.sh` script to clean up your environment for this lab.

```
[oracle@edrsr33p1-orcl Trace_Event]$ ./te_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Apr 9 22:18:01 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Trace_Event]$
```

Practices for Lesson 4

Practice 4-1: Using Different Access Paths

In this practice, you explore various access paths the optimizer can use, and compare them. You have the possibility of exploring 16 different scenarios, each of which are self-contained. All scripts needed for this lab can be found in your `$HOME/solutions/Access_Paths` directory.

1) Case 1: Connected as the `oracle` user from a terminal session, execute the `ap_setup.sh` script.

```
[oracle@edrsr33p1-orcl Access_Paths]$ ./ap_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Apr 9 14:24:51 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> alter user sh identified by sh account unlock;

User altered.

SQL>
SQL> grant dba to sh;

Grant succeeded.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Access_Paths]$
```

2) In the same terminal session, connect as the `SH` user in the SQL*Plus session.

```
[oracle@edrsr33p1-orcl Access_Paths]$ sqlplus sh/sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Apr 9 14:24:58 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
```

Practice 4-1: Using Different Access Paths (continued)

3) Unless otherwise notified, stay connected to your SQL*Plus session as the `SH` user. Execute the `idx_setup.sql` script to set up your environment for case 1.

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


SQL>
SQL> create table mysales as select * from sh.sales;

Table created.

SQL>
SQL> insert into mysales select * from mysales;

918843 rows created.

SQL> commit;

Commit complete.

SQL>
SQL>
SQL> insert into mysales select * from mysales;

1837686 rows created.

SQL> commit;

Commit complete.

SQL>
SQL> insert into mysales select * from mysales;

3675372 rows created.

SQL> commit;

Commit complete.

SQL>
SQL> insert into mysales select * from mysales;

7350744 rows created.

SQL> commit;

Commit complete.

SQL>
SQL> insert into mysales select * from mysales;
```

Practice 4-1: Using Different Access Paths (continued)

```
14701488 rows created.

SQL> commit;

Commit complete.

SQL>
SQL> insert into mysales values (0,0,sysdate,0,0,0,0);

1 row created.

SQL> commit;

Commit complete.

SQL>
SQL> exec dbms_stats.gather_schema_stats('SH');

PL/SQL procedure successfully completed.

SQL>
```

4) Set up your session with the following commands: `set timing on`, `set autotrace trace only`, and `set linesize 200`. After this, execute the `select * from mysales where prod_id=0;` query. What do you observe?

   a) Basically, there are no indexes on the `MYSALES` table.

   b) .The only possibility for the optimizer is to use the full table scan to retrieve only one row. You can see that the scan takes a long time.

```
SQL> @with_and_without_index
SQL> set echo on
SQL>
SQL> set timing on
SQL> set autotrace traceonly
SQL> set linesize 200 pagesize 1000
SQL>
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.61
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:15.04
SQL>
SQL> select * from mysales where prod_id=0;

Elapsed: 00:00:28.85

Execution Plan
----------------------------------------------------------
Plan hash value: 3597614299
```

Practice 4-1: Using Different Access Paths (continued)

```
---------------------------------------------------------------------
---------
| Id  | Operation          | Name     | Rows   | Bytes | Cost (%CPU)|
Time     |
---------------------------------------------------------------------
---------
|   0 | SELECT STATEMENT  |          |   402K|    11M| 40249   (2)|
00:08:03 |
|*  1 |  TABLE ACCESS FULL| MYSALES |   402K|    11M| 40249   (2)|
00:08:03 |
---------------------------------------------------------------------
---------

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

   1 - filter("PROD_ID"=0)


Statistics
----------------------------------------------------------
        421  recursive calls
          0  db block gets
     141606  consistent gets
     141529  physical reads
          0  redo size
        790  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          5  sorts (memory)
          0  sorts (disk)
          1  rows processed

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

5) How do you enhance the performance of the query in step 4? Implement your solution.

```
SQL> @create_mysales_index
SQL> set echo on
SQL>
SQL> create index mysales_prodid_idx on mysales(prod_id) nologging
compute statistics;

Index created.

SQL>
```

6) Execute the same query again and verify that performance is enhanced now.

   a) After implementing the index, the optimizer can use it to speed up the query execution time. You can see a dramatic improvement in performance.

```
SQL> @with_and_without_index
SQL> set echo on
```

Practice 4-1: Using Different Access Paths (continued)

```
SQL>
SQL> set timing on
SQL> set autotrace traceonly
SQL> set linesize 200 pagesize 1000
SQL>
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.29
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:01.71
SQL>
SQL> select * from mysales where prod_id=0;

Elapsed: 00:00:00.88

Execution Plan
----------------------------------------------------------
Plan hash value: 3009203711

--------------------------------------------------------------------
------------------------------
| Id  | Operation                   | Name               | Rows  |
Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
------------------------------
|   0 | SELECT STATEMENT            |                    |   402K|
11M|  6079   (1)| 00:01:13 |
|   1 |  TABLE ACCESS BY INDEX ROWID| MYSALES            |   402K|
11M|  6079   (1)| 00:01:13 |
|*  2 |   INDEX RANGE SCAN          | MYSALES_PRODID_IDX |   402K|
|   822   (1)| 00:00:10 |
--------------------------------------------------------------------
------------------------------

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

   2 - access("PROD_ID"=0)


Statistics
----------------------------------------------------------
        496  recursive calls
          0  db block gets
        104  consistent gets
         21  physical reads
          0  redo size
        794  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
```

```
          1  rows processed

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

7) Clean up your environment for case 1 by executing the `idx_cleanup.sql` script.

```
SQL> @idx_cleanup
SQL> set echo on
SQL>
SQL> drop table mysales purge;

Table dropped.

SQL>
SQL>
```

8) Case 2: Drop all indexes currently created on the `CUSTOMERS` table except its primary key index.

```
SQL> @drop_customers_indexes
SQL> set    termout off
SQL>

--------------------------------------------------------------

set    termout off
store  set sqlplus_settings replace
save   buffer.sql replace
set    timing off heading off verify off autotrace off feedback off

spool  dait.sql

SELECT 'drop index '||i.index_name||';'
FROM   user_indexes i
WHERE  i.table_name = 'CUSTOMERS'
AND    NOT EXISTS
      (SELECT 'x'
       FROM   user_constraints c
       WHERE  c.index_name = i.index_name
       AND    c.table_name = i.table_name
       AND    c.status = 'ENABLED');

spool  off
@dait

get    buffer.sql nolist
@sqlplus_settings
set    termout on
```

Practice 4-1: Using Different Access Paths (continued)

9) Execute the following query:

```
SELECT /*+ FULL(c) */ c.*
FROM   customers c
WHERE  cust_gender   = 'M'
AND    cust_postal_code = 40804
AND   cust_credit_limit = 10000;
```

What do you observe?

a) The optimizer uses a full table scan and the cost for this query is relatively high.

```
SQL> @query00
SQL> set echo on
SQL>
SQL> set timing on
SQL> set autotrace traceonly
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT /*+ FULL(c) */ c.*
  2  FROM   customers c
  3  WHERE  cust_gender   = 'M'
  4  AND    cust_postal_code = 40804
  5  AND   cust_credit_limit = 10000
  6  /

6 rows selected.

Elapsed: 00:00:00.29

Execution Plan
----------------------------------------------------------
Plan hash value: 2008213504

--------------------------------------------------------------------
-----------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)|
Time     |
--------------------------------------------------------------------
-----------
|   0 | SELECT STATEMENT  |           |     6 |  1080 |   407   (1)|
00:00:05 |
|*  1 |  TABLE ACCESS FULL| CUSTOMERS |     6 |  1080 |   407   (1)|
00:00:05 |
--------------------------------------------------------------------
-----------

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

   1 - filter(TO_NUMBER("CUST_POSTAL_CODE")=40804 AND
              "CUST_CREDIT_LIMIT"=10000 AND "CUST_GENDER"='M')


Statistics
----------------------------------------------------------
       1067  recursive calls
          0  db block gets
       1722  consistent gets
```

```
       1461  physical reads
        116  redo size
       2570  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         27  sorts (memory)
          0  sorts (disk)
          6  rows processed

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

10) Create three B*-tree indexes on the following `CUSTOMERS` table columns:
`cust_gender`
`cust_postal_code`
`cust_credit_limit`

```
SQL> @create_cust_gender_index
SQL> set echo on
SQL>
SQL> CREATE INDEX cust_cust_gender_idx
  2  ON customers(cust_gender)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL> @create_cust_postal_code_index
SQL> set echo on
SQL>
SQL> CREATE INDEX cust_cust_postal_code_idx
  2  ON customers(cust_postal_code)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL> @create_cust_credit_limit_index
SQL> set echo on
SQL>
SQL> CREATE INDEX cust_cust_credit_limit_idx
  2  ON customers(cust_credit_limit)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL> @list_customers_indexes
SQL> set echo on
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT      ui.table_name
  2  ,        decode(ui.index_type
  3                 ,'NORMAL', ui.uniqueness
  4                 ,ui.index_type) AS index_type
```

Practice 4-1: Using Different Access Paths (continued)

```
  5  ,         ui.index_name
  6  FROM         user_indexes  ui
  7  WHERE        ui.table_name = 'CUSTOMERS'
  8  ORDER BY ui.table_name
  9  ,         ui.uniqueness desc;

TABLE_NAME                     INDEX_TYPE
INDEX_NAME
------------------------------ --------------------------- ---------
---------------------
CUSTOMERS                      UNIQUE
CUSTOMERS_PK
CUSTOMERS                      NONUNIQUE
CUST_CUST_GENDER_IDX
CUSTOMERS                      NONUNIQUE
CUST_CUST_CREDIT_LIMIT_IDX
CUSTOMERS                      NONUNIQUE
CUST_CUST_POSTAL_CODE_IDX

SQL>
```

11) Start monitoring all the `CUSTOMERS` indexes.

```
SQL> @start_monitoring_indexes
SQL> set echo on
SQL>
SQL> ALTER INDEX CUSTOMERS_PK MONITORING USAGE;

Index altered.

SQL>
SQL> ALTER INDEX CUST_CUST_POSTAL_CODE_IDX MONITORING USAGE;

Index altered.

SQL>
SQL> ALTER INDEX CUST_CUST_GENDER_IDX MONITORING USAGE;

Index altered.

SQL>
SQL> ALTER INDEX CUST_CUST_CREDIT_LIMIT_IDX MONITORING USAGE;

Index altered.

SQL>
SQL> @show_index_usage
SQL> set echo on
SQL>
SQL> set linesize 200
SQL>
SQL> select * from v$object_usage;

INDEX_NAME                     TABLE_NAME                     MON
USE START_MONITORING    END_MONITORING
------------------------------ ------------------------------ --- --
- ------------------- -------------------
```

```
CUSTOMERS_PK                     CUSTOMERS                        YES NO
04/09/2008 14:52:43
CUST_CUST_POSTAL_CODE_IDX        CUSTOMERS                        YES NO
04/09/2008 14:52:43
CUST_CUST_GENDER_IDX             CUSTOMERS                        YES NO
04/09/2008 14:52:43
CUST_CUST_CREDIT_LIMIT_IDX       CUSTOMERS                        YES NO
04/09/2008 14:52:43

SQL>
```

12) Execute the following query:

```
SELECT /*+ INDEX(c) */ c.*
FROM   customers c
WHERE  cust_gender   = 'M'
AND    cust_postal_code = 40804
AND   cust_credit_limit = 10000;
```

What do you observe?

a) The optimizer chooses to use only one index to do a fast full scan. The cost is lower than the full table scan.

```
SQL> @query01
SQL> set echo on
SQL>
SQL> set timing on
SQL> set autotrace traceonly
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT /*+ INDEX(c) */ c.*
  2  FROM   customers c
  3  WHERE  cust_gender   = 'M'
  4  AND    cust_postal_code = 40804
  5  AND   cust_credit_limit = 10000
  6  /

6 rows selected.

Elapsed: 00:00:00.04

Execution Plan
----------------------------------------------------------
Plan hash value: 1928091631

--------------------------------------------------------------------
-------------------------------------
| Id  | Operation                    | Name                       |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
-------------------------------------
|   0 | SELECT STATEMENT             |                            |
6 |  1080 |   218   (1)| 00:00:03 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                  |
6 |  1080 |   218   (1)| 00:00:03 |
|*  2 |   INDEX FULL SCAN            | CUST_CUST_POSTAL_CODE_IDX |
89 |       |   134   (1)| 00:00:02 |
```

```
-------------------------------------------------------------------
------------------------------------

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

   1 - filter("CUST_CREDIT_LIMIT"=10000 AND "CUST_GENDER"='M')
   2 - filter(TO_NUMBER("CUST_POSTAL_CODE")=40804)


Statistics
----------------------------------------------------------
        395  recursive calls
          3  db block gets
        384  consistent gets
        132  physical reads
        604  redo size
       2570  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
          6  rows processed

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

13) Execute the following query:

```
SELECT /*+ INDEX_COMBINE(c) */ c.*
FROM   customers c
WHERE  cust_gender   = 'M'
AND    cust_postal_code = 40804
AND   cust_credit_limit = 10000;
```

What do you observe?

a) This time the optimizer uses multiple indexes and combines them to access the table. However, the cost is higher than that from the previous step as well as the full table scan.

```
SQL> @query02
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT /*+ INDEX_COMBINE(c) */ c.*
  2  FROM   customers c
  3  WHERE  cust_gender   = 'M'
  4  AND    cust_postal_code = 40804
  5  AND   cust_credit_limit = 10000
  6  /

6 rows selected.
```

Practice 4-1: Using Different Access Paths (continued)

```
Elapsed: 00:00:00.06

Execution Plan
----------------------------------------------------------
Plan hash value: 4093665856

---------------------------------------------------------------------
-------------------------------------------
| Id  | Operation                        | Name
| Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------
-------------------------------------------
|   0 | SELECT STATEMENT                 |
|     6 |  1080 |   466   (1)| 00:00:06 |
|*  1 |  TABLE ACCESS BY INDEX ROWID     | CUSTOMERS
|     6 |  1080 |   466   (1)| 00:00:06 |
|   2 |   BITMAP CONVERSION TO ROWIDS    |
|       |       |            |          |
|   3 |    BITMAP AND                    |
|       |       |            |          |
|   4 |     BITMAP CONVERSION FROM ROWIDS|
|       |       |            |          |
|*  5 |      INDEX RANGE SCAN            |
CUST_CUST_CREDIT_LIMIT_IDX |       |       |    14   (0)| 00:00:01 |
|   6 |     BITMAP CONVERSION FROM ROWIDS|
|       |       |            |          |
|*  7 |      INDEX RANGE SCAN            | CUST_CUST_GENDER_IDX
|       |       |    51   (0)| 00:00:01 |
---------------------------------------------------------------------
-------------------------------------------

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

   1 - filter(TO_NUMBER("CUST_POSTAL_CODE")=40804)
   5 - access("CUST_CREDIT_LIMIT"=10000)
   7 - access("CUST_GENDER"='M')


Statistics
----------------------------------------------------------
          3  recursive calls
          7  db block gets
        894  consistent gets
         81  physical reads
       1020  redo size
       2570  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed

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

Practice 4-1: Using Different Access Paths (continued)

14) Confirm the list of indexes that were accessed in this case.

```
SQL> @show_index_usage
SQL> set echo on
SQL>
SQL> set linesize 200
SQL>
SQL> select * from v$object_usage;

INDEX_NAME                     TABLE_NAME                     MON
USE START_MONITORING    END_MONITORING
------------------------------ ------------------------------ --- --
- ------------------- -------------------
CUSTOMERS_PK                   CUSTOMERS                      YES NO
04/09/2008 14:52:43
CUST_CUST_POSTAL_CODE_IDX      CUSTOMERS                      YES
YES 04/09/2008 14:52:43
CUST_CUST_GENDER_IDX           CUSTOMERS                      YES
YES 04/09/2008 14:52:43
CUST_CUST_CREDIT_LIMIT_IDX     CUSTOMERS                      YES
YES 04/09/2008 14:52:43

SQL>
SQL>
```

15) Case 3: Drop all the CUSTOMERS indexes except its primary key index. After this, make sure you create a concatenated index on the following CUSTOMERS columns, and in the order mentioned here:

```
cust_gender
cust_credit_limit
cust_postal_code
```

```
SQL> @drop_customers_indexes
SQL> set    termout off
SQL>
SQL> @create_gender_limit_code_index
SQL> set echo on
SQL>
SQL> CREATE INDEX cust_gender_limit_code_idx
  2  ON customers(cust_gender,cust_credit_limit,cust_postal_code)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
```

16) Execute the following query:

```
SELECT /*+ INDEX(c) */ c.*
FROM   customers c
WHERE  cust_gender   = 'M'
AND    cust_postal_code = 40804
AND   cust_credit_limit = 10000;
```

What do you observe?

a) The optimizer uses your concatenated index, and the resulting cost is by far the best compared to the previous steps.

Practice 4-1: Using Different Access Paths (continued)

```
SQL> @query01
SQL> set echo on
SQL>
SQL> set timing on
SQL> set autotrace traceonly
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT /*+ INDEX(c) */ c.*
  2  FROM   customers c
  3  WHERE  cust_gender   = 'M'
  4  AND    cust_postal_code = 40804
  5  AND   cust_credit_limit = 10000
  6  /

6 rows selected.

Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------
Plan hash value: 2871279522

--------------------------------------------------------------------
--------------------------------------
| Id  | Operation                    | Name                        |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
--------------------------------------
|   0 | SELECT STATEMENT             |                             |
7 |  1260 |    18   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                   |
7 |  1260 |    18   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN           | CUST_GENDER_LIMIT_CODE_IDX |
6 |       |    12   (0)| 00:00:01 |
--------------------------------------------------------------------
--------------------------------------

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

   2 - access("CUST_GENDER"='M' AND "CUST_CREDIT_LIMIT"=10000)
       filter(TO_NUMBER("CUST_POSTAL_CODE")=40804)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
         22  consistent gets
         14  physical reads
          0  redo size
       2981  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed
```

Practice 4-1: Using Different Access Paths (continued)

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

17) Execute the following query:

```
SELECT /*+ INDEX(c) */ c.*
FROM   customers c
WHERE  cust_gender   = 'M'
AND   cust_credit_limit = 10000;
```

What do you observe?

a) The query is almost the same as in the previous step, but the predicate is removed. The optimizer can still use the concatenated index, but the resulting cost is much higher because `cust_credit_limit` is not very selective.

```
SQL> @query03
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT /*+ INDEX(c) */ c.*
  2  FROM   customers c
  3  WHERE  cust_gender   = 'M'
  4  AND   cust_credit_limit = 10000
  5  /

4101 rows selected.

Elapsed: 00:00:00.08

Execution Plan
----------------------------------------------------------
Plan hash value: 2871279522

--------------------------------------------------------------------
--------------------------------------
| Id  | Operation                   | Name                        |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
--------------------------------------
|   0 | SELECT STATEMENT            |                             |
3469 |   609K|  3454   (1)| 00:00:42 |
|   1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                   |
3469 |   609K|  3454   (1)| 00:00:42 |
|*  2 |   INDEX RANGE SCAN          | CUST_GENDER_LIMIT_CODE_IDX |
3469 |       |    12   (0)| 00:00:01 |
--------------------------------------------------------------------
--------------------------------------

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

```
   2 - access("CUST_GENDER"='M' AND "CUST_CREDIT_LIMIT"=10000)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       4391  consistent gets
          0  physical reads
          0  redo size
     795172  bytes sent via SQL*Net to client
       3423  bytes received via SQL*Net from client
        275  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       4101  rows processed

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

18) Execute the following query:

```
SELECT /*+ INDEX(c) */ c.*
FROM   customers c
WHERE cust_gender = 'M'
AND   cust_postal_code = 40804;
```

What do you observe?

a) You replaced `cust_credit_limit` with `cust_postal_code`, which has better selectivity. The index is used and the resulting cost is better.

```
SQL> @query04
SQL> set echo on
SQL>
SQL> set timing on
SQL> set autotrace traceonly
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT /*+ INDEX(c) */ c.*
  2  FROM   customers c
  3  WHERE cust_gender = 'M'
  4  AND   cust_postal_code = 40804
  5  /

75 rows selected.

Elapsed: 00:00:00.02

Execution Plan
----------------------------------------------------------
Plan hash value: 2871279522

--------------------------------------------------------------------
--------------------------------------
```

Practice 4-1: Using Different Access Paths (continued)

```
| Id  | Operation                   | Name                        |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
--------------------------------------
|   0 | SELECT STATEMENT            |                             |
45 |  8100 |   133   (1)| 00:00:02 |
|   1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                   |
45 |  8100 |   133   (1)| 00:00:02 |
|*  2 |   INDEX RANGE SCAN          | CUST_GENDER_LIMIT_CODE_IDX |
45 |       |    87   (0)| 00:00:02 |
--------------------------------------------------------------------
--------------------------------------

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

   2 - access("CUST_GENDER"='M')
       filter(TO_NUMBER("CUST_POSTAL_CODE")=40804)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        196  consistent gets
        101  physical reads
          0  redo size
      16237  bytes sent via SQL*Net to client
        464  bytes received via SQL*Net from client
          6  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         75  rows processed

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

19) Execute the following query:

```
SELECT /*+ INDEX(c) */ c.*
FROM   customers c
WHERE  cust_postal_code = 40804
AND   cust_credit_limit = 10000;
```

What do you observe?

a) The leading part of the concatenated index is no longer part of the query. However, the optimizer is still able to use the index by doing a fast full index scan. Nevertheless, the resulting cost is not the best.

```
SQL> @query05
SQL> set echo on
SQL>
SQL> set timing on
SQL> set autotrace traceonly
SQL> set linesize 200 pagesize 1000
SQL>
```

Practice 4-1: Using Different Access Paths (continued)

```
SQL> SELECT /*+ INDEX(c) */ c.*
  2  FROM   customers c
  3  WHERE  cust_postal_code = 40804
  4   AND   cust_credit_limit = 10000
  5  /

15 rows selected.

Elapsed: 00:00:00.02

Execution Plan
----------------------------------------------------------
Plan hash value: 2438361736

--------------------------------------------------------------------
--------------------------------------
| Id  | Operation                     | Name                        |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
--------------------------------------
|   0 | SELECT STATEMENT              |                             |
11 |  1980 |   185   (1)| 00:00:03 |
|   1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                   |
11 |  1980 |   185   (1)| 00:00:03 |
|*  2 |   INDEX FULL SCAN           | CUST_GENDER_LIMIT_CODE_IDX |
11 |       |   173   (1)| 00:00:03 |
--------------------------------------------------------------------
--------------------------------------

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

   2 - access("CUST_CREDIT_LIMIT"=10000)
       filter(TO_NUMBER("CUST_POSTAL_CODE")=40804 AND
"CUST_CREDIT_LIMIT"=10000)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        189  consistent gets
         56  physical reads
          0  redo size
       4617  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         15  rows processed

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

Practice 4-1: Using Different Access Paths (continued)

20) Case 4: Drop all the `CUSTOMERS` indexes except its primary key index. After this, create three different bitmap indexes on the following columns of the `CUSTOMERS` table:
`cust_gender`
`cust_postal_code`
`cust_credit_limit`

```
SQL> @drop_customers_indexes
SQL> set    termout off
SQL>
SQL> @create_cust_gender_bindex
SQL> set echo on
SQL>
SQL> CREATE BITMAP INDEX cust_cust_gender_bidx ON
customers(cust_gender)
  2  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL> @create_cust_postal_code_bindex
SQL> set echo on
SQL>
SQL> CREATE BITMAP INDEX cust_cust_postal_code_bidx ON
customers(cust_postal_code)
  2  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL> @create_cust_credit_limit_bindex
SQL> set echo on
SQL>
SQL> CREATE BITMAP INDEX cust_cust_credit_limit_bidx ON
customers(cust_credit_limit)
  2  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL> @list_customers_indexes
SQL> set echo on
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT      ui.table_name
  2  ,        decode(ui.index_type
  3                 ,'NORMAL', ui.uniqueness
  4                 ,ui.index_type) AS index_type
  5  ,        ui.index_name
  6  FROM        user_indexes  ui
  7  WHERE       ui.table_name = 'CUSTOMERS'
  8  ORDER BY ui.table_name
  9  ,        ui.uniqueness desc;

TABLE_NAME                     INDEX_TYPE
INDEX_NAME
```

Practice 4-1: Using Different Access Paths (continued)

```
------------------------------ -------------------------- ---------
---------------------
CUSTOMERS                      UNIQUE
CUSTOMERS_PK
CUSTOMERS                      BITMAP
CUST_CUST_GENDER_BIDX
CUSTOMERS                      BITMAP
CUST_CUST_CREDIT_LIMIT_BIDX
CUSTOMERS                      BITMAP
CUST_CUST_POSTAL_CODE_BIDX

SQL>
```

21) Execute the following query:

```
SELECT /*+ INDEX_COMBINE(c) */ c.*
FROM   customers c
WHERE  cust_gender    = 'M'
AND    cust_postal_code = 40804
AND   cust_credit_limit = 10000;
```

What do you observe?

a) The optimizer uses only two bitmap indexes to solve this query. However, the cost is not a good one. It is a little lesser than cost of the full table scan.

```
SQL> @query02
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT /*+ INDEX_COMBINE(c) */ c.*
  2  FROM   customers c
  3  WHERE  cust_gender    = 'M'
  4  AND    cust_postal_code = 40804
  5  AND   cust_credit_limit = 10000
  6  /

6 rows selected.

Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------
Plan hash value: 3047829365

--------------------------------------------------------------------
----------------------------------------
| Id  | Operation                     | Name                        |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
----------------------------------------
|   0 | SELECT STATEMENT              |                             |
6 |  1080 |   402   (1)| 00:00:05 |
|*  1 |  TABLE ACCESS BY INDEX ROWID | CUSTOMERS                   |
6 |  1080 |   402   (1)| 00:00:05 |
```

```
|   2 |   BITMAP CONVERSION TO ROWIDS|                               |
|       |            |          |
|   3 |    BITMAP AND            |                               |
|       |            |          |
|*  4 |     BITMAP INDEX SINGLE VALUE| CUST_CUST_CREDIT_LIMIT_BIDX |
|       |            |          |
|*  5 |     BITMAP INDEX SINGLE VALUE| CUST_CUST_GENDER_BIDX       |
|       |            |          |
--------------------------------------------------------------------
----------------------------------------

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

   1 - filter(TO_NUMBER("CUST_POSTAL_CODE")=40804)
   4 - access("CUST_CREDIT_LIMIT"=10000)
   5 - access("CUST_GENDER"='M')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        813  consistent gets
          5  physical reads
          0  redo size
       2570  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed

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

22) Case 5: Drop all the `CUSTOMERS` indexes except its primary key index. After this, create two bitmap indexes on the following columns of the `CUSTOMERS` table:
`cust_year_of_birth`
`cust_credit_limit`

```
SQL> @drop_customers_indexes
SQL> set    termout off
SQL>
SQL> @create_cust_year_of_birth_bindex
SQL> set echo on
SQL>
SQL> CREATE BITMAP INDEX cust_cust_year_of_birth_bidx
  2  ON customers(cust_year_of_birth)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
```

```
SQL> @create_cust_credit_limit_bindex
SQL> set echo on
SQL>
SQL> CREATE BITMAP INDEX cust_cust_credit_limit_bidx ON
customers(cust_credit_limit)
  2  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
```

23) Execute the following query:

```
SELECT /*+ INDEX_COMBINE(c) */ c.*
FROM   customers c
WHERE  c.cust_year_of_birth   = 1953
OR   c.cust_credit_limit  = 10000;
```

What do you observe?

a) The query uses an `OR` construct. The optimizer can use both bitmap indexes. However, this resulting cost is not the best.

```
SQL> @query06
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT /*+ INDEX_COMBINE(c) */ c.*
  2  FROM   customers c
  3  WHERE  c.cust_year_of_birth   = 1953
  4  OR   c.cust_credit_limit  = 10000
  5  /

7158 rows selected.

Elapsed: 00:00:00.12

Execution Plan
----------------------------------------------------------
Plan hash value: 1912490408

--------------------------------------------------------------------
-----------------------------------------
| Id  | Operation                     | Name
| Rows   | Bytes | Cost  (%CPU)| Time      |
--------------------------------------------------------------------
-----------------------------------------
|   0 | SELECT STATEMENT              |
|  7585 |  1333K|   581   (1)| 00:00:07 |
|   1 |  TABLE ACCESS BY INDEX ROWID | CUSTOMERS
|  7585 |  1333K|   581   (1)| 00:00:07 |
|   2 |   BITMAP CONVERSION TO ROWIDS|
|       |       |            |          |
|   3 |    BITMAP OR                 |
|       |       |            |          |
```

```
|*  4 |      BITMAP INDEX SINGLE VALUE| CUST_CUST_CREDIT_LIMIT_BIDX
|       |       |            |          |
|*  5 |      BITMAP INDEX SINGLE VALUE| CUST_CUST_YEAR_OF_BIRTH_BIDX
|       |       |            |          |
--------------------------------------------------------------------
-----------------------------------------

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

   4 - access("C"."CUST_CREDIT_LIMIT"=10000)
   5 - access("C"."CUST_YEAR_OF_BIRTH"=1953)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       1683  consistent gets
          3  physical reads
          0  redo size
    1391886  bytes sent via SQL*Net to client
       5667  bytes received via SQL*Net from client
        479  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       7158  rows processed

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

24) Case 6: Drop all the `CUSTOMERS` indexes except its primary key index. After this, create three different bitmap indexes on the following columns of the `CUSTOMERS` table:
`cust_year_of_birth`
`cust_credit_limit`
`cust_postal_code`

```
SQL> @drop_customers_indexes
SQL> set    termout off
SQL>
SQL> @create_cust_year_of_birth_bindex
SQL> set echo on
SQL>
SQL> CREATE BITMAP INDEX cust_cust_year_of_birth_bidx
  2  ON customers(cust_year_of_birth)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL> @create_cust_credit_limit_bindex
SQL> set echo on
```

```
SQL>
SQL> CREATE BITMAP INDEX cust_cust_credit_limit_bidx ON
customers(cust_credit_limit)
  2  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL> @create_cust_postal_code_bindex
SQL> set echo on
SQL>
SQL> CREATE BITMAP INDEX cust_cust_postal_code_bidx ON
customers(cust_postal_code)
  2  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
```

25) Execute the following query:

```
SELECT     c.*
FROM  customers c
WHERE  (c.cust_year_of_birth = '1970' And c.cust_postal_code  =
40804 )
AND NOT  cust_credit_limit = 15000;
```

What do you observe?

a) The query has a complex `WHERE` clause that is well suited for using bitmap indexes. The optimizer uses two bitmap indexes and the resulting cost is better than the full table scan cost.

```
SQL> @query07
SQL> set echo on
SQL>
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT     c.*
  2  FROM  customers c
  3  WHERE  (c.cust_year_of_birth = '1970' And c.cust_postal_code  =
40804 )
  4  AND NOT  cust_credit_limit = 15000
  5  /

Elapsed: 00:00:00.03

Execution Plan
----------------------------------------------------------
Plan hash value: 576122600

--------------------------------------------------------------------
-----------------------------------------
| Id  | Operation                     | Name
| Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
-----------------------------------------
```

```
|   0 | SELECT STATEMENT              |
     1 |   180 |   133   (0)| 00:00:02 |
|*  1 |  TABLE ACCESS BY INDEX ROWID  | CUSTOMERS
     1 |   180 |   133   (0)| 00:00:02 |
|   2 |   BITMAP CONVERSION TO ROWIDS |
|       |       |            |          |
|   3 |    BITMAP MINUS               |
|       |       |            |          |
|   4 |     BITMAP MINUS              |
|       |       |            |          |
|*  5 |      BITMAP INDEX SINGLE VALUE| CUST_CUST_YEAR_OF_BIRTH_BIDX
|       |       |            |          |
|*  6 |      BITMAP INDEX SINGLE VALUE| CUST_CUST_CREDIT_LIMIT_BIDX
|       |       |            |          |
|*  7 |     BITMAP INDEX SINGLE VALUE | CUST_CUST_CREDIT_LIMIT_BIDX
|       |       |            |          |
---------------------------------------------------------------------
------------------------------------------

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

   1 - filter(TO_NUMBER("C"."CUST_POSTAL_CODE")=40804)
   5 - access("C"."CUST_YEAR_OF_BIRTH"=1970)
   6 - access("CUST_CREDIT_LIMIT"=15000)
   7 - access("CUST_CREDIT_LIMIT" IS NULL)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        773  consistent gets
          3  physical reads
          0  redo size
       2024  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

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

26) Make sure the optimizer can no longer use the bitmap index you created on the `cust_year_of_birth` column.

a) The best solution is to render it invisible.

```
SQL> show parameter optimizer_use_invisible_indexes

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------------------
-----------
```

```
optimizer_use_invisible_indexes      boolean     FALSE
SQL> alter index cust_cust_year_of_birth_bidx invisible;

Index altered.

SQL> select index_name, visibility from user_indexes where
table_owner='SH' and table_name='CUSTOMERS';

INDEX_NAME                     VISIBILIT
------------------------------ ---------
CUST_CUST_CREDIT_LIMIT_BIDX    VISIBLE
CUST_CUST_YEAR_OF_BIRTH_BIDX   INVISIBLE
CUST_CUST_POSTAL_CODE_BIDX     VISIBLE
CUSTOMERS_PK                   VISIBLE

4 rows selected.

SQL>
```

27) Execute the following query:

```
SELECT     c.*
FROM  customers c
WHERE  (c.cust_year_of_birth = '1970' And c.cust_postal_code  =
40804 )
AND NOT  cust_credit_limit = 15000;
```

What do you observe?

a) This is the same query as in the previous step. However, the optimizer can no longer find a good plan that uses bitmap indexes.

```
SQL> @query07
SQL> set echo on
SQL>
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT     c.*
  2  FROM  customers c
  3  WHERE  (c.cust_year_of_birth = '1970' And c.cust_postal_code  =
40804 )
  4  AND NOT  cust_credit_limit = 15000
  5  /

Elapsed: 00:00:00.03

Execution Plan
----------------------------------------------------------
Plan hash value: 2008213504

--------------------------------------------------------------------
-----------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)|
Time     |
--------------------------------------------------------------------
-----------
|   0 | SELECT STATEMENT  |           |     1 |   180 |   407   (1)|
00:00:05 |
```

Practice 4-1: Using Different Access Paths (continued)

```
|*  1 |  TABLE ACCESS FULL| CUSTOMERS |     1 |   180 |   407   (1)|
00:00:05 |
--------------------------------------------------------------------
-----------

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

   1 - filter(TO_NUMBER("C"."CUST_POSTAL_CODE")=40804 AND
              "C"."CUST_YEAR_OF_BIRTH"=1970 AND
"CUST_CREDIT_LIMIT"<>15000)


Statistics
----------------------------------------------------------
        393  recursive calls
          0  db block gets
       1590  consistent gets
          0  physical reads
          0  redo size
       2024  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          6  sorts (memory)
          0  sorts (disk)
          1  rows processed

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

28) Case 7: Execute the following query:

```
SELECT c.*
FROM   customers c
WHERE  cust_id IN (88340,104590,44910);
```

What do you observe?

a) The optimizer can use the `CUSTOMERS` primary key index to solve this query. The cost is very low for the resulting plan.

```
SQL> @query08
SQL> set echo on
SQL>
SQL> set timing on
SQL> set linesize 200
SQL> set autotrace traceonly
SQL>
SQL> SELECT c.*
  2  FROM   customers c
  3  WHERE  cust_id IN (88340,104590,44910)
  4  /

Elapsed: 00:00:00.07

Execution Plan
----------------------------------------------------------
```

Practice 4-1: Using Different Access Paths (continued)

```
Plan hash value: 293792914

--------------------------------------------------------------------
-------------------------
| Id  | Operation                    | Name          | Rows  | Bytes
| Cost (%CPU)| Time     |
--------------------------------------------------------------------
-------------------------
|   0 | SELECT STATEMENT             |               |     3 |   540
|     7   (0)| 00:00:01 |
|   1 |  INLIST ITERATOR             |               |       |
|            |          |
|   2 |   TABLE ACCESS BY INDEX ROWID| CUSTOMERS     |     3 |   540
|     7   (0)| 00:00:01 |
|*  3 |    INDEX UNIQUE SCAN         | CUSTOMERS_PK |     3 |
|     4   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------------

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

   3 - access("CUST_ID"=44910 OR "CUST_ID"=88340 OR
"CUST_ID"=104590)


Statistics
----------------------------------------------------------
          2  recursive calls
          3  db block gets
          9  consistent gets
          4  physical reads
        560  redo size
       2022  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

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

29) Case 8: Drop all the indexes on the `CUSTOMERS` table except its primary key index. After this, create a concatenated B*-tree index on the following columns of the `CUSTOMERS` table, and in the order here:
`cust_last_name`
`cust_first_name`

```
SQL> @drop_customers_indexes
SQL> set    termout off
SQL>
SQL> @create_last_first_name_index
SQL> set echo on
```

```
SQL>
SQL> CREATE INDEX cust_last_first_name_idx
  2  ON customers(cust_last_name,cust_first_name)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
```

30) Execute the following query:

```
SELECT c.cust_last_name
,      c.cust_first_name
FROM   customers c;
```

What do you observe?

a) The optimizer can use only the index without accessing the table itself. The resulting cost is very good.

```
SQL> @query09
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT c.cust_last_name
  2  ,      c.cust_first_name
  3  FROM   customers c
  4  /

55500 rows selected.

Elapsed: 00:00:00.19

Execution Plan
----------------------------------------------------------
Plan hash value: 445338993

--------------------------------------------------------------------
-----------------------------
| Id  | Operation            | Name                       | Rows   |
Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
-----------------------------
|   0 | SELECT STATEMENT     |                            | 55500 |
812K|    55   (2)| 00:00:01 |
|   1 |  INDEX FAST FULL SCAN| CUST_LAST_FIRST_NAME_IDX | 55500 |
812K|    55   (2)| 00:00:01 |
--------------------------------------------------------------------
-----------------------------


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       3886  consistent gets
```

```
        193  physical reads
          0  redo size
     792248  bytes sent via SQL*Net to client
      41109  bytes received via SQL*Net from client
       3701  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
      55500  rows processed

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

31) Case 9: Drop all the indexes on the `CUSTOMERS` table except its primary key index. After this, create two B*-tree indexes on the following columns of the `CUSTOMERS` table:
`cust_last_name`
`cust_first_name`

```
SQL> @drop_customers_indexes
SQL> set    termout off
SQL>
SQL> @create_last_name_index
SQL> set echo on
SQL>
SQL> CREATE INDEX cust_cust_last_name_idx
  2  ON customers(cust_last_name)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL> @create_first_name_index
SQL> set echo on
SQL>
SQL> CREATE INDEX cust_cust_first_name_idx
  2  ON customers(cust_first_name)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
```

32) Execute the following query:

```
SELECT /*+ INDEX_JOIN(c cust_cust_first_name_idx
cust_cust_last_name_idx) */ c.cust_last_name
,      c.cust_first_name
FROM   customers c;
```

What do you observe?

a) Although the optimizer can use both the indexes, the resulting cost is not better than the concatenated index case.

Practice 4-1: Using Different Access Paths (continued)

```
SQL> @query10
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT /*+ INDEX_JOIN(c cust_cust_first_name_idx
cust_cust_last_name_idx) */ c.cust_last_name
  2  ,       c.cust_first_name
  3  FROM   customers c
  4  /

55500 rows selected.

Elapsed: 00:00:00.27

Execution Plan
----------------------------------------------------------
Plan hash value: 3557918892

--------------------------------------------------------------------
-------------------------------
| Id  | Operation               | Name                       | Rows  |
Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
-------------------------------
|   0 | SELECT STATEMENT        |                            | 55500 |
812K|   511   (1)| 00:00:07 |
|   1 |  VIEW                   | index$_join$_001           | 55500 |
812K|   511   (1)| 00:00:07 |
|*  2 |   HASH JOIN             |                            |       |
|            |          |
|   3 |    INDEX FAST FULL SCAN| CUST_CUST_FIRST_NAME_IDX | 55500 |
812K|   175   (1)| 00:00:03 |
|   4 |    INDEX FAST FULL SCAN| CUST_CUST_LAST_NAME_IDX  | 55500 |
812K|   178   (0)| 00:00:03 |
--------------------------------------------------------------------
-------------------------------

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

   2 - access(ROWID=ROWID)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       3985  consistent gets
        279  physical reads
          0  redo size
     811440  bytes sent via SQL*Net to client
      41109  bytes received via SQL*Net from client
       3701  SQL*Net roundtrips to/from client
          0  sorts (memory)
```

Practice 4-1: Using Different Access Paths (continued)

```
         0  sorts (disk)
     55500  rows processed

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

33) CASE 10: Drop all the indexes on the `CUSTOMERS` table except its primary key index. Then, create one bitmap index on the following column of the `CUSTOMERS` table:
`cust_credit_limit`

```
SQL> @drop_customers_indexes
SQL> set    termout off
SQL>
SQL> @create_cust_credit_limit_bindex
SQL> set echo on
SQL>
SQL> CREATE BITMAP INDEX cust_cust_credit_limit_bidx ON
customers(cust_credit_limit)
  2  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
```

34) Execute the following query:

```
SELECT count(*)  credit_limit
FROM   customers
WHERE  cust_credit_limit = 10000;
```

What do you observe?

a) Although `cust_credit_limit` is not a selective column, the `COUNT` operation on its bitmap index is very efficient.

```
SQL> @query11
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT count(*)  credit_limit
  2  FROM   customers
  3  WHERE  cust_credit_limit = 10000
  4  /

Elapsed: 00:00:00.00

Execution Plan
----------------------------------------------------------
Plan hash value: 37937133
```

```
--------------------------------------------------------------------
---------------------------------------
| Id  | Operation                    | Name                        |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
---------------------------------------
|   0 | SELECT STATEMENT             |                             |
1 |     4 |     1   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |                             |
1 |     4 |            |          |
|   2 |   BITMAP CONVERSION COUNT    |                             |
6938 | 27752 |     1   (0)| 00:00:01 |
|*  3 |    BITMAP INDEX SINGLE VALUE| CUST_CUST_CREDIT_LIMIT_BIDX |
|       |             |          |
--------------------------------------------------------------------
---------------------------------------

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

   3 - access("CUST_CREDIT_LIMIT"=10000)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
          3  consistent gets
          2  physical reads
          0  redo size
        423  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

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

35) Case 11: Drop all the CUSTOMERS indexes except its primary key index. After this, create one B*-tree index on the following column of the CUSTOMERS table: cust_credit_limit

```
SQL> @drop_customers_indexes
SQL> set    termout off
SQL>
SQL> @create_cust_credit_limit_index
SQL> set echo on
SQL>
SQL> CREATE INDEX cust_cust_credit_limit_idx
  2  ON customers(cust_credit_limit)
  3  NOLOGGING COMPUTE STATISTICS;
```

Practice 4-1: Using Different Access Paths (continued)

```
Index created.

SQL>
```

36) Execute the following query:

```
SELECT count(*)  credit_limit
FROM   customers
WHERE  cust_credit_limit = 10000;
```

What do you observe?

a) The optimizer can still use the index; however, this is less efficient compared to the corresponding bitmap index from the previous case.

```
SQL> @query11
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT count(*)  credit_limit
  2  FROM   customers
  3  WHERE  cust_credit_limit = 10000
  4  /

Elapsed: 00:00:00.00

Execution Plan
----------------------------------------------------------
Plan hash value: 3421880850

--------------------------------------------------------------------
----------------------------
| Id  | Operation         | Name                        | Rows  |
Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
----------------------------
|   0 | SELECT STATEMENT  |                             |     1 |
4 |    14   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE   |                             |     1 |
4 |            |          |
|*  2 |   INDEX RANGE SCAN| CUST_CUST_CREDIT_LIMIT_IDX |  6938 |
27752 |    14   (0)| 00:00:01 |
--------------------------------------------------------------------
----------------------------

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

   2 - access("CUST_CREDIT_LIMIT"=10000)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
```

Practice 4-1: Using Different Access Paths (continued)

```
         14  consistent gets
         13  physical reads
          0  redo size
        423  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

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

37) Case 12: Drop all the indexes on the `CUSTOMERS` table except its primary key index. After this, create one B*-tree index on the following column of the `CUSTOMERS` table:
`cust_last_name`

```
SQL> @drop_customers_indexes
SQL> set    termout off
SQL>
SQL> @create_last_name_index
SQL> set echo on
SQL>
SQL> CREATE INDEX cust_cust_last_name_idx
  2  ON customers(cust_last_name)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
```

38) Execute the following query:
```
SELECT cust_id, country_id
FROM customers
WHERE LOWER( cust_last_name) LIKE 'gentle';
```

What do you observe?

a) Although there is an index, it cannot be used because its column is modified by a function.

```
SQL> @query12
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT cust_id, country_id
  2  FROM customers
  3  WHERE LOWER( cust_last_name) LIKE 'gentle'
  4  /
```

```
80 rows selected.

Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------
Plan hash value: 2008213504

---------------------------------------------------------------------
-----------
| Id  | Operation          | Name       | Rows  | Bytes | Cost (%CPU)|
Time     |
---------------------------------------------------------------------
-----------
|   0 | SELECT STATEMENT  |            |   555 |  9990 |   406   (1)|
00:00:05 |
|*  1 |  TABLE ACCESS FULL| CUSTOMERS |   555 |  9990 |   406   (1)|
00:00:05 |
---------------------------------------------------------------------
-----------

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

   1 - filter(LOWER("CUST_LAST_NAME")='gentle')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       1464  consistent gets
          0  physical reads
          0  redo size
       2077  bytes sent via SQL*Net to client
        475  bytes received via SQL*Net from client
          7  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
         80  rows processed

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

39) How can you enhance the performance of the previous query without modifying the statement itself? Implement your solution.

a) You can create a function-based index.

```
SQL> @create_lower_cust_last_name_index
SQL> set echo on
SQL>
SQL> CREATE INDEX lower_cust_last_name_idx ON
  2  customers(LOWER(cust_last_name))
  3  /
```

Practice 4-1: Using Different Access Paths (continued)

```
Index created.

SQL>
```

40) Check if your solution executes faster than in the case of the query in step 38.

```
SQL> @query12
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT cust_id, country_id
  2  FROM customers
  3  WHERE LOWER( cust_last_name) LIKE 'gentle'
  4  /

80 rows selected.

Elapsed: 00:00:00.00

Execution Plan
----------------------------------------------------------
Plan hash value: 967065894

--------------------------------------------------------------------
------------------------------------
| 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_IDX |
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
         23  consistent gets
          1  physical reads
          0  redo size
       2077  bytes sent via SQL*Net to client
        475  bytes received via SQL*Net from client
          7  SQL*Net roundtrips to/from client
```

Practice 4-1: Using Different Access Paths (continued)

```
          0  sorts (memory)
          0  sorts (disk)
         80  rows processed

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

41) Case 13: Execute the `iot_setup.sql` script to set up the environment for this case.

```
SQL> @iot_setup
SQL> set echo on
SQL>
SQL> CREATE table promotions_iot
  2  (promo_id number primary key
  3  ,  promo_name VARCHAR2(40)
  4  ,  promo_subcategory VARCHAR2 (30)
  5  ,  promo_category  VARCHAR2 (30)
  6  ,  promo_cost NUMBER
  7  ,  promo_begin_date DATE
  8  ,  promo_end_date DATE)
  9  ORGANIZATION INDEX
 10  /

Table created.

SQL>
SQL> INSERT INTO promotions_iot
  2  SELECT promo_id, promo_name, promo_subcategory, promo_category,
promo_cost, promo_begin_date, promo_end_date
  3  FROM promotions
  4  /

503 rows created.

SQL>
```

42) Execute the following two queries:

```
SELECT *
FROM promotions
WHERE promo_id > 300;

SELECT /*+ INDEX(promotions) */ *
FROM promotions
WHERE promo_id > 300;
```

What do you observe?

a) The first lets the optimizer decide the plan, and the best it can find is to do a full table scan. Forcing the use of the index is not a good idea, as it takes longer to execute.

```
SQL> @query13
SQL> set echo on
SQL>
```

```
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT *
  2  FROM promotions
  3  WHERE promo_id > 300
  4  /

235 rows selected.

Elapsed: 00:00:00.02

Execution Plan
----------------------------------------------------------
Plan hash value: 4106015420

----------------------------------------------------------------------
------------
| Id  | Operation          | Name        | Rows   | Bytes | Cost
(%CPU)| Time     |
----------------------------------------------------------------------
------------
|   0 | SELECT STATEMENT  |             |   364 | 35308 |    17
(0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| PROMOTIONS |   364 | 35308 |    17
(0)| 00:00:01 |
----------------------------------------------------------------------
------------

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

   1 - filter("PROMO_ID">300)


Statistics
----------------------------------------------------------
        123  recursive calls
          0  db block gets
         88  consistent gets
          3  physical reads
          0  redo size
      21829  bytes sent via SQL*Net to client
        585  bytes received via SQL*Net from client
         17  SQL*Net roundtrips to/from client
          4  sorts (memory)
          0  sorts (disk)
        235  rows processed

SQL>
SQL> SELECT /*+ INDEX(promotions) */ *
  2  FROM promotions
  3  WHERE promo_id > 300
  4  /

235 rows selected.
```

Practice 4-1: Using Different Access Paths (continued)

```
Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------
Plan hash value: 4044283270

--------------------------------------------------------------------
----------------------
| Id  | Operation                   | Name       | Rows  | Bytes |
Cost (%CPU)| Time     |
--------------------------------------------------------------------
----------------------
|   0 | SELECT STATEMENT            |            |   364 | 35308 |
353   (0)| 00:00:05 |
|   1 |  TABLE ACCESS BY INDEX ROWID| PROMOTIONS |   364 | 35308 |
353   (0)| 00:00:05 |
|*  2 |   INDEX RANGE SCAN          | PROMO_PK   |   364 |       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
----------------------

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

   2 - access("PROMO_ID">300)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        243  consistent gets
          1  physical reads
          0  redo size
      27052  bytes sent via SQL*Net to client
        585  bytes received via SQL*Net from client
         17  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        235  rows processed

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

43) Execute the following query:

```
SELECT *
FROM promotions_iot
WHERE promo_id > 300;
```

What do you observe?

a) The optimizer directly uses the index-organized structure, which is extremely efficient in this case compared to the previous step.

```
SQL> @query14
```

Practice 4-1: Using Different Access Paths (continued)

```
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> SELECT *
  2  FROM promotions_iot
  3  WHERE promo_id > 300
  4  /

235 rows selected.

Elapsed: 00:00:00.00

Execution Plan
----------------------------------------------------------
Plan hash value: 1463021396

---------------------------------------------------------------------
------------------
| Id  | Operation        | Name              | Rows  | Bytes | Cost
(%CPU)| Time     |
---------------------------------------------------------------------
------------------
|   0 | SELECT STATEMENT |                   |   235 | 23500 |     2
(0)| 00:00:01 |
|*  1 |  INDEX RANGE SCAN| SYS_IOT_TOP_72170 |   235 | 23500 |     2
(0)| 00:00:01 |
---------------------------------------------------------------------
------------------

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

   1 - access("PROMO_ID">300)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
         42  consistent gets
          0  physical reads
        116  redo size
      19922  bytes sent via SQL*Net to client
        585  bytes received via SQL*Net from client
         17  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        235  rows processed

SQL>
```

Practice 4-1: Using Different Access Paths (continued)

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

44) Execute the `iot_cleanup.sql` script to clean up your environment.

```
SQL> @iot_cleanup
SQL> set echo on
SQL>
SQL> drop table promotions_iot purge;

Table dropped.

SQL>
SQL>
```

45) Case 14: Execute `shc_setup.sql` to set up your environment for this lab.

```
SQL> @shc_setup
SQL> set echo on
SQL>
SQL> set linesize 200
SQL>
SQL> drop cluster bigemp_cluster including tables;
drop cluster bigemp_cluster including tables
*
ERROR at line 1:
ORA-00943: cluster does not exist


SQL>
SQL> CREATE CLUSTER bigemp_cluster
  2  (deptno number, sal number sort)
  3  HASHKEYS 10000
  4  single table HASH IS deptno SIZE 50
  5  tablespace users;

Cluster created.

SQL>
SQL> create table bigemp_fact (
  2  empno number primary key, sal number sort, job varchar2(12) not
null,
  3  deptno number not null, hiredate date not null)
  4  CLUSTER bigemp_cluster (deptno, sal);

Table created.

SQL>
SQL>
SQL> begin
  2  for i in 1..1400000 loop
  3   insert into bigemp_fact values(i,i,'J1',10,sysdate);
  4  end loop;
  5  commit;
  6  end;
  7  /
```

```
PL/SQL procedure successfully completed.

SQL>
SQL> begin
  2  for i in 1..1400000 loop
  3   insert into bigemp_fact values(1400000+i,i,'J1',20,sysdate);
  4  end loop;
  5  commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> exec dbms_stats.gather_schema_stats('SH');

PL/SQL procedure successfully completed.

SQL>
```

46) Execute the `query15.sql` script. What do you observe?

a) Because you may have a lot of memory on your system, the script first reduces the amount of memory available to your session. The optimizer decides to use the cluster access path to retrieve the data. The cost is minimal.

```
SQL> @query15
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> alter session set workarea_size_policy=manual;

Session altered.

Elapsed: 00:00:00.02
SQL> alter session set sort_area_size=50000;

Session altered.

Elapsed: 00:00:00.01
SQL>
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.16
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:06.69
SQL>
SQL>
```

Practice 4-1: Using Different Access Paths (continued)

```
SQL> select * from bigemp_fact where deptno=10;

1400000 rows selected.

Elapsed: 00:00:07.21

Execution Plan
----------------------------------------------------------
Plan hash value: 865757019

--------------------------------------------------------------------
-------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
-------------
|   0 | SELECT STATEMENT  |              |  1400K|    32M|     1
(0)| 00:00:01 |
|*  1 |  TABLE ACCESS HASH| BIGEMP_FACT |  1400K|    32M|     1
(0)| 00:00:01 |
--------------------------------------------------------------------
-------------

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

   1 - access("DEPTNO"=10)


Statistics
----------------------------------------------------------
        689  recursive calls
          0  db block gets
      99451  consistent gets
       5999  physical reads
        116  redo size
   33605911  bytes sent via SQL*Net to client
    1027083  bytes received via SQL*Net from client
      93335  SQL*Net roundtrips to/from client
         10  sorts (memory)
          0  sorts (disk)
    1400000  rows processed

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

47) Execute the `query16.sql` script. What do you observe?

   a) Again, the script first ensures that the amount of memory available to your session is reduced. Then the script executes the same query, but asks for ordering the result based on the sorted `sal` column. The optimizer can still use the cluster access path without sorting the data. The cost is still minimal.

```
SQL> @query16
SQL> set echo on
```

Practice 4-1: Using Different Access Paths (continued)

```
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> alter session set workarea_size_policy=manual;

Session altered.

Elapsed: 00:00:00.00
SQL> alter session set sort_area_size=50000;

Session altered.

Elapsed: 00:00:00.00
SQL>
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.10
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.04
SQL>
SQL> select * from bigemp_fact where deptno=10 order by sal;

1400000 rows selected.

Elapsed: 00:00:07.41

Execution Plan
----------------------------------------------------------
Plan hash value: 865757019

---------------------------------------------------------------------
-------------
| Id  | Operation          | Name         | Rows  | Bytes | Cost
(%CPU)| Time     |
---------------------------------------------------------------------
-------------
|   0 | SELECT STATEMENT  |              |  1400K|    32M|     1
(0)| 00:00:01 |
|*  1 |  TABLE ACCESS HASH| BIGEMP_FACT |  1400K|    32M|     1
(0)| 00:00:01 |
---------------------------------------------------------------------
-------------

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

   1 - access("DEPTNO"=10)


Statistics
```

Practice 4-1: Using Different Access Paths (continued)

```
----------------------------------------------------------
       1090  recursive calls
         10  db block gets
      99512  consistent gets
       6012  physical reads
          0  redo size
   33605911  bytes sent via SQL*Net to client
    1027083  bytes received via SQL*Net from client
      93335  SQL*Net roundtrips to/from client
         12  sorts (memory)
          0  sorts (disk)
    1400000  rows processed

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

48) Execute the `query17.sql` script. What do you observe?

a) Again, the script first ensures that the amount of memory available to your session is reduced. Then the script executes the same query, but asks to order the result based on the sorted `sal` column in the descending order. The optimizer can still use the cluster access path without sorting the data. The cost is still minimal.

```
SQL> @query17
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> alter session set workarea_size_policy=manual;

Session altered.

Elapsed: 00:00:00.00
SQL> alter session set sort_area_size=50000;

Session altered.

Elapsed: 00:00:00.00
SQL>
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.09
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.12
SQL>
SQL> select * from bigemp_fact where deptno=10 order by sal desc;

1400000 rows selected.
```

```
Elapsed: 00:00:07.35

Execution Plan
----------------------------------------------------------
Plan hash value: 865757019

--------------------------------------------------------------------
-------------
| Id  | Operation         | Name        | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
-------------
|   0 | SELECT STATEMENT  |             |  1400K|    32M|     1
(0)| 00:00:01 |
|*  1 |  TABLE ACCESS HASH| BIGEMP_FACT |  1400K|    32M|     1
(0)| 00:00:01 |
--------------------------------------------------------------------
-------------

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

   1 - access("DEPTNO"=10)


Statistics
----------------------------------------------------------
       1090  recursive calls
         10  db block gets
      99509  consistent gets
       6005  physical reads
          0  redo size
   33605911  bytes sent via SQL*Net to client
    1027083  bytes received via SQL*Net from client
      93335  SQL*Net roundtrips to/from client
         12  sorts (memory)
          0  sorts (disk)
    1400000  rows processed

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

49) Execute the `query18.sql` script. What do you observe?

a) Again, the script first ensures that the amount of memory available to your session is reduced. Then the script executes the same query but this time asks to order the result based on the nonsorted `empno` column. The optimizer can still make use of the cluster access path, but must sort the data making the cost of the query higher.

```
SQL> @query18
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
```

Practice 4-1: Using Different Access Paths (continued)

```
SQL> set autotrace traceonly
SQL>
SQL> alter session set workarea_size_policy=manual;

Session altered.

Elapsed: 00:00:00.00
SQL> alter session set sort_area_size=50000;

Session altered.

Elapsed: 00:00:00.00
SQL>
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.10
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.04
SQL>
SQL> select * from bigemp_fact where deptno=10 order by empno;

1400000 rows selected.

Elapsed: 00:00:10.01

Execution Plan
----------------------------------------------------------
Plan hash value: 1775608660

--------------------------------------------------------------------
----------------------
| Id  | Operation          | Name         | Rows   | Bytes |TempSpc|
Cost (%CPU)| Time     |
--------------------------------------------------------------------
----------------------
|   0 | SELECT STATEMENT   |              |  1400K|    32M|       |
47728   (1)| 00:09:33 |
|   1 |  SORT ORDER BY     |              |  1400K|    32M|   107M|
47728   (1)| 00:09:33 |
|*  2 |   TABLE ACCESS HASH| BIGEMP_FACT |  1400K|    32M|       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
----------------------

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

   2 - access("DEPTNO"=10)


Statistics
----------------------------------------------------------
```

```
       1139  recursive calls
         12  db block gets
       6178  consistent gets
      12238  physical reads
          0  redo size
   33605911  bytes sent via SQL*Net to client
    1027083  bytes received via SQL*Net from client
      93335  SQL*Net roundtrips to/from client
         12  sorts (memory)
          1  sorts (disk)
    1400000  rows processed

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

50) Execute the `query19.sql` script. What do you observe?

a) Again, the script first ensures that the amount of memory available to your session is reduced. Then the script executes the same query, but this time asks to order the result based on the `sal`, `deptno` key. The optimizer can still make use of the cluster access path, but must sort the data making the cost of the query higher.

```
SQL> @query19
SQL> set echo on
SQL>
SQL> set linesize 200
SQL> set timing on
SQL> set autotrace traceonly
SQL>
SQL> alter session set workarea_size_policy=manual;

Session altered.

Elapsed: 00:00:00.00
SQL> alter session set sort_area_size=50000;

Session altered.

Elapsed: 00:00:00.00
SQL>
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.10
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.09
SQL>
SQL> select * from bigemp_fact where deptno=10 order by sal,empno;

1400000 rows selected.
```

Practice 4-1: Using Different Access Paths (continued)

```
Elapsed: 00:00:09.25

Execution Plan
----------------------------------------------------------
Plan hash value: 1775608660

--------------------------------------------------------------------
----------------------
| Id  | Operation          | Name         | Rows  | Bytes |TempSpc|
Cost (%CPU)| Time     |
--------------------------------------------------------------------
----------------------
|   0 | SELECT STATEMENT   |              |  1400K|    32M|       |
47728   (1)| 00:09:33 |
|   1 |  SORT ORDER BY     |              |  1400K|    32M|   107M|
47728   (1)| 00:09:33 |
|*  2 |   TABLE ACCESS HASH| BIGEMP_FACT |  1400K|    32M|       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
----------------------

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

   2 - access("DEPTNO"=10)


Statistics
----------------------------------------------------------
       1139  recursive calls
         12  db block gets
       6178  consistent gets
      12238  physical reads
          0  redo size
   33605911  bytes sent via SQL*Net to client
    1027083  bytes received via SQL*Net from client
      93335  SQL*Net roundtrips to/from client
         12  sorts (memory)
          1  sorts (disk)
    1400000  rows processed

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

51) Execute the `shc_cleanup.sql` script to clean up your environment.

```
SQL> @shc_cleanup
SQL> set echo on
SQL>
SQL> drop cluster bigemp_cluster including tables;

Cluster dropped.

SQL>
SQL>
```

Practice 4-1: Using Different Access Paths (continued)

52) Case 15: Execute the `nic_setup.sql` script to set up your environment for this case.

```
SQL> @nic_setup
SQL> set echo on
SQL>
SQL> drop cluster emp_dept including tables;
drop cluster emp_dept including tables
*
ERROR at line 1:
ORA-00943: cluster does not exist


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


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


SQL>
SQL>
SQL> CREATE TABLE emp (
  2     empno     NUMBER(7)             ,
  3     ename     VARCHAR2(15) NOT NULL,
  4     job       VARCHAR2(9)           ,
  5     mgr       NUMBER(7)             ,
  6     hiredate  DATE                  ,
  7     sal       NUMBER(7)           ,
  8     comm      NUMBER(7)           ,
  9     deptno    NUMBER(3)
 10  );

Table created.

SQL>
SQL> CREATE TABLE dept (
  2     deptno NUMBER(3) ,
  3     dname  VARCHAR2(14),
  4     loc    VARCHAR2(14),
  5     c      VARCHAR2(500)
  6  );

Table created.

SQL>
SQL> CREATE INDEX emp_index
  2     ON emp(deptno)
  3     TABLESPACE users
```

```
  4     STORAGE (INITIAL 50K
  5        NEXT 50K
  6        MINEXTENTS 2
  7        MAXEXTENTS 10
  8        PCTINCREASE 33);

Index created.

SQL>
SQL> CREATE INDEX dept_index
  2     ON dept(deptno)
  3     TABLESPACE users
  4     STORAGE (INITIAL 50K
  5        NEXT 50K
  6        MINEXTENTS 2
  7        MAXEXTENTS 10
  8        PCTINCREASE 33);

Index created.

SQL>
SQL>
SQL> begin
  2   for i in 1..999 loop
  3    insert into dept values
(i,'D'||i,'L'||i,dbms_random.string('u',500));
  4   end loop;
  5   commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> begin
  2   for i in 1..500000 loop
  3    insert into emp values
(i,dbms_random.string('u',15),dbms_random.string('u',9),i,sysdate,i,
i,mod(i,999));
  4   end loop;
  5   commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL> exec dbms_stats.gather_schema_stats('SH');

PL/SQL procedure successfully completed.

SQL>
```

53) Execute the `nic_query.sql` script. What do you observe?

Practice 4-1: Using Different Access Paths (continued)

a) The script first ensures that the amount of memory available to your session is reduced. Then the script executes a join between the `EMP` and `DEPT` tables. The optimizer is able to make use of the index to resolve the join.

```
SQL> @nic_query
SQL> set echo on
SQL>
SQL> set timing on
SQL> set autotrace traceonly
SQL> set linesize 200
SQL>
SQL> alter session set workarea_size_policy=manual;

Session altered.

Elapsed: 00:00:00.00
SQL> alter session set sort_area_size=50000;

Session altered.

Elapsed: 00:00:00.00
SQL> alter session set hash_area_size=5000;

Session altered.

Elapsed: 00:00:00.00
SQL>
SQL>
SQL> select * from emp,dept where emp.deptno=dept.deptno and
emp.deptno > 800;

99000 rows selected.

Elapsed: 00:00:02.88

Execution Plan
----------------------------------------------------------
Plan hash value: 128236434

---------------------------------------------------------------------
-------------------------------
| Id  | Operation                     | Name        | Rows  | Bytes
|TempSpc| Cost (%CPU)| Time     |
---------------------------------------------------------------------
-------------------------------
|   0 | SELECT STATEMENT              |             | 19780 |    10M|
|  3449   (1)| 00:00:42 |
|*  1 |  HASH JOIN                    |             | 19780 |    10M|
104K|  3449   (1)| 00:00:42 |
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPT        |   199 |    99K|
|    18   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN           | DEPT_INDEX |   199 |       |
|     2   (0)| 00:00:01 |
|*  4 |   TABLE ACCESS FULL           | EMP         | 99198 |  5521K|
|  1207   (2)| 00:00:15 |
---------------------------------------------------------------------
-------------------------------
```

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

   1 - access("EMP"."DEPTNO"="DEPT"."DEPTNO")
   3 - access("DEPT"."DEPTNO">800)
   4 - filter("EMP"."DEPTNO">800)


Statistics
----------------------------------------------------------
          9  recursive calls
          0  db block gets
       4365  consistent gets
      10146  physical reads
          0  redo size
   57968984  bytes sent via SQL*Net to client
      73009  bytes received via SQL*Net from client
       6601  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
      99000  rows processed

SQL>
SQL> set autotrace off
SQL> set timing off;
SQL>
```

54) How would you enhance the performance of the previous query? Implement your solution.

a) Create a cluster to store the two tables.

```
SQL> @ic_setup
SQL> set echo on
SQL>
SQL> drop table emp purge;

Table dropped.

SQL> drop table dept purge;

Table dropped.

SQL>
SQL> drop cluster emp_dept including tables;
drop cluster emp_dept including tables
*
ERROR at line 1:
ORA-00943: cluster does not exist


SQL>
SQL> CREATE CLUSTER emp_dept (deptno NUMBER(3))
  2     SIZE 600
  3     TABLESPACE users
  4     STORAGE (INITIAL 200K
  5        NEXT 300K
```

Practice 4-1: Using Different Access Paths (continued)

```
  6        MINEXTENTS 2
  7        PCTINCREASE 33);

Cluster created.

SQL>
SQL> CREATE TABLE emp (
  2     empno     NUMBER(7)            ,
  3     ename     VARCHAR2(15) NOT NULL,
  4     job       VARCHAR2(9)          ,
  5     mgr       NUMBER(7)            ,
  6     hiredate  DATE                 ,
  7     sal       NUMBER(7)          ,
  8     comm      NUMBER(7)          ,
  9     deptno    NUMBER(3))
 10     CLUSTER emp_dept (deptno);

Table created.

SQL>
SQL> CREATE TABLE dept (
  2     deptno NUMBER(3) ,
  3     dname  VARCHAR2(14),
  4     loc    VARCHAR2(14),
  5     c      VARCHAR2(500))
  6     CLUSTER emp_dept (deptno);

Table created.

SQL>
SQL> CREATE INDEX emp_dept_index
  2     ON CLUSTER emp_dept
  3     TABLESPACE users
  4     STORAGE (INITIAL 50K
  5        NEXT 50K
  6        MINEXTENTS 2
  7        MAXEXTENTS 10
  8        PCTINCREASE 33);

Index created.

SQL>
SQL> begin
  2   for i in 1..999 loop
  3    insert into dept values
(i,'D'||i,'L'||i,dbms_random.string('u',500));
  4   end loop;
  5   commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> begin
  2   for i in 1..500000 loop
```

```
  3    insert into emp values
(i,dbms_random.string('u',15),dbms_random.string('u',9),i,sysdate,i,
i,mod(i,999));
  4   end loop;
  5   commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL> exec dbms_stats.gather_schema_stats('SH');

PL/SQL procedure successfully completed.

SQL>
```

55) Execute the following query to confirm the performance enhancement of the previous query:

```
select *
from emp,dept
where emp.deptno=dept.deptno and emp.deptno > 800;
```

What do you observe?

a) The optimizer is able to use the cluster access path that makes the query execute faster.

```
SQL> @ic_query
SQL> set echo on
SQL>
SQL> set timing on
SQL> set autotrace traceonly
SQL> set linesize 200
SQL>
SQL> select * from emp,dept where emp.deptno=dept.deptno and
emp.deptno > 800;

99000 rows selected.

Elapsed: 00:00:01.20

Execution Plan
----------------------------------------------------------
Plan hash value: 593050162

--------------------------------------------------------------------
--------------------
| Id  | Operation              | Name             | Rows  | Bytes |
Cost (%CPU)| Time     |
--------------------------------------------------------------------
--------------------
|   0 | SELECT STATEMENT       |                  | 19780 |    10M|
11515   (1)| 00:02:19 |
|   1 |  NESTED LOOPS          |                  | 19780 |    10M|
11515   (1)| 00:02:19 |
|   2 |   TABLE ACCESS CLUSTER| DEPT              |   199 |    99K|
167   (0)| 00:00:03 |
```

```
|*  3 |     INDEX RANGE SCAN    | EMP_DEPT_INDEX |      1 |         |
2   (0)| 00:00:01 |
|*  4 |    TABLE ACCESS CLUSTER| EMP            |     99 |  5643 |
57   (0)| 00:00:01 |
--------------------------------------------------------------------
--------------------

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

   3 - access("DEPT"."DEPTNO">800)
   4 - filter("EMP"."DEPTNO">800 AND "EMP"."DEPTNO"="DEPT"."DEPTNO")


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
      30131  consistent gets
          0  physical reads
          0  redo size
    6259293  bytes sent via SQL*Net to client
      73009  bytes received via SQL*Net from client
       6601  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
      99000  rows processed

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

56) Execute the `ic_cleanup.sql` script to clean up your environment for this case.

```
SQL> @ic_cleanup
SQL> set echo on
SQL>
SQL> drop cluster emp_dept including tables;

Cluster dropped.

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


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


SQL>
SQL>
```

Practice 4-1: Using Different Access Paths (continued)

57) Case 16: Execute the `iss_setup.sql` script to set up your environment for this lab.

```
SQL> @iss_setup
SQL> set echo on
SQL>
SQL> create table t(c number, d number);

Table created.

SQL>
SQL> begin
  2   for i in 1..10000 loop
  3     insert into t values(1,i);
  4   end loop;
  5  end;
  6  /

PL/SQL procedure successfully completed.

SQL>
SQL> create index it on t(c,d);

Index created.

SQL>
```

58) Execute the following query:
`select count(*) from t where d=1;`

What do you observe?

a) The optimizer is not using the index and does a full table scan.

```
SQL> @query20
SQL> set echo on
SQL>
SQL> set linesize 200
SQL>
SQL> set timing on
SQL> set autotrace on
SQL>
SQL> select count(*) from t where d=1;

  COUNT(*)
----------
         1

Elapsed: 00:00:00.01

Execution Plan
----------------------------------------------------------
Plan hash value: 2966233522

--------------------------------------------------------------------
-------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
```

Practice 4-1: Using Different Access Paths (continued)

```
---------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT   |      |     1 |    13 |     7   (0)|
00:00:01 |
|   1 |  SORT AGGREGATE    |      |     1 |    13 |            |
|
|*  2 |   TABLE ACCESS FULL| T    |     1 |    13 |     7   (0)|
00:00:01 |
---------------------------------------------------------------------
-------

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

   2 - filter("D"=1)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          5  recursive calls
          0  db block gets
         48  consistent gets
          0  physical reads
          0  redo size
        418  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

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

59) How would you improve the performance of a query, such as the one in the previous step? Implement your solution.

a) Make sure you gather correctly the statistics for your table so that the index skip scan can be used.

```
SQL> @iss_gather_stats
SQL> set echo on
SQL>
SQL> execute dbms_stats.gather_table_stats('SH','T',cascade=>TRUE);

PL/SQL procedure successfully completed.

SQL>
```

60) Execute the following query:
`select count(*) from t where d=1;`

Practice 4-1: Using Different Access Paths (continued)

What do you observe?

a) The optimizer now uses the index to perform an index skip scan.

```
SQL> @query20
SQL> set echo on
SQL>
SQL> set linesize 200
SQL>
SQL> set timing on
SQL> set autotrace on
SQL>
SQL> select count(*) from t where d=1;

  COUNT(*)
----------
         1

Elapsed: 00:00:00.00

Execution Plan
----------------------------------------------------------
Plan hash value: 2609927160

--------------------------------------------------------------------
-----
| Id  | Operation        | Name | Rows  | Bytes | Cost (%CPU)| Time
|
--------------------------------------------------------------------
-----
|   0 | SELECT STATEMENT |      |     1 |     4 |     2   (0)|
00:00:01 |
|   1 |  SORT AGGREGATE  |      |     1 |     4 |            |
|
|*  2 |   INDEX SKIP SCAN| IT   |     1 |     4 |     2   (0)|
00:00:01 |
--------------------------------------------------------------------
-----

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

   2 - access("D"=1)
       filter("D"=1)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
         23  consistent gets
          0  physical reads
          0  redo size
        418  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
```

```
          0  sorts (disk)
          1  rows processed

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

61) Compare the result of executing the previous query with the result you obtain when you execute the following query:
`select /*+ INDEX_FFS(t it) */ count(*) from t where d=1;`

What do you observe?

a) The optimizer uses a fast full index scan, but this is not better than the index skip scan.

```
SQL> @query21
SQL> set echo on
SQL>
SQL> set linesize 200
SQL>
SQL> set timing on
SQL> set autotrace on
SQL>
SQL> select /*+ INDEX_FFS(t it) */ count(*) from t where d=1;

  COUNT(*)
----------
         1

Elapsed: 00:00:00.00

Execution Plan
----------------------------------------------------------
Plan hash value: 273610729

--------------------------------------------------------------------
----------
| Id  | Operation              | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
--------------------------------------------------------------------
----------
|   0 | SELECT STATEMENT       |      |     1 |     4 |     9   (0)|
00:00:01 |
|   1 |  SORT AGGREGATE        |      |     1 |     4 |            |
|
|*  2 |   INDEX FAST FULL SCAN| IT   |     1 |     4 |     9   (0)|
00:00:01 |
--------------------------------------------------------------------
----------

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

   2 - filter("D"=1)
```

```
Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
         31  consistent gets
          0  physical reads
          0  redo size
        418  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

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

62) Execute the `iss_cleanup.sql` script to clean up your environment for this case.

```
SQL> @iss_cleanup
SQL> set echo on
SQL>
SQL> drop table t purge;

Table dropped.

SQL>
SQL>
```

63) Exit from your SQL*Plus session, and execute the `ap_cleanup.sh` script to clean up your environment for this lab.

```
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Access_Paths]$ ./ap_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Apr 9 15:22:25 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> revoke dba from sh;

Revoke succeeded.
```

Practice 4-1: Using Different Access Paths (continued)

```
SQL>
SQL> @sh_main sh example temp oracle
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/
/home/oracle/ v3
SQL> Rem
SQL> Rem $Header: sh_main.sql 06-mar-2008.15:00:45 cbauwens Exp $
SQL> Rem

....

<<<<< CHANNELS DIMENSION >>>>>
Dimension - display name, description and plural name
Level - display name and description
Hierarchy - display name and description
        - default calculation hierarchy
        - default display hierarchy
Level Attributes - name, display name, description
Drop dimension attributes prior to re-creation
No attribute to drop
Create dimension attributes and add their level attributes
        - Long Description created
        - Short Description created
Classify entity descriptor use
        - Long Description
        - Short Description
-
<<<<< FINAL PROCESSING >>>>>
        - Changes have been committed

PL/SQL procedure successfully completed.


Commit complete.


gathering statistics ...

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Access_Paths]$
```

Practice 4-2: Using the Result Cache

In this practice, you explore the various possibilities of caching query results in the System Global Area (SGA). Perform the following steps to understand the use of Query Result Cache.

1) Change to the `$HOME/solutions/Query_Result_Cache` directory and execute the `result_cache_setup.sh` script.

```
[oracle@edrsr33p1-orcl ~]$ cd $HOME/solutions/Query_Result_Cache
[oracle@edrsr33p1-orcl Query_Result_Cache]$ ./result_cache_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 25 14:46:14
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
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.
```

Practice 4-2: Using the Result Cache (continued)

```
SQL>
4 rows created.

SQL>
8 rows created.

SQL>
16 rows created.

SQL>
32 rows created.

SQL>
64 rows created.

SQL>
128 rows created.

SQL>
256 rows created.

SQL>
512 rows created.

SQL>
1024 rows created.

SQL>
2048 rows created.

SQL>
4096 rows created.

SQL>
8192 rows created.

SQL>
16384 rows created.

SQL>
32768 rows created.

SQL>
65536 rows created.

SQL>
131072 rows created.

SQL>
262144 rows created.

SQL>
524288 rows created.

SQL>
1048576 rows created.
```

Practice 4-2: Using the Result Cache (continued)

```
SQL> SQL>
1 row created.

SQL> SQL>
Commit complete.

SQL> SQL>
System altered.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Query_Result_Cache]$


--------------------------------------------------------------


#!/bin/bash

cd /home/oracle/solutions/Query_Result_Cache

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin

sqlplus / as sysdba <<FIN!

set echo on

drop user qrc cascade;

create user qrc identified by qrc
default tablespace users
temporary tablespace temp;

grant connect, resource, dba to qrc;

connect qrc/qrc

exec dbms_result_cache.flush;

drop table cachejfv purge;

create table cachejfv(c varchar2(500)) tablespace users;

insert into cachejfv
values('aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa');
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
```

```
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;
insert into cachejfv select * from cachejfv;

insert into cachejfv values('b');

commit;

alter system flush buffer_cache;

FIN!
```

2) Open a terminal window, invoke SQL*Plus, and connect as the `qrc` user. From now on, do not disconnect from this session. Determine the current content of the query cache using the following statement:
`select type,status,name,object_no,row_count,row_size_avg from v$result_cache_objects order by 1;`
What do you observe?

   a) Use the `check_result_cache.sql` script. Right now, the query cache should be empty.

```
[oracle@edrsr33p1-orcl Query_Result_Cache]$ sqlplus qrc/qrc

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 25 14:49:02
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> @check_result_cache
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;
```

Practice 4-2: Using the Result Cache (continued)

```
no rows selected

SQL>
SQL>


--------------------------------------------------------------

set echo on

select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;
```

3) Set timing on and execute the query as follows. You can use the `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,cachejfv c6, cachejfv c7
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b' and c6.c='b' and c7.c='b';
```

```
SQL> set timing on
SQL> @query1

  COUNT(*)
----------
         1

Elapsed: 00:00:02.46
SQL>

--------------------------------------------------------------

select /*+ result_cache q_name(Q1) */ count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b'
and c6.c='b' and c7.c='b';
```

4) Determine the execution plan of the previous query by using the `explain_query1.sql` script. What do you observe?

   a) Because of the `result_cache` hint, the result of the query is computed using the result cache.

```
SQL> @explain_query1
SQL> set echo on
SQL>
SQL> explain plan for
  2  select /*+ result_cache q_name(Q1) */ count(*)
  3  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
  4  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b' and c6.c='b' and c7.c='b';
```

Practice 4-2: Using the Result Cache (continued)

```
Explained.

Elapsed: 00:00:00.05
SQL>
SQL> set linesize 180
SQL> set pagesize 200
SQL>
SQL> select plan_table_output from
table(dbms_xplan.display('plan_table',null,'ALL'));

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------
---------------------------------------------------------------------
--------------------------------------------
Plan hash value: 2531260445

---------------------------------------------------------------------
--------------------------------------
| Id  | Operation                    | Name                        |
Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------
--------------------------------------
|   0 | SELECT STATEMENT             |                             |
1 |  1764 |  7316P  (1)|999:59:59 |
|   1 |  RESULT CACHE                | b7rh5vw33py4ug4n8xaav6525g |
|       |            |          |
|   2 |   SORT AGGREGATE             |                             |
1 |  1764 |            |          |
|   3 |    MERGE JOIN CARTESIAN      |                             |
294P|    15E|  7316P  (1)|999:59:59 |
|   4 |     MERGE JOIN CARTESIAN     |                             |
941T|  1264P|    23P  (1)|999:59:59 |
|   5 |      MERGE JOIN CARTESIAN    |                             |
3007G|  3446T|    74T  (1)|999:59:59 |
|   6 |       MERGE JOIN CARTESIAN   |                             |
9606M|  9018G|   238G  (1)|999:59:59 |
|   7 |        MERGE JOIN CARTESIAN |                             |
30M|    21G|   761M  (1)|999:59:59 |
|   8 |         MERGE JOIN CARTESIAN|                             |
98011 |    47M|  2432K  (1)| 08:06:26 |
|*  9 |          TABLE ACCESS FULL  | CACHEJFV                    |
313 | 78876 |  7748   (1)| 00:01:33 |
|  10 |          BUFFER SORT        |                             |
313 | 78876 |  2424K  (1)| 08:04:53 |
|* 11 |           TABLE ACCESS FULL | CACHEJFV                    |
313 | 78876 |  7746   (1)| 00:01:33 |
|  12 |         BUFFER SORT         |                             |
313 | 78876 |   761M  (1)|999:59:59 |
|* 13 |          TABLE ACCESS FULL  | CACHEJFV                    |
313 | 78876 |  7746   (1)| 00:01:33 |
|  14 |        BUFFER SORT          |                             |
313 | 78876 |   238G  (1)|999:59:59 |
|* 15 |         TABLE ACCESS FULL   | CACHEJFV                    |
313 | 78876 |  7746   (1)| 00:01:33 |
|  16 |       BUFFER SORT           |                             |
313 | 78876 |    74T  (1)|999:59:59 |
```

```
|* 17 |        TABLE ACCESS FULL     | CACHEJFV                     |
313 | 78876 |  7746   (1)| 00:01:33 |
|  18 |       BUFFER SORT            |                              |
313 | 78876 |    23P  (1)|999:59:59 |
|* 19 |        TABLE ACCESS FULL     | CACHEJFV                     |
313 | 78876 |  7746   (1)| 00:01:33 |
|  20 |      BUFFER SORT             |                              |
313 | 78876 |  7316P  (1)|999:59:59 |
|* 21 |       TABLE ACCESS FULL      | CACHEJFV                     |
313 | 78876 |  7746   (1)| 00:01:33 |
--------------------------------------------------------------------
--------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$1
   9 - SEL$1 / C7@SEL$1
  11 - SEL$1 / C6@SEL$1
  13 - SEL$1 / C5@SEL$1
  15 - SEL$1 / C4@SEL$1
  17 - SEL$1 / C3@SEL$1
  19 - SEL$1 / C2@SEL$1
  21 - SEL$1 / C1@SEL$1

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

   9 - filter("C7"."C"='b')
  11 - filter("C6"."C"='b')
  13 - filter("C5"."C"='b')
  15 - filter("C4"."C"='b')
  17 - filter("C3"."C"='b')
  19 - filter("C2"."C"='b')
  21 - filter("C1"."C"='b')

Column Projection Information (identified by operation id):
-----------------------------------------------------------

   1 - COUNT(*)[22]
   2 - (#keys=0) COUNT(*)[22]
   3 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500],
"C5"."C"[VARCHAR2,500],
       "C4"."C"[VARCHAR2,500], "C3"."C"[VARCHAR2,500],
"C2"."C"[VARCHAR2,500], "C1"."C"[VARCHAR2,500]
   4 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500],
"C5"."C"[VARCHAR2,500],
       "C4"."C"[VARCHAR2,500], "C3"."C"[VARCHAR2,500],
"C2"."C"[VARCHAR2,500]
   5 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500],
"C5"."C"[VARCHAR2,500],
       "C4"."C"[VARCHAR2,500], "C3"."C"[VARCHAR2,500]
   6 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500],
"C5"."C"[VARCHAR2,500],
       "C4"."C"[VARCHAR2,500]
   7 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500],
"C5"."C"[VARCHAR2,500]
```

```
   8 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500]
   9 - "C7"."C"[VARCHAR2,500]
  10 - (#keys=0) "C6"."C"[VARCHAR2,500]
  11 - "C6"."C"[VARCHAR2,500]
  12 - (#keys=0) "C5"."C"[VARCHAR2,500]
  13 - "C5"."C"[VARCHAR2,500]
  14 - (#keys=0) "C4"."C"[VARCHAR2,500]
  15 - "C4"."C"[VARCHAR2,500]
  16 - (#keys=0) "C3"."C"[VARCHAR2,500]
  17 - "C3"."C"[VARCHAR2,500]
  18 - (#keys=0) "C2"."C"[VARCHAR2,500]
  19 - "C2"."C"[VARCHAR2,500]
  20 - (#keys=0) "C1"."C"[VARCHAR2,500]
  21 - "C1"."C"[VARCHAR2,500]

Result Cache Information (identified by operation id):
------------------------------------------------------

   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,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cac"


Note
-----
   - dynamic sampling used for this statement

89 rows selected.

Elapsed: 00:00:00.03
SQL>

--------------------------------------------------------------

set echo on

explain plan for
select /*+ result_cache q_name(Q1) */ count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b'
and c6.c='b' and c7.c='b';

set linesize 180
set pagesize 200

select plan_table_output from
table(dbms_xplan.display('plan_table',null,'ALL'));
```

5) Determine the current content of the query cache by using the `check_result_cache.sql` script. What do you observe?

   a) You can now see the result of your query cached.

Practice 4-2: Using the Result Cache (continued)

```
SQL> @check_result_cache
SQL> set echo on
SQL>
SQL> set long 2000
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ------------------------------------------
------------------------------------------------------------------
------------- ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
71474   0
           0

Result     Published select /*+ result_cache q_name(Q1) */ count(*)
0   1
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5,cachejfv c6, cac
           5


Elapsed: 00:00:00.00
SQL>

Elapsed: 00:00:00.00
SQL>

--------------------------------------------------------------

set echo on

set long 2000

select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;
```

6) Flush the buffer cache of your instance and rerun the query executed in step 3. What do you observe?

   a) The execution time for the query is now almost instantaneous.

```
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.05
SQL> @query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
```

Practice 4-2: Using the Result Cache (continued)

```
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b' and c6.c='b' and c7.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:00.00
SQL>

--------------------------------------------------------------

select /*+ result_cache q_name(Q1) */ count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b'
and c6.c='b' and c7.c='b';
```

7) Insert a new row into the `CACHEJFV` table using the following statement:
   `insert into cachejfv values('c');`
   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> @check_result_cache
SQL> set echo on
SQL>
SQL> set long 2000
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ---------------------------------------------
--------------------------------------------------------------------
------------- ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
71474   0
           0

Result     Invalid   select /*+ result_cache q_name(Q1) */ count(*)
0   1
```

```
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5,cachejfv c6, cac
           5


Elapsed: 00:00:00.00
SQL>

--------------------------------------------------------------

set echo on

set long 2000

select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;
```

8) Execute your query from step 3 and step 6 again and check the result cache. What do you observe?

   a) Again, it takes some time to execute the query. The result cache shows that a new entry has been added for the new result.

```
SQL> @query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b' and c6.c='b' and c7.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:02.33
SQL>
SQL> @check_result_cache
SQL> set echo on
SQL>
SQL> set long 2000
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ---------------------------------------------
--------------------------------------------------------------------
------------- ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
71474   0
           0
```

Practice 4-2: Using the Result Cache (continued)

```
Result     Invalid   select /*+ result_cache q_name(Q1) */ count(*)
0   1
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5,cachejfv c6, cac
           5

Result     Published select /*+ result_cache q_name(Q1) */ count(*)
0   1
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5,cachejfv c6, cac
           5


Elapsed: 00:00:00.00
SQL>

--------------------------------------------------------------

select /*+ result_cache q_name(Q1) */ count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b'
and c6.c='b' and c7.c='b';



set echo on

set long 2000

select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;
```

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  = 2080K bytes (2080 blocks)
Maximum Result Size = 104K bytes (104 blocks)
[Memory]
Total Memory = 116008 bytes [0.035% of the Shared Pool]
... Fixed Memory = 5132 bytes [0.002% of the Shared Pool]
....... Cache Mgr  = 108 bytes
....... Memory Mgr = 124 bytes
....... Bloom Fltr = 2K bytes
....... State Objs = 2852 bytes
... Dynamic Memory = 110876 bytes [0.033% of the Shared Pool]
....... Overhead = 78108 bytes
........... Hash Table    = 32K bytes (4K buckets)
........... Chunk Ptrs    = 12K bytes (3K slots)
........... Chunk Maps    = 12K bytes
........... Miscellaneous = 20764 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.

Elapsed: 00:00:00.12
SQL>
```

10) Execute your query from step 3 and step 6 again. What do you observe?

a) The query again uses the result that was previously cached.

```
SQL> @query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b' and c6.c='b' and c7.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:00.01
SQL>

--------------------------------------------------------------

select /*+ result_cache q_name(Q1) */ count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b'
and c6.c='b' and c7.c='b';
```

11) Ensure that you bypass the result cache before performing the next step.

```
SQL>
SQL> exec DBMS_RESULT_CACHE.BYPASS(bypass_mode=>true);

PL/SQL procedure successfully completed.

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

12) Execute your query again. What do you observe?

a) The query again takes longer to execute because it no longer uses the result cache.

```
SQL> @query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b' and c6.c='b' and c7.c='b';
```

Practice 4-2: Using the Result Cache (continued)

```
  COUNT(*)
----------
         1

Elapsed: 00:00:02.34
SQL>

--------------------------------------------------------------

select /*+ result_cache q_name(Q1) */ count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b'
and c6.c='b' and c7.c='b';
```

13) Ensure that you no longer bypass the result cache and check that your query uses it again.

```
SQL> exec DBMS_RESULT_CACHE.BYPASS(bypass_mode=>false);

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.01
SQL> @query1
SQL> select /*+ result_cache q_name(Q1) */ count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b' and c6.c='b' and c7.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:00.00
SQL>
SQL>

--------------------------------------------------------------

select /*+ result_cache q_name(Q1) */ count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b'
and c6.c='b' and c7.c='b';
```

14) Execute the following query by using the `query2.sql` script:
`select count(*) from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv c5,cachejfv c6, cachejfv c7 where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b' and c6.c='b' and c7.c='b';`
What do you observe?

Practice 4-2: Using the Result Cache (continued)

a) Although the query is the same as the one used in step 3, it is not recognized as cached because it does not contain the hint. So its execution time is long again.

```
SQL> @query2
SQL> select count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b' and c6.c='b' and c7.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:02.41
SQL>

--------------------------------------------------------------

select count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b'
and c6.c='b' and c7.c='b';
```

15) How would you force the previous query to use the cached result without using hints? Use the `force_query2.sql` script and then verify that you successfully used the cached result. Finally, undo your change.

```
SQL> @force_query2
SQL> set echo on
SQL>
SQL> show parameter result_cache_mode

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------------------
-----------
result_cache_mode                    string      MANUAL
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- --------------------------------------------------
--------------------------------------------------------------------
------------ --------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
71474    0
           0

Result     Invalid   select /*+ result_cache q_name(Q1) */ count(*)
0   1
```

Practice 4-2: Using the Result Cache (continued)

```
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5,cachejfv c6, cac
           5

Result     Published select /*+ result_cache q_name(Q1) */ count(*)
0   1
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5,cachejfv c6, cac
           5


Elapsed: 00:00:00.00
SQL>
SQL> alter session set result_cache_mode=force;

Session altered.

Elapsed: 00:00:00.04
SQL>
SQL> explain plan for
  2  select count(*)
  3  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
  4  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b' and c6.c='b' and c7.c='b';

Explained.

Elapsed: 00:00:00.05
SQL>
SQL> set linesize 180
SQL> set pagesize 200
SQL>
SQL> select plan_table_output from
table(dbms_xplan.display('plan_table',null,'ALL'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
--------------------------------------------
Plan hash value: 2531260445

--------------------------------------------------------------------
--------------------------------------
| Id  | Operation                   | Name                       |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
--------------------------------------
|   0 | SELECT STATEMENT            |                            |
1 |  1764 |  7316P  (1)|999:59:59 |
|   1 |  RESULT CACHE               | b7rh5vw33py4ug4n8xaav6525g |
|       |            |          |
|   2 |   SORT AGGREGATE            |                            |
1 |  1764 |            |          |
|   3 |    MERGE JOIN CARTESIAN     |                            |
294P|    15E|  7316P  (1)|999:59:59 |
```

```
|   4 |     MERGE JOIN CARTESIAN    |                             |
941T|  1264P|    23P  (1)|999:59:59 |
|   5 |      MERGE JOIN CARTESIAN   |                             |
3007G|  3446T|    74T  (1)|999:59:59 |
|   6 |       MERGE JOIN CARTESIAN  |                             |
9606M|  9018G|   238G  (1)|999:59:59 |
|   7 |        MERGE JOIN CARTESIAN |                             |
30M|    21G|   761M  (1)|999:59:59 |
|   8 |         MERGE JOIN CARTESIAN|                             |
98011 |    47M|  2432K  (1)| 08:06:26 |
|*  9 |          TABLE ACCESS FULL  | CACHEJFV                    |
313 | 78876 |  7748   (1)| 00:01:33 |
|  10 |          BUFFER SORT        |                             |
313 | 78876 |  2424K  (1)| 08:04:53 |
|* 11 |           TABLE ACCESS FULL | CACHEJFV                    |
313 | 78876 |  7746   (1)| 00:01:33 |
|  12 |         BUFFER SORT         |                             |
313 | 78876 |   761M  (1)|999:59:59 |
|* 13 |          TABLE ACCESS FULL  | CACHEJFV                    |
313 | 78876 |  7746   (1)| 00:01:33 |
|  14 |        BUFFER SORT          |                             |
313 | 78876 |   238G  (1)|999:59:59 |
|* 15 |         TABLE ACCESS FULL   | CACHEJFV                    |
313 | 78876 |  7746   (1)| 00:01:33 |
|  16 |       BUFFER SORT           |                             |
313 | 78876 |    74T  (1)|999:59:59 |
|* 17 |        TABLE ACCESS FULL    | CACHEJFV                    |
313 | 78876 |  7746   (1)| 00:01:33 |
|  18 |      BUFFER SORT            |                             |
313 | 78876 |    23P  (1)|999:59:59 |
|* 19 |       TABLE ACCESS FULL     | CACHEJFV                    |
313 | 78876 |  7746   (1)| 00:01:33 |
|  20 |     BUFFER SORT             |                             |
313 | 78876 |  7316P  (1)|999:59:59 |
|* 21 |      TABLE ACCESS FULL      | CACHEJFV                    |
313 | 78876 |  7746   (1)| 00:01:33 |
--------------------------------------------------------------------
--------------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$1
   9 - SEL$1 / C7@SEL$1
  11 - SEL$1 / C6@SEL$1
  13 - SEL$1 / C5@SEL$1
  15 - SEL$1 / C4@SEL$1
  17 - SEL$1 / C3@SEL$1
  19 - SEL$1 / C2@SEL$1
  21 - SEL$1 / C1@SEL$1

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

   9 - filter("C7"."C"='b')
  11 - filter("C6"."C"='b')
  13 - filter("C5"."C"='b')
```

```
  15 - filter("C4"."C"='b')
  17 - filter("C3"."C"='b')
  19 - filter("C2"."C"='b')
  21 - filter("C1"."C"='b')

Column Projection Information (identified by operation id):
-----------------------------------------------------------

   1 - COUNT(*)[22]
   2 - (#keys=0) COUNT(*)[22]
   3 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500],
"C5"."C"[VARCHAR2,500],
       "C4"."C"[VARCHAR2,500], "C3"."C"[VARCHAR2,500],
"C2"."C"[VARCHAR2,500], "C1"."C"[VARCHAR2,500]
   4 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500],
"C5"."C"[VARCHAR2,500],
       "C4"."C"[VARCHAR2,500], "C3"."C"[VARCHAR2,500],
"C2"."C"[VARCHAR2,500]
   5 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500],
"C5"."C"[VARCHAR2,500],
       "C4"."C"[VARCHAR2,500], "C3"."C"[VARCHAR2,500]
   6 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500],
"C5"."C"[VARCHAR2,500],
       "C4"."C"[VARCHAR2,500]
   7 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500],
"C5"."C"[VARCHAR2,500]
   8 - (#keys=0) "C7"."C"[VARCHAR2,500], "C6"."C"[VARCHAR2,500]
   9 - "C7"."C"[VARCHAR2,500]
  10 - (#keys=0) "C6"."C"[VARCHAR2,500]
  11 - "C6"."C"[VARCHAR2,500]
  12 - (#keys=0) "C5"."C"[VARCHAR2,500]
  13 - "C5"."C"[VARCHAR2,500]
  14 - (#keys=0) "C4"."C"[VARCHAR2,500]
  15 - "C4"."C"[VARCHAR2,500]
  16 - (#keys=0) "C3"."C"[VARCHAR2,500]
  17 - "C3"."C"[VARCHAR2,500]
  18 - (#keys=0) "C2"."C"[VARCHAR2,500]
  19 - "C2"."C"[VARCHAR2,500]
  20 - (#keys=0) "C1"."C"[VARCHAR2,500]
  21 - "C1"."C"[VARCHAR2,500]

Result Cache Information (identified by operation id):
------------------------------------------------------

   1 - column-count=1; dependencies=(QRC.CACHEJFV);
attributes=(single-row); parameters=(nls); name="select count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
where c1.c='b' and c2."


Note
-----
   - dynamic sampling used for this statement

89 rows selected.
```

```
Elapsed: 00:00:00.07
SQL>
SQL> select count(*)
  2  from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
  3  where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and
c5.c='b' and c6.c='b' and c7.c='b';

  COUNT(*)
----------
         1

Elapsed: 00:00:00.04
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ------------------------------------------
------------------------------------------------------------------
------------- ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV
71474   0
           0

Result     Invalid   select /*+ result_cache q_name(Q1) */ count(*)
0   1
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5,cachejfv c6, cac
           5

Result     Published select /*+ result_cache q_name(Q1) */ count(*)
0   1
                     from cachejfv c1,cachejfv c2,cachejfv
c3,cachejfv c4,cachejfv c5,cachejfv c6, cac
           5


Elapsed: 00:00:00.00
SQL>
SQL> alter session set result_cache_mode=manual;

Session altered.

Elapsed: 00:00:00.00
SQL>
SQL>

--------------------------------------------------------------

set echo on

show parameter result_cache_mode
```

```
select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;

alter session set result_cache_mode=force;

explain plan for
select count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b'
and c6.c='b' and c7.c='b';

set linesize 180
set pagesize 200

select plan_table_output from
table(dbms_xplan.display('plan_table',null,'ALL'));

select count(*)
from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
where c1.c='b' and c2.c='b' and c3.c='b' and c4.c='b' and c5.c='b'
and c6.c='b' and c7.c='b';

select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;

alter session set result_cache_mode=manual;
```

16) 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: 00:00:00.01
SQL> select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;

no rows selected

Elapsed: 00:00:00.00
SQL>
```

17) Create a PL/SQL function that uses the result cache by running the `cre_func.sql` script.

```
SQL> @cre_func
SQL> create or replace function CACHEJFV_COUNT(v varchar2)
     return number
     result_cache relies_on (cachejfv)
     is
      cnt number;
     begin
      select count(*) into cnt
```

Practice 4-2: Using the Result Cache (continued)

```
  8   from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
  9   where c1.c=v and c2.c=v and c3.c=v and c4.c=v and c5.c=v and
c6.c=v and c7.c=v;
 10   return cnt;
 11  end;
 12  /

Function created.

Elapsed: 00:00:00.15
SQL>

--------------------------------------------------------------

create or replace function CACHEJFV_COUNT(v varchar2)
return number
result_cache relies_on (cachejfv)
is
 cnt number;
begin
 select count(*) into cnt
 from cachejfv c1,cachejfv c2,cachejfv c3,cachejfv c4,cachejfv
c5,cachejfv c6, cachejfv c7
 where c1.c=v and c2.c=v and c3.c=v and c4.c=v and c5.c=v and c6.c=v
and c7.c=v;
 return cnt;
end;
/
```

18) Determine what is in the result cache by querying `V$RESULT_CACHE_OBJECTS`.

```
SQL> select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;

no rows selected

Elapsed: 00:00:00.00
SQL>
```

19) Call the new function with `'b'` as its argument. What do you observe?

   a) It takes a long time to execute because the result is not cached yet. After executing the function, the function's result for the `'b'` argument is cached.

```
SQL> select cachejfv_count('b') from dual;

CACHEJFV_COUNT('B')
-------------------
                  1

Elapsed: 00:00:01.56
SQL>
SQL> select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;
```

Practice 4-2: Using the Result Cache (continued)

```
TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ------------------------------------------
------------------------------------------------------------------
------------ ---------- ----------
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV_COUNT
71475   0
           0

Dependency Published QRC.CACHEJFV
71474   0
           0

Result     Published
"QRC"."CACHEJFV_COUNT"::8."CACHEJFV_COUNT"#8440831613f0f5d3 #1
0   1
           4


Elapsed: 00:00:00.00
SQL>
SQL>
```

20) Call the new function with `'b'` as its argument again. What do you observe?

a) This time the function executes almost instantaneously.

```
SQL> select cachejfv_count('b') from dual;

CACHEJFV_COUNT('B')
-------------------
                  1

Elapsed: 00:00:00.00
SQL>
```

21) Call the new function with `'c'` as its argument again. What do you observe?

a) Again it takes a long time to execute the function because of the new value for the argument. After execution, the second result is cached.

```
SQL> select cachejfv_count('c') from dual;

CACHEJFV_COUNT('C')
-------------------
                  1

Elapsed: 00:00:04.15
SQL> select type,status,name,object_no,row_count,row_size_avg from
v$result_cache_objects order by 1;

TYPE       STATUS    NAME
OBJECT_NO  ROW_COUNT
---------- --------- ------------------------------------------
------------------------------------------------------------------
------------ ---------- ----------
```

Practice 4-2: Using the Result Cache (continued)

```
ROW_SIZE_AVG
------------
Dependency Published QRC.CACHEJFV_COUNT
71475   0
           0

Dependency Published QRC.CACHEJFV
71474   0
           0

Result     Published
"QRC"."CACHEJFV_COUNT"::8."CACHEJFV_COUNT"#8440831613f0f5d3 #1
0   1
           4

Result     Published
"QRC"."CACHEJFV_COUNT"::8."CACHEJFV_COUNT"#8440831613f0f5d3 #1
0   1
           4


Elapsed: 00:00:00.01
SQL>
SQL>
SQL> select cachejfv_count('c') from dual;

CACHEJFV_COUNT('C')
-------------------
                  1

Elapsed: 00:00:00.00
SQL>
```

Practices for Lesson 5

Practice 5-1: Extracting Execution Plans

In this practice, you use various methods to extract the execution plan used by the optimizer to execute a query. Note that all scripts needed for this lab can be found in your `$HOME/solutions/Explain_Plan` directory.

1) Connected as the `oracle` user from a terminal session, execute the `ep_setup.sh` script. This script creates a new user called `EP` and a table called `TEST` used throughout this lab.

```
[oracle@edrsr33p1-orcl Explain_Plan]$ ./ep_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Apr 2 20:11:45 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user ep cascade;
drop user ep cascade
          *
ERROR at line 1:
ORA-01918: user 'EP' does not exist


SQL>
SQL> create user ep identified by ep default tablespace users
temporary tablespace temp;

User created.

SQL>
SQL> grant connect, resource, dba to ep;

Grant succeeded.

SQL>
SQL> connect ep/ep
Connected.
SQL>
SQL> drop table test purge;
drop table test purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
SQL> create table test(c number, d varchar2(500));

Table created.
```

Practice 5-1: Extracting Execution Plans (continued)

```
SQL>
SQL> begin
  2  for i in 1..20000 loop
  3  insert into test
values(1,'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa');
  4  end loop;
  5  commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL> create index test_c_indx on test(c);

Index created.

SQL>
SQL> exec dbms_stats.gather_schema_stats('EP');

PL/SQL procedure successfully completed.

SQL>
SQL> alter system flush shared_pool;

System altered.

SQL>
SQL> alter system flush buffer_cache;

System altered.

SQL>
SQL> set echo off
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Explain_Plan]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Explain_Plan

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin
```

Practice 5-1: Extracting Execution Plans (continued)

```
sqlplus / as sysdba @ep_setup.sql

--------------------------------------------------------------

set echo on

drop user ep cascade;

create user ep identified by ep default tablespace users temporary
tablespace temp;

grant connect, resource, dba to ep;

connect ep/ep

drop table test purge;

create table test(c number, d varchar2(500));

begin
for i in 1..20000 loop
insert into test
values(1,'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa');
end loop;
commit;
end;
/

create index test_c_indx on test(c);

exec dbms_stats.gather_schema_stats('EP');

alter system flush shared_pool;

alter system flush buffer_cache;

set echo off
set term off

select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1;

exit;
```

2) From the same terminal session (referred to as session 1 in the rest of this lab), be ready to execute the `ep_session_issue.sh` script. Enter the command, but do not execute it yet.

```
Session 1:
----------

[oracle@edrsr33p1-orcl Explain_Plan]$ ./ep_session_issue.sh

--------------------------------------------------------------
```

Practice 5-1: Extracting Execution Plans (continued)

```
#!/bin/bash

cd /home/oracle/solutions/Explain_Plan

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus ep/ep @ep_session_issue.sql

--------------------------------------------------------------

set echo off
set termout off

alter session set optimizer_mode=rule;

set termout on
set echo on

set timing on

select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1;

exit;
```

3) From a second terminal session (referred to as session 2 in the rest of this lab), connect as the `oracle` user. After this, connect to a SQL*Plus session as the `SYS` user. From that SQL*Plus session, be ready to use SQL Monitoring to monitor the execution plan used by session 1. You can execute the `ep_monitoring.sql` script for that purpose. Enter the command, but do not execute it yet. **Note:** Ensure that you understand the coordination between both sessions by pre-reading steps 4 and 5 before you continue.

```
Session 2:
----------

[oracle@edrsr33p1-orcl Explain_Plan]$ sqlplus / as sysdba

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Apr 2 20:12:28 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> @ep_monitor
```

Practice 5-1: Extracting Execution Plans (continued)

```
--------------------------------------------------------------

set echo on
set long 10000000
set longchunksize 10000000
set linesize 200
set pagesize 1000

exec dbms_lock.sleep(8);

select
dbms_sqltune.report_sql_monitor(sql_id=>'dkz7v96ym42c6',report_level
=>'ALL') from dual;
```

4) After you are ready in both the sessions, press [Enter] in session 1 to start the execution of the `ep_session_issue.sh` script. **Note:** Do not wait. Proceed with the next step immediately.

```
Session 1:
----------

[oracle@edrsr33p1-orcl Explain_Plan]$ ./ep_session_issue.sh
SQL*Plus: Release 11.1.0.6.0 - Production on Wed Apr 2 20:12:47 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> set timing on
SQL>
SQL> select count(*) from test t1, test t2 where t1.c=t2.c and
t1.c=1;
```

5) In session 2, enter return to start the execution of the `ep_monitor.sql` script. After the execution, enter "`/`" and go back to your SQL*Plus session as many times as necessary until session 1 is done with its execution. What do you observe?

   a) You can see that session 1 uses `NESTED LOOPS` on top of two `INDEX RANGE SCANS` to execute the query. It takes approximately 47 seconds to execute session 1's query. The time depends on your environment. The big advantage of SQL Monitoring is that you can clearly see which steps in the execution plan take most of the resources. In this case, you clearly see that you do only one scan of the index, and that for each row returned, you execute another index scan to probe. This is not really efficient. Also there is no costs information for this monitored plan.

Practice 5-1: Extracting Execution Plans (continued)

```
Session 2:
----------

SQL> @ep_monitor
SQL> set long 10000000
SQL> set longchunksize 10000000
SQL> set linesize 200
SQL> set pagesize 1000
SQL>
SQL> exec dbms_lock.sleep(8);

PL/SQL procedure successfully completed.

SQL>
SQL> select
dbms_sqltune.report_sql_monitor(sql_id=>'dkz7v96ym42c6',report_level
=>'ALL') from dual;

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL Monitoring Report

SQL Text
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------

Global Information
 Status              :  EXECUTING
 Instance ID         :  1
 Session ID          :  138
 SQL ID              :  dkz7v96ym42c6
 SQL Execution ID    :  16777222
 Plan Hash Value     :  1643938535
 Execution Started   :  04/02/2008 20:12:46
 First Refresh Time  :  04/02/2008 20:12:54
 Last Refresh Time   :  04/02/2008 20:12:56

-----------------------------------------
| Elapsed |   Cpu   |  Other   | Buffer |
| Time(s) | Time(s) | Waits(s) |  Gets  |
-----------------------------------------
|    8.10 |    8.09 |     0.01 |   134K |
-----------------------------------------



SQL Plan Monitoring Details
====================================================================
=================================================================
```

Practice 5-1: Extracting Execution Plans (continued)

```
| Id   |      Operation       |     Name      |  Rows    | Cost |
Time    | Start  | Starts |   Rows    | Activity  | Activity Detail |
|      |                      |               | (Estim)  |      |
Active(s) | Active |        | (Actual) | (percent) |   (sample #)
|
====================================================================
==================================================================
|    0 | SELECT STATEMENT     |               |          |      |
|       |       1 |         |           |                |
|    1 |   SORT AGGREGATE     |               |          |      |
1 |     +6 |       1 |        0 |     12.50 | Cpu (1)          |
| -> 2 |    NESTED LOOPS      |               |          |      |
3 |     +8 |       1 |   79096K |           |                  |
| -> 3 |     INDEX RANGE SCAN | TEST_C_INDX   |          |      |
3 |     +8 |       1 |     3954 |           |                  |
| -> 4 |     INDEX RANGE SCAN | TEST_C_INDX   |          |      |
9 |     +2 |    3955 |   79096K |     87.50 | Cpu (7)          |
====================================================================
==================================================================


SQL>
SQL> /

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL Monitoring Report

SQL Text
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------


Global Information
 Status              :  EXECUTING
 Instance ID         :  1
 Session ID          :  138
 SQL ID              :  dkz7v96ym42c6
 SQL Execution ID    :  16777222
 Plan Hash Value     :  1643938535
 Execution Started   :  04/02/2008 20:12:46
 First Refresh Time  :  04/02/2008 20:12:54
 Last Refresh Time   :  04/02/2008 20:12:58

-----------------------------------------
| Elapsed |   Cpu   |  Other   | Buffer |
| Time(s) | Time(s) | Waits(s) |  Gets  |
-----------------------------------------
|      10 |      10 |     0.01 |   167K |
-----------------------------------------
```

Practice 5-1: Extracting Execution Plans (continued)

```
SQL Plan Monitoring Details
====================================================================
================================================================
| Id   |      Operation       |    Name     |  Rows    | Cost |
Time    | Start   | Starts |   Rows    | Activity  | Activity Detail |
|      |                      |             | (Estim) |      |
Active(s) | Active |        | (Actual) | (percent) |   (sample #)
|
====================================================================
================================================================
|    0 | SELECT STATEMENT     |             |          |      |
|        |      1 |          |           |                  |
|    1 |   SORT AGGREGATE     |             |          |      |
1 |     +6 |      1 |        0 |     10.00 | Cpu (1)          |
| -> 2 |    NESTED LOOPS      |             |          |      |
5 |     +8 |      1 |   96151K |           |                  |
| -> 3 |     INDEX RANGE SCAN | TEST_C_INDX |          |      |
5 |     +8 |      1 |     4807 |           |                  |
| -> 4 |     INDEX RANGE SCAN | TEST_C_INDX |          |      |
11 |     +2 |   4808 |   96151K |     90.00 | Cpu (9)         |
====================================================================
================================================================


SQL> /

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL Monitoring Report

SQL Text
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------

Global Information
 Status              :  EXECUTING
 Instance ID         :  1
 Session ID          :  138
 SQL ID              :  dkz7v96ym42c6
 SQL Execution ID    :  16777222
 Plan Hash Value     :  1643938535
 Execution Started   :  04/02/2008 20:12:46
 First Refresh Time  :  04/02/2008 20:12:54
 Last Refresh Time   :  04/02/2008 20:13:00

-----------------------------------------
| Elapsed |   Cpu   |  Other   | Buffer |
```

Practice 5-1: Extracting Execution Plans (continued)

```
| Time(s) | Time(s) | Waits(s) |  Gets  |
-----------------------------------------
|      12 |      12 |     0.01 |   201K |
-----------------------------------------


SQL Plan Monitoring Details
====================================================================
=================================================================
| Id   |      Operation       |     Name      |  Rows    | Cost |
Time    | Start  | Starts |   Rows    | Activity  | Activity Detail |
|      |                      |               | (Estim)  |      |
Active(s) | Active |        | (Actual) | (percent) |   (sample #)
|
====================================================================
=================================================================
|    0 | SELECT STATEMENT     |               |          |      |
|        |      1 |           |           |                 |
|    1 |   SORT AGGREGATE     |               |          |      |
1 |     +6 |      1 |        0 |      8.33 | Cpu (1)          |
| -> 2 |    NESTED LOOPS      |               |          |      |
7 |     +8 |      1 |     113M |           |                  |
| -> 3 |     INDEX RANGE SCAN | TEST_C_INDX |          |      |
7 |     +8 |      1 |     5664 |           |                  |
| -> 4 |     INDEX RANGE SCAN | TEST_C_INDX |          |      |
13 |     +2 |   5665 |     113M |     91.67 | Cpu (11)        |
====================================================================
=================================================================


SQL> /

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL Monitoring Report

SQL Text
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------

Global Information
 Status              :  EXECUTING
 Instance ID         :  1
 Session ID          :  138
 SQL ID              :  dkz7v96ym42c6
 SQL Execution ID    :  16777222
 Plan Hash Value     :  1643938535
 Execution Started   :  04/02/2008 20:12:46
 First Refresh Time  :  04/02/2008 20:12:54
```

```
 Last Refresh Time   :  04/02/2008 20:13:08

-----------------------------------------
| Elapsed |   Cpu   |  Other   | Buffer |
| Time(s) | Time(s) | Waits(s) |  Gets  |
-----------------------------------------
|      20 |      20 |     0.04 |   334K |
-----------------------------------------


SQL Plan Monitoring Details
====================================================================
=================================================================
| Id   |       Operation       |    Name      |  Rows    | Cost |
Time    | Start   | Starts |   Rows    | Activity  | Activity Detail |
|      |                       |              | (Estim) |      |
Active(s) | Active |        | (Actual) | (percent) |   (sample #)
|
====================================================================
=================================================================
|    0 | SELECT STATEMENT      |              |          |      |
|        |      1 |           |           |                 |
|    1 |   SORT AGGREGATE      |              |          |      |
1 |     +6 |      1 |        0 |      5.00 | Cpu (1)         |
| -> 2 |    NESTED LOOPS       |              |          |      |
15 |     +8 |      1 |     182M |            |                 |
| -> 3 |     INDEX RANGE SCAN | TEST_C_INDX |          |      |
15 |     +8 |      1 |     9082 |            |                 |
| -> 4 |     INDEX RANGE SCAN | TEST_C_INDX |          |      |
21 |     +2 |   9083 |     182M |     95.00 | Cpu (19)        |
====================================================================
=================================================================


SQL> /

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL Monitoring Report

SQL Text
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------

Global Information
 Status              :  EXECUTING
 Instance ID         :  1
 Session ID          :  138
 SQL ID              :  dkz7v96ym42c6
```

Practice 5-1: Extracting Execution Plans (continued)

```
 SQL Execution ID    :  16777222
 Plan Hash Value     :  1643938535
 Execution Started   :  04/02/2008 20:12:46
 First Refresh Time  :  04/02/2008 20:12:54
 Last Refresh Time   :  04/02/2008 20:13:17

-----------------------------------------
| Elapsed |   Cpu   |  Other   | Buffer |
| Time(s) | Time(s) | Waits(s) |  Gets  |
-----------------------------------------
|      28 |      28 |     0.05 |   468K |
-----------------------------------------


SQL Plan Monitoring Details
====================================================================
==================================================================
| Id   |      Operation       |     Name      |  Rows    | Cost |
Time     | Start   | Starts |    Rows     | Activity  | Activity Detail |
|      |                      |               | (Estim) |      |
Active(s) | Active |         | (Actual) | (percent) |   (sample #)
|
====================================================================
==================================================================
|    0 | SELECT STATEMENT     |               |         |      |
|       |      1 |           |           |                  |
|    1 |  SORT AGGREGATE      |               |         |      |
1 |     +6 |      1 |        0 |      3.45 | Cpu (1)         |
| -> 2 |   NESTED LOOPS       |               |         |      |
24 |     +8 |      1 |     250M |               |                |
| -> 3 |    INDEX RANGE SCAN | TEST_C_INDX |         |      |
24 |     +8 |      1 |    12518 |               |                |
| -> 4 |    INDEX RANGE SCAN | TEST_C_INDX |         |      |
30 |     +2 |  12519 |     250M |     96.55 | Cpu (28)        |
====================================================================
==================================================================


SQL> /

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
--------------------------------------------------------------------
--------------------------------------------------------------------
-----------------------------------------------------------------
SQL Monitoring Report

SQL Text
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------

Global Information
```

```
 Status              :  EXECUTING
 Instance ID         :  1
 Session ID          :  138
 SQL ID              :  dkz7v96ym42c6
 SQL Execution ID    :  16777222
 Plan Hash Value     :  1643938535
 Execution Started   :  04/02/2008 20:12:46
 First Refresh Time  :  04/02/2008 20:12:54
 Last Refresh Time   :  04/02/2008 20:13:29

-----------------------------------------
| Elapsed |   Cpu   |  Other   | Buffer |
| Time(s) | Time(s) | Waits(s) |  Gets  |
-----------------------------------------
|      40 |      40 |     0.06 |   669K |
-----------------------------------------


SQL Plan Monitoring Details
====================================================================
=================================================================
| Id   |       Operation        |     Name      |  Rows    | Cost |
Time    | Start   | Starts |    Rows    | Activity   | Activity Detail |
|      |                        |               | (Estim) |      |
Active(s) | Active |         | (Actual) | (percent) |    (sample #)
|
====================================================================
=================================================================
|    0 | SELECT STATEMENT       |               |          |      |
|        |       1 |           |            |                  |
|    1 |   SORT AGGREGATE       |               |          |      |
35 |      +6 |      1 |        0 |      4.88 | Cpu (2)          |
| -> 2 |    NESTED LOOPS        |               |          |      |
36 |      +8 |      1 |     353M |             |                  |
| -> 3 |     INDEX RANGE SCAN | TEST_C_INDX |          |      |
36 |      +8 |      1 |    17663 |             |                  |
| -> 4 |     INDEX RANGE SCAN | TEST_C_INDX |          |      |
42 |      +2 |  17664 |     353M |     95.12 | Cpu (39)         |
====================================================================
=================================================================



SQL> /

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL Monitoring Report

SQL Text
--------------------------------------------------------------------
--------------------------------------------------------------------
------------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
```

Practice 5-1: Extracting Execution Plans (continued)

```
-----------------------------------------------------------------------
-----------------------------------------------------------------------
-------------------------------------------

Global Information
 Status              :  DONE (ALL ROWS)
 Instance ID         :  1
 Session ID          :  138
 SQL ID              :  dkz7v96ym42c6
 SQL Execution ID    :  16777222
 Plan Hash Value     :  1643938535
 Execution Started   :  04/02/2008 20:12:46
 First Refresh Time  :  04/02/2008 20:12:54
 Last Refresh Time   :  04/02/2008 20:13:34

-------------------------------------------------
| Elapsed |   Cpu   |  Other   | Fetch | Buffer |
| Time(s) | Time(s) | Waits(s) | Calls |  Gets  |
-------------------------------------------------
|      46 |      46 |     0.06 |     1 |   760K |
-------------------------------------------------


SQL Plan Monitoring Details
=======================================================================
==================================================================
| Id |      Operation        |    Name      |  Rows    | Cost |   Time
| Start   | Starts |   Rows   | Activity   | Activity Detail |
|    |                       |              | (Estim) |      |
Active(s) | Active |         | (Actual) | (percent) |   (sample #)
|
=======================================================================
==================================================================
|  0 | SELECT STATEMENT      |              |         |      |
1 |    +48 |       1 |       1 |           |                 |
|  1 |   SORT AGGREGATE      |              |         |      |
43 |     +6 |       1 |       1 |      4.35 | Cpu (2)         |
|  2 |    NESTED LOOPS       |              |         |      |
41 |     +8 |       1 |    400M |           |                 |
|  3 |     INDEX RANGE SCAN | TEST_C_INDX |         |      |
41 |     +8 |       1 |   20000 |           |                 |
|  4 |     INDEX RANGE SCAN | TEST_C_INDX |         |      |
47 |     +2 |  20000 |    400M |     95.65 | Cpu (44)        |
=======================================================================
==================================================================


SQL>
```

6) After approximately 47 seconds (depending on your environment), you should see the following output in your session 1:

```
Session 1:
----------
```

```
…

  COUNT(*)
----------
 400000000

Elapsed: 00:00:47.21
SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options


[oracle@edrsr33p1-orcl Explain_Plan]$
```

7) From session 1, connect as the `EP` user in the SQL*Plus session.

```
Session 1:
----------

[oracle@edrsr33p1-orcl Explain_Plan]$ sqlplus ep/ep

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Apr 2 20:14:03 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
```

8) Use `PLAN_TABLE` to determine the execution plan of the query that was executed in step 4. What do you observe?
`select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1;`

   a) This time the execution plan uses a hash join on top of two index fast full scans.

```
Session 1:
----------

SQL> @ep_explain
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> explain plan for
  2  select count(*) from test t1, test t2 where t1.c=t2.c and
t1.c=1;

Explained.

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

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
Plan hash value: 3253233075

--------------------------------------------------------------------
------------------
| Id  | Operation              | Name         | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
------------------
|   0 | SELECT STATEMENT       |              |     1 |     6 |  2131
(99)| 00:00:22 |
|   1 |  SORT AGGREGATE        |              |     1 |     6 |
|        |
|*  2 |   HASH JOIN            |              |   400M|  2288M|  2131
(99)| 00:00:22 |
|*  3 |    INDEX FAST FULL SCAN| TEST_C_INDX | 20000 | 60000 |    13
(0)| 00:00:01 |
|*  4 |    INDEX FAST FULL SCAN| TEST_C_INDX | 20000 | 60000 |    13
(0)| 00:00:01 |
--------------------------------------------------------------------
------------------

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

   2 - access("T1"."C"="T2"."C")
   3 - filter("T1"."C"=1)
   4 - filter("T2"."C"=1)

18 rows selected.

SQL>

--------------------------------------------------------------

set echo on

set linesize 200 pagesize 1000

explain plan for
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1;

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

9) Now, you want to monitor the previous execution plan to compare it with the one generated in step 4. In addition, you want to make sure you use the correct plan this time. So, in your session 1, start autotrace, and be ready to execute the following query. Do not execute it yet as you need to start SQL Monitoring in your session 2:
`select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1;`

Practice 5-1: Extracting Execution Plans (continued)

```
Session 1:
----------

SQL> set autotrace on
SQL> @ep_execute
```

10) From your session 2, be ready to execute your SQL Monitoring command again. Do not execute it yet though.

```
Session 2:
----------

SQL> @ep_monitor

---------------------------------------------------------------

set echo on
set long 10000000
set longchunksize 10000000
set linesize 200
set pagesize 1000

exec dbms_lock.sleep(8);

select
dbms_sqltune.report_sql_monitor(sql_id=>'dkz7v96ym42c6',report_level
=>'ALL') from dual;
```

11) Start the execution of your query from session 1 by pressing [Enter]. **Note:** Move to next stepwithout waiting.

```
Session 1:
----------

SQL> @ep_execute
SQL> set echo on
SQL>
SQL> set timing on
SQL>
SQL> select count(*) from test t1, test t2 where t1.c=t2.c and
t1.c=1;

---------------------------------------------------------------

set echo on

set timing on

select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1;
```

12) From your session 2, start monitoring your query by pressing the return key. After the query is executed, enter "/" and go back to your SQL*Plus session as many times as necessary until session 1 is done with its execution. What do you observe?

   a) You can see that the optimizer uses a hash join on top of two index fast full scans. Looking at the various reports, you can clearly see how the optimizer processes a hash join by reading the driving index in memory first. This operation is quick. Though you cannot see it run, it is already done the first time you can look at it. Then the probe in performed on the index again. This operation takes more time. Also note that cost information is provided in the execution plan.

```
Session 2:
----------

SQL> @ep_monitor
SQL> set echo on
SQL> set long 10000000
SQL> set longchunksize 10000000
SQL> set linesize 200
SQL> set pagesize 1000
SQL>
SQL> exec dbms_lock.sleep(8);

PL/SQL procedure successfully completed.

SQL>
SQL> select
dbms_sqltune.report_sql_monitor(sql_id=>'dkz7v96ym42c6',report_level
=>'ALL') from dual;

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL Monitoring Report

SQL Text
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------

Global Information
 Status              :  EXECUTING
 Instance ID         :  1
 Session ID          :  124
 SQL ID              :  dkz7v96ym42c6
 SQL Execution ID    :  16777223
 Plan Hash Value     :  3253233075
 Execution Started   :  04/02/2008 20:14:41
 First Refresh Time  :  04/02/2008 20:14:49
```

Practice 5-1: Extracting Execution Plans (continued)

```
 Last Refresh Time   :  04/02/2008 20:14:49

-----------------------------------------
| Elapsed |   Cpu   |  Other   | Buffer |
| Time(s) | Time(s) | Waits(s) |  Gets  |
-----------------------------------------
|    6.29 |    6.29 |     0.00 |      8 |
-----------------------------------------


SQL Plan Monitoring Details
====================================================================
====================================================================
==========
| Id   |        Operation          |    Name     |  Rows    | Cost |
Time     | Start  | Starts |   Rows    | Memory | Activity  | Activity
Detail |
|      |                           |             | (Estim) |      |
Active(s) | Active |        | (Actual) |        | (percent) |
(sample #)    |
====================================================================
====================================================================
==========
|    0 | SELECT STATEMENT          |             |         | 2131 |
|        |      1 |          |        |           |
|
|    1 |   SORT AGGREGATE          |             |       1 |      |
8 |     +0 |      1 |        0 |        |     25.00 | Cpu (2)
|
| -> 2 |    HASH JOIN              |             |    400M | 2131 |
8 |     +1 |      1 |     113M |  1138K |     75.00 | Cpu (6)
|
| -> 3 |     INDEX FAST FULL SCAN | TEST_C_INDX |   20000 |   13 |
1 |     +8 |      1 |    20000 |        |           |
|
| -> 4 |     INDEX FAST FULL SCAN | TEST_C_INDX |   20000 |   13 |
1 |     +8 |      1 |     5632 |        |           |
|
====================================================================
====================================================================
==========


SQL>
SQL> /

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL Monitoring Report

SQL Text
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------
```

Practice 5-1: Extracting Execution Plans (continued)

```
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
---------------------------------------------------------------------
---------------------------------------------------------------------
-------------------------------------------

Global Information
 Status              :  EXECUTING
 Instance ID         :  1
 Session ID          :  124
 SQL ID              :  dkz7v96ym42c6
 SQL Execution ID    :  16777223
 Plan Hash Value     :  3253233075
 Execution Started   :  04/02/2008 20:14:41
 First Refresh Time  :  04/02/2008 20:14:49
 Last Refresh Time   :  04/02/2008 20:14:51

-----------------------------------------
| Elapsed |   Cpu   |  Other   | Buffer |
| Time(s) | Time(s) | Waits(s) |  Gets  |
-----------------------------------------
|    7.86 |    7.86 |     0.00 |     10 |
-----------------------------------------


SQL Plan Monitoring Details
=====================================================================
=====================================================================
==========
| Id   |         Operation          |    Name      |  Rows    | Cost |
Time     | Start   | Starts |    Rows    | Memory | Activity   | Activity
Detail |
|      |                            |              | (Estim)  |      |
Active(s) | Active |        | (Actual) |        | (percent) |
(sample #)    |
=====================================================================
=====================================================================
==========
|    0 | SELECT STATEMENT           |              |          | 2131 |
|        |      1 |          |        |           |
|
|    1 |   SORT AGGREGATE           |              |        1 |      |
8 |     +0 |      1 |        0 |        |     20.00 | Cpu (2)
|
| -> 2 |    HASH JOIN               |              |     400M | 2131 |
10 |     +1 |      1 |     133M |  1138K |     80.00 | Cpu (8)
|
|    3 |     INDEX FAST FULL SCAN | TEST_C_INDX |    20000 |    13 |
1 |     +8 |      1 |    20000 |        |           |
|
| -> 4 |     INDEX FAST FULL SCAN | TEST_C_INDX |    20000 |    13 |
3 |     +8 |      1 |     6656 |        |           |
|
=====================================================================
=====================================================================
==========
```

Practice 5-1: Extracting Execution Plans (continued)

```
SQL> /

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL Monitoring Report

SQL Text
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------

Global Information
 Status               :  EXECUTING
 Instance ID          :  1
 Session ID           :  124
 SQL ID               :  dkz7v96ym42c6
 SQL Execution ID     :  16777223
 Plan Hash Value      :  3253233075
 Execution Started    :  04/02/2008 20:14:41
 First Refresh Time   :  04/02/2008 20:14:49
 Last Refresh Time    :  04/02/2008 20:14:55

-----------------------------------------
| Elapsed |   Cpu   |  Other   | Buffer |
| Time(s) | Time(s) | Waits(s) |  Gets  |
-----------------------------------------
|      12 |      12 |     0.00 |     15 |
-----------------------------------------


SQL Plan Monitoring Details
====================================================================
====================================================================
==========
| Id   |        Operation         |    Name     |  Rows   | Cost |
Time     | Start  | Starts |   Rows    | Memory | Activity  | Activity
Detail |
|      |                          |             | (Estim) |      |
Active(s) | Active |         | (Actual) |        | (percent) |
(sample #)     |
====================================================================
====================================================================
==========
|    0 | SELECT STATEMENT         |             |         | 2131 |
|        |      1 |          |        |           |
|
|    1 |   SORT AGGREGATE         |             |       1 |      |
8 |     +0 |      1 |        0 |        |     14.29 | Cpu (2)
|
```

Practice 5-1: Extracting Execution Plans (continued)

```
| -> 2 |    HASH JOIN             |              |    400M | 2131 |
14 |     +1 |      1 |     184M |  1138K |     85.71 | Cpu (12)
|
|    3 |     INDEX FAST FULL SCAN | TEST_C_INDX |   20000 |   13 |
1 |     +8 |      1 |    20000 |        |            |
|
| -> 4 |     INDEX FAST FULL SCAN | TEST_C_INDX |   20000 |   13 |
7 |     +8 |      1 |     9216 |        |            |
|
=====================================================================
=====================================================================
==========


SQL> /

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
---------------------------------------------------------------------
---------------------------------------------------------------------
-----------------------------------------------------------------
SQL Monitoring Report

SQL Text
---------------------------------------------------------------------
---------------------------------------------------------------------
-------------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
---------------------------------------------------------------------
---------------------------------------------------------------------
-------------------------------------------


Global Information
 Status              :  EXECUTING
 Instance ID         :  1
 Session ID          :  124
 SQL ID              :  dkz7v96ym42c6
 SQL Execution ID    :  16777223
 Plan Hash Value     :  3253233075
 Execution Started   :  04/02/2008 20:14:41
 First Refresh Time  :  04/02/2008 20:14:49
 Last Refresh Time   :  04/02/2008 20:15:05

-----------------------------------------
| Elapsed |   Cpu   |  Other   | Buffer |
| Time(s) | Time(s) | Waits(s) |  Gets  |
-----------------------------------------
|      22 |      22 |     0.01 |     28 |
-----------------------------------------


SQL Plan Monitoring Details
=====================================================================
=====================================================================
==========
```

```
| Id   |         Operation          |    Name     |  Rows   | Cost |
Time     | Start  | Starts |   Rows    | Memory | Activity  | Activity
Detail |
|      |                            |             | (Estim) |      |
Active(s) | Active |         | (Actual) |        | (percent) |
(sample #)    |
====================================================================
====================================================================
==========
|    0 | SELECT STATEMENT           |             |         | 2131 |
|         |      1 |          |        |           |
|
|    1 |   SORT AGGREGATE           |             |       1 |      |
8 |     +0 |      1 |        0 |        |      8.33 | Cpu (2)
|
| -> 2 |    HASH JOIN               |             |    400M | 2131 |
24 |     +1 |      1 |     317M |  1138K |     91.67 | Cpu (22)
|
|    3 |     INDEX FAST FULL SCAN | TEST_C_INDX |   20000 |   13 |
1 |     +8 |      1 |    20000 |        |           |
|
| -> 4 |     INDEX FAST FULL SCAN | TEST_C_INDX |   20000 |   13 |
17 |     +8 |      1 |    15872 |        |           |
|
====================================================================
====================================================================
==========


SQL> /

DBMS_SQLTUNE.REPORT_SQL_MONITOR(SQL_ID=>'DKZ7V96YM42C6',SESSION_ID=>
:SESSID,REPORT_LEVEL=>'ALL')
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL Monitoring Report

SQL Text
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------------------------------


Global Information
 Status              :  DONE (ALL ROWS)
 Instance ID         :  1
 Session ID          :  124
 SQL ID              :  dkz7v96ym42c6
 SQL Execution ID    :  16777223
 Plan Hash Value     :  3253233075
 Execution Started   :  04/02/2008 20:14:41
 First Refresh Time  :  04/02/2008 20:14:49
 Last Refresh Time   :  04/02/2008 20:15:11
```

Practice 5-1: Extracting Execution Plans (continued)

```
-------------------------------------------------
| Elapsed |   Cpu   |  Other   | Fetch | Buffer |
| Time(s) | Time(s) | Waits(s) | Calls |  Gets  |
-------------------------------------------------
|      28 |      28 |     0.01 |     1 |     37 |
-------------------------------------------------


SQL Plan Monitoring Details
====================================================================
====================================================================
========
| Id |         Operation         |    Name      |  Rows   | Cost |
Time     | Start   | Starts |   Rows    | Memory | Activity   | Activity
Detail |
|    |                           |              | (Estim) |      |
Active(s) | Active |        | (Actual) | (Max)   | (percent) |
(sample #)     |
====================================================================
====================================================================
========
|  0 | SELECT STATEMENT          |              |         | 2131 |
1 |    +30 |      1 |        1 |        |           |
|
|  1 |   SORT AGGREGATE          |              |       1 |      |
31 |     +0 |      1 |        1 |        |     10.00 | Cpu (3)
|
|  2 |    HASH JOIN              |              |    400M | 2131 |
30 |     +1 |      1 |     400M |  1138K |     90.00 | Cpu (27)
|
|  3 |     INDEX FAST FULL SCAN | TEST_C_INDX |   20000 |   13 |
1 |     +8 |      1 |    20000 |        |           |
|
|  4 |     INDEX FAST FULL SCAN | TEST_C_INDX |   20000 |   13 |
23 |     +8 |      1 |    20000 |        |           |
|
====================================================================
====================================================================
========


SQL>
```

13) When your query is executed, what do you observe in your session 1?

a) Session 1 also reports the same execution plan as the one you observed in session 2.

```
Session 1:
----------

…

  COUNT(*)
```

```
----------
 400000000

Elapsed: 00:00:30.70

Execution Plan
----------------------------------------------------------
Plan hash value: 3253233075

--------------------------------------------------------------------
------------------
| Id  | Operation              | Name         | Rows   | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
------------------
|   0 | SELECT STATEMENT       |              |      1 |     6 |  2131
(99)| 00:00:22 |
|   1 |  SORT AGGREGATE        |              |      1 |     6 |
|          |
|*  2 |   HASH JOIN            |              |   400M|  2288M|  2131
(99)| 00:00:22 |
|*  3 |    INDEX FAST FULL SCAN| TEST_C_INDX | 20000 | 60000 |    13
(0)| 00:00:01 |
|*  4 |    INDEX FAST FULL SCAN| TEST_C_INDX | 20000 | 60000 |    13
(0)| 00:00:01 |
--------------------------------------------------------------------
------------------

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

   2 - access("T1"."C"="T2"."C")
   3 - filter("T1"."C"=1)
   4 - filter("T2"."C"=1)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
         90  consistent gets
          0  physical reads
          0  redo size
        418  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
```

14) In session 1, disable autotrace.

```

Session 1:
----------
```

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

15) From your session 1, how can you ensure that you gather all execution plan statistics for the following query without changing any session parameters? Implement your solution.
`select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1;`

```
Session 1:
----------

SQL> @ep_execute_with_all
SQL> set echo on
SQL>
SQL> set timing on
SQL>
SQL> select /*+ gather_plan_statistics */ count(*) from test t1,
test t2 where t1.c=t2.c and t1.c=1;

  COUNT(*)
----------
 400000000

Elapsed: 00:00:59.95
SQL>
```

16) From your session 1, retrieve all execution plans corresponding to all the queries you executed since the beginning of this lab. What is your conclusion?

a) The easiest way to find out all the plans is to look at the content of the SGA using the `dbms_xplan.display_cursor` function. First, you must determine the `SQL_Ids` used to represent your queries. You essentially have two queries, and one that has two children. You should now understand what happened at step 4. The fact that there was no cost information is probably due to the use of the rule-based optimizer instead of the cost-based one.

```
Session 1:
----------

SQL> @ep_retrieve_all_plans
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> col sql_text format a50
SQL>
SQL> select sql_id,plan_hash_value,sql_text from v$sql where
sql_text like '%from test t1, test t2%';

SQL_ID        PLAN_HASH_VALUE SQL_TEXT
------------- --------------- ---------------------------------------
------------
```

Practice 5-1: Extracting Execution Plans (continued)

```
dkz7v96ym42c6      3253233075 select count(*) from test t1, test t2
where t1.c=t
                              2.c and t1.c=1

dkz7v96ym42c6      1643938535 select count(*) from test t1, test t2
where t1.c=t
                              2.c and t1.c=1

8w580dd6ncgqw      3253233075 select /*+ gather_plan_statistics */
count(*) from
                               test t1, test t2 where t1.c=t2.c and
t1.c=1

0w0va2d7hhtxa      3253233075 explain plan for select count(*) from
test t1, tes
                              t t2 where t1.c=t2.c and t1.c=1

dd09kf5dnp1gt       903671040 select sql_id,plan_hash_value,sql_text
from v$sql
                              where sql_text like '%from test t1,
test t2%'

32fqwuk16uf23      3253233075 EXPLAIN PLAN SET
STATEMENT_ID='PLUS2140495' FOR se
                              lect count(*) from test t1, test t2
where t1.c=t2.
                              c and t1.c=1


6 rows selected.

Elapsed: 00:00:00.02
SQL>
SQL> select * from
table(dbms_xplan.display_cursor('dkz7v96ym42c6',null,'TYPICAL'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL_ID  dkz7v96ym42c6, child number 0
-------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1

Plan hash value: 3253233075

--------------------------------------------------------------------
------------------
| Id  | Operation              | Name         | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
------------------
|   0 | SELECT STATEMENT       |              |       |       |  2131
(100)|          |
|   1 |  SORT AGGREGATE        |              |     1 |     6 |
|          |
```

Practice 5-1: Extracting Execution Plans (continued)

```
|*  2 |   HASH JOIN            |              |   400M|  2288M|  2131
(99)| 00:00:22 |
|*  3 |    INDEX FAST FULL SCAN| TEST_C_INDX | 20000 | 60000 |    13
(0)| 00:00:01 |
|*  4 |    INDEX FAST FULL SCAN| TEST_C_INDX | 20000 | 60000 |    13
(0)| 00:00:01 |
--------------------------------------------------------------------
------------------

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

   2 - access("T1"."C"="T2"."C")
   3 - filter("T1"."C"=1)
   4 - filter("T2"."C"=1)

SQL_ID  dkz7v96ym42c6, child number 1
-------------------------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1

Plan hash value: 1643938535

------------------------------------------
| Id  | Operation          | Name        |
------------------------------------------
|   0 | SELECT STATEMENT   |             |
|   1 |  SORT AGGREGATE    |             |
|   2 |   NESTED LOOPS     |             |
|*  3 |    INDEX RANGE SCAN| TEST_C_INDX |
|*  4 |    INDEX RANGE SCAN| TEST_C_INDX |
------------------------------------------

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

   3 - access("T1"."C"=1)
   4 - access("T1"."C"="T2"."C")

Note
-----
   - rule based optimizer used (consider using cbo)


49 rows selected.

Elapsed: 00:00:00.04
SQL>
SQL> select * from
table(dbms_xplan.display_cursor('8w580dd6ncgqw',null,'ADVANCED
ALLSTATS LAST'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL_ID  8w580dd6ncgqw, child number 0
-------------------------------------
```

Practice 5-1: Extracting Execution Plans (continued)

```
select /*+ gather_plan_statistics */ count(*) from test t1, test t2
where t1.c=t2.c and t1.c=1

Plan hash value: 3253233075

--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------
| Id  | Operation              | Name         | Starts | E-Rows |E-
Bytes| Cost (%CPU)| E-Time    | A-Rows |   A-Time    | Buffers |  OMem
|  1Mem | Used-Mem |
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------
|   1 |  SORT AGGREGATE        |              |      1 |      1 |
6 |            |          |      1 |00:00:59.94 |      90 |       |
|          |
|*  2 |   HASH JOIN            |              |      1 |    400M|
2288M|  2131  (99)| 00:00:22 |    400M|00:00:00.01 |      90 |
1155K|  1155K| 1115K (0)|
|*  3 |    INDEX FAST FULL SCAN| TEST_C_INDX |      1 |  20000 |
60000 |    13   (0)| 00:00:01 |  20000 |00:00:00.02 |      45 |
|       |          |
|*  4 |    INDEX FAST FULL SCAN| TEST_C_INDX |      1 |  20000 |
60000 |    13   (0)| 00:00:01 |  20000 |00:00:00.16 |      45 |
|       |          |
--------------------------------------------------------------------
--------------------------------------------------------------------
-------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$1
   3 - SEL$1 / T1@SEL$1
   4 - SEL$1 / T2@SEL$1

Outline Data
-------------

  /*+
      BEGIN_OUTLINE_DATA
      IGNORE_OPTIM_EMBEDDED_HINTS
      OPTIMIZER_FEATURES_ENABLE('11.1.0.6')
      DB_VERSION('11.1.0.6')
      ALL_ROWS
      OUTLINE_LEAF(@"SEL$1")
      INDEX_FFS(@"SEL$1" "T1"@"SEL$1" ("TEST"."C"))
      INDEX_FFS(@"SEL$1" "T2"@"SEL$1" ("TEST"."C"))
      LEADING(@"SEL$1" "T1"@"SEL$1" "T2"@"SEL$1")
      USE_HASH(@"SEL$1" "T2"@"SEL$1")
      END_OUTLINE_DATA
  */

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

```
   2 - access("T1"."C"="T2"."C")
   3 - filter("T1"."C"=1)
   4 - filter("T2"."C"=1)

Column Projection Information (identified by operation id):
-----------------------------------------------------------

   1 - (#keys=0) COUNT(*)[22]
   2 - (#keys=1)
   3 - "T1"."C"[NUMBER,22]
   4 - "T2"."C"[NUMBER,22]


55 rows selected.

Elapsed: 00:00:00.12
SQL>

--------------------------------------------------------------

set echo on

set linesize 200 pagesize 1000

col sql_text format a50

select sql_id,plan_hash_value,sql_text from v$sql where sql_text
like '%from test t1, test t2%';

select * from
table(dbms_xplan.display_cursor('dkz7v96ym42c6',null,'TYPICAL'));

select * from
table(dbms_xplan.display_cursor('8w580dd6ncgqw',null,'ADVANCED
ALLSTATS LAST'));
```

17) From session 1, try to retrieve your execution plans from the Automatic Workload Repository. What happens and why?

a) You can use the previously found `SQL_Ids` to search through the `DBA_HIST_SQLTEXT` view. You should see that right now, none of your queries were stored in the AWR.

```
Session 1:
----------

SQL> @ep_retrieve_awr
SQL> set echo on
SQL>
SQL> set linesize 200
SQL>
SQL> SELECT SQL_ID, SQL_TEXT FROM dba_hist_sqltext
  2  WHERE SQL_ID in ('dkz7v96ym42c6','8w580dd6ncgqw');

no rows selected
```

Practice 5-1: Extracting Execution Plans (continued)

```
SQL>

Elapsed: 00:00:00.01
SQL>

--------------------------------------------------------------

set echo on

set linesize 200

SELECT SQL_ID, SQL_TEXT FROM dba_hist_sqltext
WHERE SQL_ID in ('dkz7v96ym42c6','8w580dd6ncgqw');
```

18) How can you ensure that you retrieve your queries from the Automatic Workload Repository? Implement your solution.

   a) You must flush the SGA information to the AWR. You can use `DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT` for this purpose.

```
Session 1:
----------

SQL> @ep_save_awr
SQL> set echo on
SQL>
SQL> EXEC DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT('ALL');

PL/SQL procedure successfully completed.

Elapsed: 00:00:02.27
SQL>

--------------------------------------------------------------

set echo on

EXEC DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT('ALL');
```

19) Verify that your solution works.

```
Session 1:
----------

SQL> @ep_show_awr
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT PLAN_TABLE_OUTPUT
  2  FROM
  3  TABLE (DBMS_XPLAN.DISPLAY_AWR('dkz7v96ym42c6'));
```

Practice 5-1: Extracting Execution Plans (continued)

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL_ID dkz7v96ym42c6
--------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1

Plan hash value: 1643938535

------------------------------------------
| Id  | Operation          | Name         |
------------------------------------------
|   0 | SELECT STATEMENT   |              |
|   1 |  SORT AGGREGATE    |              |
|   2 |   NESTED LOOPS     |              |
|   3 |    INDEX RANGE SCAN| TEST_C_INDX |
|   4 |    INDEX RANGE SCAN| TEST_C_INDX |
------------------------------------------

Note
-----
   - rule based optimizer used (consider using cbo)

SQL_ID dkz7v96ym42c6
--------------------
select count(*) from test t1, test t2 where t1.c=t2.c and t1.c=1

Plan hash value: 3253233075

--------------------------------------------------------------------
------------------
| Id  | Operation              | Name        | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
------------------
|   0 | SELECT STATEMENT       |             |       |       |   543
(100)|          |
|   1 |  SORT AGGREGATE        |             |     1 |     6 |
|          |
|   2 |   HASH JOIN            |             |   100M|   572M|   543
(98)| 00:00:06 |
|   3 |    INDEX FAST FULL SCAN| TEST_C_INDX | 10000 | 30000 |     8
(0)| 00:00:01 |
|   4 |    INDEX FAST FULL SCAN| TEST_C_INDX | 10000 | 30000 |     8
(0)| 00:00:01 |
--------------------------------------------------------------------
------------------


36 rows selected.

Elapsed: 00:00:00.04
SQL>
SQL> SELECT PLAN_TABLE_OUTPUT
  2  FROM
```

Practice 5-1: Extracting Execution Plans (continued)

```
  3  TABLE
(DBMS_XPLAN.DISPLAY_AWR('8w580dd6ncgqw',null,null,'TYPICAL ALLSTATS
LAST'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL_ID 8w580dd6ncgqw
--------------------
select /*+ gather_plan_statistics */ count(*) from test t1, test t2
where t1.c=t2.c and t1.c=1

Plan hash value: 3253233075

--------------------------------------------------------------------
-------------------
| Id  | Operation              | Name         | E-Rows |E-Bytes| Cost
(%CPU)| E-Time   |
--------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT       |              |        |       |
2131 (100)|          |
|   1 |  SORT AGGREGATE        |              |      1 |     6 |
|          |
|   2 |   HASH JOIN            |              |    400M|  2288M|
2131  (99)| 00:00:22 |
|   3 |    INDEX FAST FULL SCAN| TEST_C_INDX |  20000 | 60000 |
13   (0)| 00:00:01 |
|   4 |    INDEX FAST FULL SCAN| TEST_C_INDX |  20000 | 60000 |
13   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------

Note
-----
   - Warning: basic plan statistics not available. These are only
collected when:
       * hint 'gather_plan_statistics' is used for the statement or
       * parameter 'statistics_level' is set to 'ALL', at session or
system level


23 rows selected.

Elapsed: 00:00:00.02
SQL>

--------------------------------------------------------------

set echo on

set linesize 200 pagesize 1000

SELECT PLAN_TABLE_OUTPUT
FROM
TABLE (DBMS_XPLAN.DISPLAY_AWR('dkz7v96ym42c6'));
```

Practice 5-1: Extracting Execution Plans (continued)

```
SELECT PLAN_TABLE_OUTPUT
FROM
TABLE (DBMS_XPLAN.DISPLAY_AWR('8w580dd6ncgqw',null,null,'TYPICAL
ALLSTATS LAST'));
```

20) Exit from both SQL*Plus sessions, and clean up your environment by executing the `ep_cleanup.sh` script from one of your terminal sessions.

```
Session 1:
----------

SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Explain_Plan]$ ./ep_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Wed Apr 2 20:19:37 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user ep cascade;

User dropped.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Explain_Plan]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Explain_Plan

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1
```

Practice 5-1: Extracting Execution Plans (continued)

```
export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @ep_cleanup.sql

--------------------------------------------------------------

set echo on

drop user ep cascade;

exit;
```

Do not forget to exit from your session 2:

```
Session 2:
----------

SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Explain_Plan]$
```

Practices for Lesson 6

Practice 6-1: 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 located in your `/home/oracle/solutions/Star_Schema_Tuning` directory.

```
[oracle@edrsr33p1-orcl Star_Schema_Tuning]$
./setup_star_schema_lab.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Mar 21 00:17:20
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
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 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Star_Schema_Tuning]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Star_Schema_Tuning

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin

sqlplus / as sysdba <<FIN!

set echo on

grant dba to sh;
```

Practice 6-1: Star Schema Tuning (continued)

```
alter user sh identified by sh account unlock;

FIN!
```

2) From the same terminal window, start a SQL*Plus session connected as the SH user and do not disconnect from it until this lab 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 `query.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.

```
[oracle@edrsr33p1-orcl Star_Schema_Tuning]$ sqlplus sh/sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Mar 21 00:31:10
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> @first_run
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
```

Practice 6-1: Star Schema Tuning (continued)

```
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
…
Montara                        1999-02      1618.01
Indirect             Quartzhill                     1999-02
412.83

41 rows selected.

Elapsed: 00:00:00.68

Execution Plan
----------------------------------------------------------
Plan hash value: 1647000731

-------------------------------------------------------------------------------------------------------------
| Id  | Operation                       | Name      | Rows  | Bytes | Cost (%CPU)| Time     | Pstart| Pstop |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                |           |  1144 | 96096 |   935   (4)| 00:00:12 |       |       |
|   1 |  HASH GROUP BY                  |           |  1144 | 96096 |   935   (4)| 00:00:12 |       |       |
|*  2 |   HASH JOIN                     |           |  6231 |   511K|   934   (3)| 00:00:12 |       |       |
|*  3 |    TABLE ACCESS FULL            | CHANNELS  |     2 |    42 |     3   (0)| 00:00:01 |       |       |
|*  4 |    HASH JOIN                    |           | 12462 |   766K|   930   (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|   911   (3)| 00:00:11 |       |       |
|*  8 |      TABLE ACCESS FULL          | CUSTOMERS |   383 |  9958 |   406   (1)| 00:00:05 |       |       |
|   9 |      PARTITION RANGE JOIN-FILTER|           |   918K|    18M|   498   (4)| 00:00:06 |:BF0000|:BF0000|
|  10 |       TABLE ACCESS FULL         | SALES     |   918K|    18M|   498   (4)| 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
----------------------------------------------------------
      16783  recursive calls
          0  db block gets
       5491  consistent gets
       2021  physical reads
          0  redo size
       1888  bytes sent via SQL*Net to client
        442  bytes received via SQL*Net from client
```

Practice 6-1: Star Schema Tuning (continued)

```
          4  SQL*Net roundtrips to/from client
        141  sorts (memory)
          0  sorts (disk)
         41  rows processed

SQL>
SQL>

---------------------------------------------------------------

set echo on

alter system flush shared_pool;
alter system flush buffer_cache;

set pagesize 200
set linesize 250
set timing on
set autotrace on

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;
```

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.

   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: 00:00:00.10
SQL> alter system flush buffer_cache;
```

Practice 6-1: Star Schema Tuning (continued)

```
System altered.

Elapsed: 00:00:00.20
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             San Francisco                  2000-04
13227.99
Indirect             Montara                        2000-04
1319
Indirect             Cloverdale                     2000-04
7.27
…
Indirect             Pala                           2000-03
Indirect             Montara                        1999-02
1618.01
Indirect             Quartzhill                     1999-02
412.83
Indirect             San Francisco                  1999-01
3058.27
Indirect             Pala                           1999-01
3263.93

41 rows selected.

Elapsed: 00:00:00.33

Execution Plan
----------------------------------------------------------
Plan hash value: 2525768690
```

```
----------------------------------------------------------------------------------------------------
----------
| Id  | Operation                          | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
Pstart| Pstop |
----------------------------------------------------------------------------------------------------
----------
|   0 | SELECT STATEMENT                   |                   |     3 |   252 |   983   (1)| 00:00:12 |
|       |
|   1 |  HASH GROUP BY                     |                   |     3 |   252 |   983   (1)| 00:00:12 |
|       |
|*  2 |   HASH JOIN                        |                   |     3 |   252 |   982   (1)| 00:00:12 |
|       |
|*  3 |    HASH JOIN                       |                   |     7 |   441 |   978   (1)| 00:00:12 |
|       |
|*  4 |     HASH JOIN                      |                   |    27 |  1269 |   960   (1)| 00:00:12 |
|       |
|*  5 |      TABLE ACCESS FULL             | CUSTOMERS         |   383 |  9958 |   406   (1)| 00:00:05 |
|       |
|   6 |      PARTITION RANGE SUBQUERY      |                   |   507 | 10647 |   553   (1)| 00:00:07
|KEY(SQ)|KEY(SQ)|
|   7 |       TABLE ACCESS BY LOCAL INDEX ROWID| SALES         |   507 | 10647 |   553   (1)| 00:00:07
|KEY(SQ)|KEY(SQ)|
|   8 |        BITMAP CONVERSION TO ROWIDS |                   |       |       |            |          |
|       |
|   9 |         BITMAP AND                 |                   |       |       |            |          |
|       |
|  10 |          BITMAP MERGE              |                   |       |       |            |          |
|       |
|  11 |           BITMAP KEY ITERATION     |                   |       |       |            |          |
|       |
|  12 |            BUFFER SORT             |                   |       |       |            |          |
|       |
|* 13 |             TABLE ACCESS FULL      | CHANNELS          |     2 |    42 |     3   (0)| 00:00:01 |
|       |
|* 14 |            BITMAP INDEX RANGE SCAN | SALES_CHANNEL_BIX |       |       |            |
|KEY(SQ)|KEY(SQ)|
|  15 |          BITMAP MERGE              |                   |       |       |            |          |
|       |
|  16 |           BITMAP KEY ITERATION     |                   |       |       |            |          |
|       |
|  17 |            BUFFER SORT             |                   |       |       |            |          |
|       |
|* 18 |             TABLE ACCESS FULL      | TIMES             |   365 |  5840 |    18   (0)| 00:00:01 |
|       |
|* 19 |            BITMAP INDEX RANGE SCAN | SALES_TIME_BIX    |       |       |            |
|KEY(SQ)|KEY(SQ)|
|  20 |          BITMAP MERGE              |                   |       |       |            |          |
|       |
|  21 |           BITMAP KEY ITERATION     |                   |       |       |            |          |
|       |
|  22 |            BUFFER SORT             |                   |       |       |            |          |
|       |
|* 23 |             TABLE ACCESS FULL      | CUSTOMERS         |   383 |  9958 |   406   (1)| 00:00:05 |
|       |
|* 24 |            BITMAP INDEX RANGE SCAN | SALES_CUST_BIX    |       |       |            |
|KEY(SQ)|KEY(SQ)|
|* 25 |     TABLE ACCESS FULL              | TIMES             |   365 |  5840 |    18   (0)| 00:00:01 |
|       |
|* 26 |    TABLE ACCESS FULL               | CHANNELS          |     2 |    42 |     3   (0)| 00:00:01 |
|       |
----------------------------------------------------------------------------------------------------
----------

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

   2 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
   3 - access("S"."TIME_ID"="T"."TIME_ID")
   4 - access("S"."CUST_ID"="C"."CUST_ID")
   5 - filter("C"."CUST_STATE_PROVINCE"='CA')
  13 - filter("CH"."CHANNEL_DESC"='Catalog' OR "CH"."CHANNEL_DESC"='Internet')
  14 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
  18 - 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')
  19 - access("S"."TIME_ID"="T"."TIME_ID")
  23 - filter("C"."CUST_STATE_PROVINCE"='CA')
  24 - access("S"."CUST_ID"="C"."CUST_ID")
  25 - 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')
  26 - filter("CH"."CHANNEL_DESC"='Catalog' OR "CH"."CHANNEL_DESC"='Internet')

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



Statistics
----------------------------------------------------------
      17362  recursive calls
          0  db block gets
      36930  consistent gets
```

```
       2130  physical reads
          0  redo size
       1896  bytes sent via SQL*Net to client
        442  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
        147  sorts (memory)
          0  sorts (disk)
         41  rows processed

SQL>

--------------------------------------------------------------

set echo on


alter system flush shared_pool;
alter system flush buffer_cache;

set pagesize 200
set linesize 250
set timing on
set autotrace on


ALTER SESSION SET star_transformation_enabled=TEMP_DISABLE;


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;
```

4) How would you enhance the previous optimization without changing the SH schema?

   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> set echo on
SQL>
SQL> alter system flush shared_pool;

System altered.

Elapsed: 00:00:00.10
SQL> alter system flush buffer_cache;
```

Practice 6-1: Star Schema Tuning (continued)

```
System altered.

Elapsed: 00:00:00.21
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             San Francisco                  2000-04
13227.99
Indirect             Montara                        2000-04
1319
…
Indirect             San Francisco                  1999-01
3058.27
Indirect             Pala                           1999-01
3263.93

41 rows selected.

Elapsed: 00:00:00.30

Execution Plan
----------------------------------------------------------
Plan hash value: 163104418

----------------------------------------------------------------------------------------------------------
------------------
| Id  | Operation                           | Name                      | Rows  | Bytes | Cost (%CPU)| Time
| Pstart| Pstop |
----------------------------------------------------------------------------------------------------------
------------------
|   0 | SELECT STATEMENT                    |                           |     3 |   219 |   985   (1)| 00:00:12
|       |       |
|   1 |  TEMP TABLE TRANSFORMATION          |                           |       |       |            |
|       |       |
|   2 |   LOAD AS SELECT                    | SYS_TEMP_0FD9D660F_8947E |       |       |            |
|       |       |
|*  3 |    TABLE ACCESS FULL                | CUSTOMERS                 |   383 |  9958 |   406   (1)| 00:00:05
|       |       |
```

```
|   4 |   HASH GROUP BY                       |                            |     3 |   219 |   579   (2)| 00:00:07
|     |       |
|*  5 |    HASH JOIN                          |                            |     3 |   219 |   578   (1)| 00:00:07
|     |       |
|*  6 |     HASH JOIN                         |                            |     7 |   364 |   574   (1)| 00:00:07
|     |       |
|*  7 |      HASH JOIN                        |                            |    28 |  1008 |   556   (1)| 00:00:07
|     |       |
|   8 |       TABLE ACCESS FULL               | SYS_TEMP_0FD9D660F_8947E |   383 |  5745 |     2   (0)| 00:00:01
|     |       |
|   9 |       PARTITION RANGE SUBQUERY        |                            |   507 | 10647 |   553   (1)| 00:00:07
|KEY(SQ)|KEY(SQ)|
|  10 |        TABLE ACCESS BY LOCAL INDEX ROWID| SALES                    |   507 | 10647 |   553   (1)| 00:00:07
|KEY(SQ)|KEY(SQ)|
|  11 |         BITMAP CONVERSION TO ROWIDS   |                            |       |       |            |
|     |       |
|  12 |          BITMAP AND                   |                            |       |       |            |
|     |       |
|  13 |           BITMAP MERGE                |                            |       |       |            |
|     |       |
|  14 |            BITMAP KEY ITERATION       |                            |       |       |            |
|     |       |
|  15 |             BUFFER SORT               |                            |       |       |            |
|     |       |
|* 16 |              TABLE ACCESS FULL        | CHANNELS                   |     2 |    42 |     3   (0)| 00:00:01
|     |       |
|* 17 |             BITMAP INDEX RANGE SCAN   | SALES_CHANNEL_BIX          |       |       |            |
|KEY(SQ)|KEY(SQ)|
|  18 |           BITMAP MERGE                |                            |       |       |            |
|     |       |
|  19 |            BITMAP KEY ITERATION       |                            |       |       |            |
|     |       |
|  20 |             BUFFER SORT               |                            |       |       |            |
|     |       |
|* 21 |              TABLE ACCESS FULL        | TIMES                      |   365 |  5840 |    18   (0)| 00:00:01
|     |       |
|* 22 |             BITMAP INDEX RANGE SCAN   | SALES_TIME_BIX             |       |       |            |
|KEY(SQ)|KEY(SQ)|
|  23 |           BITMAP MERGE                |                            |       |       |            |
|     |       |
|  24 |            BITMAP KEY ITERATION       |                            |       |       |            |
|     |       |
|  25 |             BUFFER SORT               |                            |       |       |            |
|     |       |
|  26 |              TABLE ACCESS FULL        | SYS_TEMP_0FD9D660F_8947E |     1 |    13 |     2   (0)| 00:00:01
|     |       |
|* 27 |             BITMAP INDEX RANGE SCAN   | SALES_CUST_BIX             |       |       |            |
|KEY(SQ)|KEY(SQ)|
|* 28 |      TABLE ACCESS FULL                | TIMES                      |   365 |  5840 |    18   (0)| 00:00:01
|     |       |
|* 29 |     TABLE ACCESS FULL                 | CHANNELS                   |     2 |    42 |     3   (0)| 00:00:01
|     |       |
----------------------------------------------------------------------------------------------------------------------
------------------

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

   3 - filter("C"."CUST_STATE_PROVINCE"='CA')
   5 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
   6 - access("S"."TIME_ID"="T"."TIME_ID")
   7 - access("S"."CUST_ID"="C0")
  16 - filter("CH"."CHANNEL_DESC"='Catalog' OR "CH"."CHANNEL_DESC"='Internet')
  17 - access("S"."CHANNEL_ID"="CH"."CHANNEL_ID")
  21 - 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')
  22 - access("S"."TIME_ID"="T"."TIME_ID")
  27 - access("S"."CUST_ID"="C0")
  28 - 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')
  29 - filter("CH"."CHANNEL_DESC"='Catalog' OR "CH"."CHANNEL_DESC"='Internet')

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


Statistics
----------------------------------------------------------
      17911  recursive calls
         19  db block gets
      35577  consistent gets
       2157  physical reads
       1616  redo size
       1896  bytes sent via SQL*Net to client
        442  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
```

Practice 6-1: Star Schema Tuning (continued)

```
        155  sorts (memory)
          0  sorts (disk)
         41  rows processed

SQL>
SQL>


--------------------------------------------------------------

set echo on

alter system flush shared_pool;
alter system flush buffer_cache;

set pagesize 200
set linesize 250
set timing on
set autotrace on

ALTER SESSION SET star_transformation_enabled=TRUE;

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;
```

5) How do you eliminate one table 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_bji.sql
SQL>
SQL> CREATE BITMAP INDEX sales_c_state_bjix ON
sales(customers.cust_state_province) FROM sales, customers WHERE
sales.cust_id=customers.cust_id
  2  LOCAL NOLOGGING COMPUTE STATISTICS;
CREATE BITMAP INDEX sales_c_state_bjix ON
sales(customers.cust_state_province) FROM sales, customers WHERE
sales.cust_id=customers.cust_id
```

Practice 6-1: Star Schema Tuning (continued)

```
*
ERROR at line 1:
ORA-25954: missing primary key or unique constraint on dimension


SQL>

--------------------------------------------------------------

set echo on

CREATE BITMAP INDEX sales_c_state_bjix ON
sales(customers.cust_state_province) FROM sales, customers WHERE
sales.cust_id=customers.cust_id
LOCAL NOLOGGING COMPUTE STATISTICS;
```

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

   a) You need to `ENABLE VALIDATE` the `CUSTOMERS_PK` constraint before you can create the bitmap join index.

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

Table altered.

SQL>
SQL> CREATE BITMAP INDEX sales_c_state_bjix ON
sales(customers.cust_state_province) FROM sales, customers WHERE
sales.cust_id=customers.cust_id
  2  LOCAL NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL>

--------------------------------------------------------------

set echo on

alter table customers enable constraint customers_pk;

CREATE BITMAP INDEX sales_c_state_bjix ON
sales(customers.cust_state_province) FROM sales, customers WHERE
sales.cust_id=customers.cust_id
LOCAL NOLOGGING COMPUTE STATISTICS;
```

8) Verify that you solved the problem from step 5.

```
SQL> @fourth_run
```

Practice 6-1: Star Schema Tuning (continued)

```
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
…
Indirect             Montara                        1999-02
1618.01
Indirect             Quartzhill                     1999-02
412.83

41 rows selected.

Elapsed: 00:00:00.23

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

-------------------------------------------------------------------------------------------------------------------
-----------
```

Practice 6-1: Star Schema Tuning (continued)

```
| Id  | Operation                          | Name                | Rows  | Bytes | Cost (%CPU)| Time     |
Pstart| Pstop |
---------------------------------------------------------------------------------------------------------------
-----------
|   0 | SELECT STATEMENT                   |                     |   498 | 41832 |   557   (2)| 00:00:07 |
|       |
|   1 |  HASH GROUP BY                     |                     |   498 | 41832 |   557   (2)| 00:00:07 |
|       |
|*  2 |   HASH JOIN                        |                     |   498 | 41832 |   556   (2)| 00:00:07 |
|       |
|*  3 |    TABLE ACCESS FULL               | CHANNELS            |     2 |    42 |     3   (0)| 00:00:01 |
|       |
|*  4 |    HASH JOIN                       |                     |   996 | 62748 |   552   (1)| 00:00:07 |
|       |
|*  5 |     TABLE ACCESS FULL              | TIMES               |   365 |  5840 |    18   (0)| 00:00:01 |
|       |
|*  6 |     HASH JOIN                      |                     |  3984 |   182K|   533   (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 |    42 |     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
----------------------------------------------------------
      18151  recursive calls
          0  db block gets
       8759  consistent gets
       2002  physical reads
          0  redo size
       1888  bytes sent via SQL*Net to client
        442  bytes received via SQL*Net from client
          4  SQL*Net roundtrips to/from client
        154  sorts (memory)
          0  sorts (disk)
```

Practice 6-1: Star Schema Tuning (continued)

```
         41  rows processed

SQL>

--------------------------------------------------------------

set echo on

ALTER SESSION SET star_transformation_enabled=TRUE;

alter system flush shared_pool;
alter system flush buffer_cache;

set pagesize 200
set linesize 250
set timing on
set autotrace on

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;
```

9) Determine how the system could dynamically determine which `SALES` partitions to access for the previous query.

   a) View the `OTHER` column of `PLAN_TABLE` for the `PARTITION RANGE` operation.

```
SQL> @dynamic_partition_pruning
SQL>
SQL> set autotrace off
SQL> set timing off
SQL>
SQL> set long 4000
SQL>
SQL> alter session set star_transformation_enabled=true;

Session altered.

SQL>
SQL> explain plan for
  2  SELECT ch.channel_class, c.cust_city, t.calendar_quarter_desc,
  3     SUM(s.amount_sold) sales_amount
  4  FROM sales s, times t, customers c, channels ch
  5  WHERE s.time_id = t.time_id
  6  AND   s.cust_id = c.cust_id
  7  AND   s.channel_id = ch.channel_id
  8  AND   c.cust_state_province = 'CA'
```

Practice 6-1: Star Schema Tuning (continued)

```
  9  AND   ch.channel_desc in ('Internet','Catalog')
 10  AND   t.calendar_quarter_desc IN ('1999-01','1999-02','2000-
03','2000-04')
 11  GROUP BY ch.channel_class, c.cust_city,
t.calendar_quarter_desc;

Explained.

SQL>
SQL> SELECT other
  2  FROM plan_table
  3  WHERE operation='PARTITION RANGE';

OTHER
--------------------------------------------------------------------
------------
SELECT distinct TBL$OR$IDX$PART$NUM("SALES", 0, d#, p#,
PAP_ALIAS_0."PAP_COL_ALI
AS_0") FROM (SELECT /*+ SEMIJOIN_DRIVER */ "T"."TIME_ID"
"PAP_COL_ALIAS_0" FROM
"TIMES" "T" WHERE "T"."CALENDAR_QUARTER_DESC"='1999-01' OR
"T"."CALENDAR_QUARTER
_DESC"='1999-02' OR "T"."CALENDAR_QUARTER_DESC"='2000-03' OR
"T"."CALENDAR_QUART
ER_DESC"='2000-04') PAP_ALIAS_0 ORDER BY 1


SQL>


--------------------------------------------------------------

set echo on

set autotrace off
set timing off

set long 4000

alter session set star_transformation_enabled=true;

explain plan for
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;

SELECT other
FROM plan_table
WHERE operation='PARTITION RANGE';
```

Practice 6-1: Star Schema Tuning (continued)

10) Clean up your environment by removing the index you created and returning the constraint to its original state.

```
SQL> @cleanup_star_schema_lab
SQL> set echo on
SQL>
SQL> set timing off
SQL> set autotrace off
SQL>
SQL> drop index sales_c_state_bjix;

Index dropped.

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

Table altered.

SQL>
SQL> connect / as sysdba
Connected.
SQL>
SQL> revoke dba from sh;

Revoke succeeded.

SQL>
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Star_Schema_Tuning]$

--------------------------------------------------------------

set echo on

set timing off
set autotrace off

drop index sales_c_state_bjix;

alter table customers enable novalidate constraint customers_pk;

connect / as sysdba

revoke dba from sh;

exit;
```

Practices for Lesson 7

Practice 7-1: Using System Statistics

In this practice, you manipulate system statistics and show that they are important for the optimizer to select the correct execution plans.

1. Connected as the `oracle` user in a terminal session, execute the `sysstats_setup.sh` script located in your `$HOME/solutions/System_Stats` directory. This script creates a user called `JFV` and some tables used throughout this lab. The script also makes sure that object statistics are correctly gathered.

```
[oracle@edrsr33p1-orcl ~]$ cd $HOME/solutions/System_Stats
[oracle@edrsr33p1-orcl System_Stats]$
[oracle@edrsr33p1-orcl System_Stats]$ ./sysstats_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Mon Mar 31 19:11:42
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> connect / as sysdba;
Connected.
SQL>
SQL> drop user jfv cascade;

User dropped.

SQL>
SQL> create user jfv identified by jfv default tablespace users
temporary tablespace temp;

User created.

SQL>
SQL> grant connect, resource, dba to jfv;

Grant succeeded.

SQL>
SQL>
SQL> connect jfv/jfv
Connected.
SQL>
SQL> drop table t purge;
drop table t purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist
```

Practice 7-1: Using System Statistics (continued)

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


SQL>
SQL> create table t(c number);

Table created.

SQL>
SQL> insert into t values (1);

1 row created.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL> insert into t select * from t;

1 row created.

SQL>
SQL> /

2 rows created.

SQL> /

4 rows created.

SQL> /

8 rows created.

SQL> /

16 rows created.

SQL> /

32 rows created.

SQL> /

64 rows created.

SQL> /
```

Practice 7-1: Using System Statistics (continued)

```
128 rows created.

SQL> /

256 rows created.

SQL> /

512 rows created.

SQL> /

1024 rows created.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL> insert into t select * from t;

2048 rows created.

SQL>
SQL> /

4096 rows created.

SQL> /

8192 rows created.

SQL> /

16384 rows created.

SQL> /

32768 rows created.

SQL> /

65536 rows created.

SQL> /

131072 rows created.

SQL> /

262144 rows created.

SQL>
SQL> commit;

Commit complete.
```

```
SQL>
SQL> create table z(d number);

Table created.

SQL>
SQL> begin
  2   for i in 1..100 loop
  3      insert into z values (i);
  4   end loop;
  5   commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL> create unique index iz on z(d);

Index created.

SQL>
SQL> execute dbms_stats.gather_table_stats('JFV','T',cascade=>true);

PL/SQL procedure successfully completed.

SQL>
SQL> execute dbms_stats.gather_table_stats('JFV','Z',cascade=>true);

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL>
SQL> connect / as sysdba;
Connected.
SQL>
SQL> alter system flush shared_pool;

System altered.

SQL>
SQL> alter system flush buffer_cache;

System altered.

SQL>
SQL> execute dbms_stats.delete_system_stats;

PL/SQL procedure successfully completed.

SQL>
SQL> execute DBMS_STATS.SET_SYSTEM_STATS (pname => 'cpuspeednw',
pvalue => 0);

PL/SQL procedure successfully completed.
```

Practice 7-1: Using System Statistics (continued)

```
SQL>
SQL> select sname,pname,pval1 from aux_stats$;

SNAME                          PNAME
PVAL1
------------------------------ ------------------------------ ------
----
SYSSTATS_INFO                  STATUS
SYSSTATS_INFO                  DSTART
SYSSTATS_INFO                  DSTOP
SYSSTATS_INFO                  FLAGS
1
SYSSTATS_MAIN                  CPUSPEEDNW
0
SYSSTATS_MAIN                  IOSEEKTIM
10
SYSSTATS_MAIN                  IOTFRSPEED
4096
SYSSTATS_MAIN                  SREADTIM
SYSSTATS_MAIN                  MREADTIM
SYSSTATS_MAIN                  CPUSPEED
SYSSTATS_MAIN                  MBRC

SNAME                          PNAME
PVAL1
------------------------------ ------------------------------ ------
----
SYSSTATS_MAIN                  MAXTHR
SYSSTATS_MAIN                  SLAVETHR

13 rows selected.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl System_Stats]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/System_Stats

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @sysstats_setup.sql
```

```
[oracle@edrsr33p1-orcl System_Stats]$
--------------------------------------------------------------

set echo on

connect / as sysdba;

drop user jfv cascade;

create user jfv identified by jfv default tablespace users temporary
tablespace temp;

grant connect, resource, dba to jfv;


connect jfv/jfv

drop table t purge;

drop table z purge;

create table t(c number);

insert into t values (1);

commit;

insert into t select * from t;

/
/
/
/
/
/
/
/
/
/

commit;

insert into t select * from t;

/
/
/
/
/
/
/

commit;

create table z(d number);
```

Practice 7-1: Using System Statistics (continued)

```
begin
 for i in 1..100 loop
    insert into z values (i);
 end loop;
 commit;
end;
/

create unique index iz on z(d);

execute dbms_stats.gather_table_stats('JFV','T',cascade=>true);

execute dbms_stats.gather_table_stats('JFV','Z',cascade=>true);



connect / as sysdba;

alter system flush shared_pool;

alter system flush buffer_cache;

execute dbms_stats.delete_system_stats;

execute DBMS_STATS.SET_SYSTEM_STATS (pname => 'cpuspeednw', pvalue
=> 0);

select sname,pname,pval1 from aux_stats$;

exit;
```

2. From your terminal session, connect as the `JFV` user in the SQL*Plus session. After this, execute the following statement and determine how long it takes to execute:
   ```
   select /* Without system stats */ count(*)
   from t,z
   where t.c=z.d;
   ```

```
[oracle@edrsr33p1-orcl System_Stats]$ sqlplus jfv/jfv

SQL*Plus: Release 11.1.0.6.0 - Production on Mon Mar 31 19:11:57
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> @select_without_sysstats
SQL>
SQL> set timing on
SQL>
```

Practice 7-1: Using System Statistics (continued)

```
SQL> select /* Without system stats */ count(*)
  2  from t,z
  3  where t.c=z.d;

  COUNT(*)
----------
    524288

Elapsed: 00:00:00.24
SQL>
SQL> set timing off
SQL>
SQL>

--------------------------------------------------------------
set echo on

set timing on

select /* Without system stats */ count(*)
from t,z
where t.c=z.d;

set timing off
```

3. Determine the execution plan used to execute the previous statement. In addition, determine the optimizer's cost, CPU cost, and I/O cost for the previous execution. What do you observe?

a) The optimizer does not use CPU costing. This is because system statistics were deleted during the first step of this lab. The plan chosen by the optimizer might not be the best one.

```
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from table(dbms_xplan.display_cursor);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
------------
SQL_ID  6avdu58tamzju, child number 0
-------------------------------------
select /* Without system stats */ count(*) from t,z where t.c=z.d

Plan hash value: 3698032250

------------------------------------------------------------
| Id  | Operation           | Name | Rows  | Bytes | Cost  |
------------------------------------------------------------
|   0 | SELECT STATEMENT    |      |       |       |   134 |
|   1 |  SORT AGGREGATE     |      |     1 |     6 |       |

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
------------
```

Practice 7-1: Using System Statistics (continued)

```
|   2 |   NESTED LOOPS       |      |   524K|  3072K|   134 |
|   3 |    TABLE ACCESS FULL| T    |   524K|  1536K|   134 |
|*  4 |    INDEX UNIQUE SCAN| IZ   |     1 |     3 |       |
------------------------------------------------------------

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

   4 - access("T"."C"="Z"."D")

Note

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
------------
-----
   - cpu costing is off (consider enabling it)


25 rows selected.

SQL>
SQL> col operations     format a20
SQL> col object_name format a11
SQL> col options        format a15
SQL> col cost_cpu_io format a30
SQL>
SQL>
SQL> select operation operations, object_name, options,
  2         cost||' -- '||cpu_cost||' -- '||io_cost cost_cpu_io
  3  from (select * from v$sql_plan where address in (select address
  4                                                    from v$sql
  5                                                    where sql_text
like '%system stats%' and
  6                                                          sql_text
not like '%connect%'));

OPERATIONS           OBJECT_NAME OPTIONS         COST_CPU_IO
-------------------- ----------- --------------- -------------------
-----------
SELECT STATEMENT                                 134 --  --
SORT                             AGGREGATE        --  --
NESTED LOOPS                                     134 --  -- 134
TABLE ACCESS         T           FULL            134 --  -- 134
INDEX                IZ          UNIQUE SCAN      --  --

SQL>

---------------------------------------------------------------

set echo on

select * from table(dbms_xplan.display_cursor);

col operations  format a20
col object_name format a11
col options     format a15
```

Practice 7-1: Using System Statistics (continued)

```
col cost_cpu_io format a30


select operation operations, object_name, options,
       cost||' -- '||cpu_cost||' -- '||io_cost cost_cpu_io
from (select * from v$sql_plan where address in (select address
                                                 from v$sql
                                                 where sql_text like
'%system stats%' and
                                                       sql_text not
like '%connect%'));
```

4. How can you ensure that the optimizer finds a better plan during future executions of the same statement? Implement your solution.

b) Gather system statistics again. Because you do not have yet a real workload, you can gather system statistics in NOWORKLOAD mode.

```
SQL> connect / as sysdba;
Connected.
SQL> @gather_system_stats
SQL> set echo on
SQL>
SQL> execute DBMS_STATS.GATHER_SYSTEM_STATS(gathering_mode =>
'NOWORKLOAD');

PL/SQL procedure successfully completed.

SQL>
SQL> select sname,pname,pval1 from aux_stats$;

SNAME                          PNAME
PVAL1
------------------------------ ------------------------------ ------
----
SYSSTATS_INFO                  STATUS
SYSSTATS_INFO                  DSTART
SYSSTATS_INFO                  DSTOP
SYSSTATS_INFO                  FLAGS
1
SYSSTATS_MAIN                  CPUSPEEDNW
1893.021
SYSSTATS_MAIN                  IOSEEKTIM
8.043
SYSSTATS_MAIN                  IOTFRSPEED
4096
SYSSTATS_MAIN                  SREADTIM
SYSSTATS_MAIN                  MREADTIM
SYSSTATS_MAIN                  CPUSPEED
SYSSTATS_MAIN                  MBRC

SNAME                          PNAME
PVAL1
------------------------------ ------------------------------ ------
----
SYSSTATS_MAIN                  MAXTHR
```

Practice 7-1: Using System Statistics (continued)

```
SYSSTATS_MAIN                      SLAVETHR

13 rows selected.

SQL>

--------------------------------------------------------------

set echo on

execute DBMS_STATS.GATHER_SYSTEM_STATS(gathering_mode =>
'NOWORKLOAD');

select sname,pname,pval1 from aux_stats$;
```

5. Before verifying your solution, you should flush the System Global Area (SGA) pools, such as the shared pool and the buffer cache. This is done to prevent interferences from the previous steps.

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

System altered.

SQL>
SQL> alter system flush buffer_cache;

System altered.

SQL>
SQL>

--------------------------------------------------------------

set echo on

alter system flush shared_pool;

alter system flush buffer_cache;
```

6. Connected as the `JFV` user again, check your solution against the following query:
```
select /* With system stats */ count(*)
from t,z
where t.c=z.d;
```

What do you observe?

c) The optimizer can make a better decision because it was able to use meaningful system statistics. You can see that the execution time is now half the value it was previously, and the execution plan now includes CPU costing information.

Practice 7-1: Using System Statistics (continued)

```
SQL> connect jfv/jfv
Connected.
SQL> @select_with_sysstats
SQL> set timing on
SQL>
SQL> select /* With system stats */ count(*)
  2  from t,z
  3  where t.c=z.d;

  COUNT(*)
----------
    524288

Elapsed: 00:00:00.11
SQL>
SQL> set timing off
SQL>
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from table(dbms_xplan.display_cursor);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
------------
SQL_ID  2x55txn3742by, child number 0
-------------------------------------
select /* With system stats */ count(*) from t,z where t.c=z.d

Plan hash value: 2407521827

--------------------------------------------------------------------
--------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
--------------------------------------------------------------------
--------
|   0 | SELECT STATEMENT    |      |       |       |   272 (100)|
|
|   1 |  SORT AGGREGATE     |      |     1 |     6 |            |
|

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
------------
|*  2 |   HASH JOIN         |      |   524K|  3072K|   272   (3)|
00:00:03 |
|   3 |    INDEX FULL SCAN  | IZ   |   100 |   300 |     1   (0)|
00:00:01 |
|   4 |    TABLE ACCESS FULL| T    |   524K|  1536K|   267   (2)|
00:00:03 |
--------------------------------------------------------------------
--------

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

```
   2 - access("T"."C"="Z"."D")


21 rows selected.

SQL>
SQL> col operations     format a20
SQL> col object_name format a11
SQL> col options        format a15
SQL> col cost_cpu_io format a30
SQL>
SQL>
SQL> select operation operations, object_name, options,
  2         cost||' -- '||cpu_cost||' -- '||io_cost cost_cpu_io
  3  from (select * from v$sql_plan where address in (select address
  4                                                   from v$sql
  5                                                   where sql_text
like '%system stats%' and
  6                                                         sql_text
not like '%connect%'));

OPERATIONS           OBJECT_NAME OPTIONS         COST_CPU_IO
-------------------- ----------- --------------- -------------------
-----------
SELECT STATEMENT                                 272 --  --
SORT                             AGGREGATE        --  --
HASH JOIN                                        272 -- 146844065 --
264
INDEX                IZ          FULL SCAN       1 -- 27121 -- 1
TABLE ACCESS         T           FULL            267 -- 84867339 --
263

SQL>

--------------------------------------------------------------

set timing on

select /* With system stats */ count(*)
from t,z
where t.c=z.d;

set timing off

--------------------------------------------------------------

set echo on

select * from table(dbms_xplan.display_cursor);

col operations  format a20
col object_name format a11
col options     format a15
col cost_cpu_io format a30


select operation operations, object_name, options,
```

Practice 7-1: Using System Statistics (continued)

```
       cost||' -- '||cpu_cost||' -- '||io_cost cost_cpu_io
from (select * from v$sql_plan where address in (select address
                                                 from v$sql
                                                 where sql_text like
'%system stats%' and
                                                       sql_text not
like '%connect%'));
```

7. Exit from your SQL*Plus session and clean up your environment for this lab by executing the `systats_cleanup.sh` script.

```
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl System_Stats]$ ./sysstats_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Mon Mar 31 19:13:54
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user jfv cascade;

User dropped.

SQL>
SQL> alter system flush shared_pool;

System altered.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl System_Stats]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/System_Stats

export ORACLE_SID=orcl
```

Practice 7-1: Using System Statistics (continued)

```
export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @sysstats_cleanup.sql

--------------------------------------------------------------

set echo on

drop user jfv cascade;

alter system flush shared_pool;

exit;
```

Practice 7-2: Automatic Statistics Gathering

In this practice, you manipulate object statistics to see their incidences on the execution plans of your SQL statements. **Note:** All scripts needed for this lab can be found in your `$HOME/solutions/Automatic_Gather_Stats` directory.

1) From a terminal window connected as the `oracle` user, set up your environment for this lab by executing the `ags_setup.sh` script. This script creates a user called `AGS` that you use throughout this lab. The script also creates a table called `EMP` and an index.

```
[oracle@edrsr33p1-orcl Automatic_Gather_Stats]$ ./ags_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Mon Apr 7 15:55:28 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user ags cascade;
drop user ags cascade
          *
ERROR at line 1:
ORA-01918: user 'AGS' does not exist


SQL>
SQL> create user ags identified by ags;

User created.

SQL>
SQL> grant dba to ags;

Grant succeeded.

SQL>
SQL> alter system flush shared_pool;

System altered.

SQL>
SQL> --
SQL> -- Turn off AUTOTASK
SQL> --
SQL>
SQL> alter system set "_enable_automatic_maintenance"=0;

System altered.
```

Practice 7-2: Automatic Statistics Gathering (continued)

```
SQL>
SQL> connect ags/ags
Connected.
SQL>
SQL> drop table emp purge;
drop table emp purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
SQL> create table emp
  2  (
  3   empno      number,
  4   ename      varchar2(20),
  5   phone      varchar2(20),
  6   deptno  number
  7  );

Table created.

SQL>
SQL>
SQL> insert into emp
  2    with tdata as
  3         (select rownum empno
  4           from all_objects
  5           where rownum <= 1000)
  6    select rownum,
  7           dbms_random.string ('u', 20),
  8           dbms_random.string ('u', 20),
  9           case
 10                  when rownum/100000 <= 0.001 then mod(rownum,
10)
 11                  else 10
 12           end
 13      from tdata a, tdata b
 14     where rownum <= 100000;

100000 rows created.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL> create index emp_i1 on emp(deptno);

Index created.

SQL>
SQL> exec dbms_stats.delete_schema_stats('AGS');

PL/SQL procedure successfully completed.
```

```
SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Automatic_Gather_Stats]$

-------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Automatic_Gather_Stats

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin

sqlplus / as sysdba @ags_setup.sql

-------------------------------------------------------------

set echo on

drop user ags cascade;

create user ags identified by ags;

grant dba to ags;

alter system flush shared_pool;

--
-- Turn off AUTOTASK
--

alter system set "_enable_automatic_maintenance"=0;

connect ags/ags

drop table emp purge;

create table emp
(
 empno   number,
 ename   varchar2(20),
 phone   varchar2(20),
 deptno  number
);


insert into emp
  with tdata as
```

```
        (select rownum empno
          from all_objects
          where rownum <= 1000)
    select rownum,
          dbms_random.string ('u', 20),
          dbms_random.string ('u', 20),
          case
                 when rownum/100000 <= 0.001 then mod(rownum, 10)
                 else 10
          end
     from tdata a, tdata b
    where rownum <= 100000;

commit;

create index emp_i1 on emp(deptno);

exec dbms_stats.delete_schema_stats('AGS');

exit;
```

2) Connect as the AGS user under a SQL*Plus session.

```
[oracle@edrsr33p1-orcl Automatic_Gather_Stats]$ sqlplus ags/ags

SQL*Plus: Release 11.1.0.6.0 - Production on Mon Apr 7 15:55:47 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
```

3) Determine the distribution of the deptno values found in the EMP table. What do you observe?

   a) You can see that there are 11 different department values, each repeating 0.01% of the time except value 10 that repeats 99.9% of the time.

```
SQL> @show_deptno_distribution
SQL>
SQL> select deptno, count(*) cnt_per_deptno, (count(*)*100)/nr
deptno_percent
  2  from emp, (select max(empno) nr
  3             from emp)
  4  group by deptno, nr
  5  order by deptno;

    DEPTNO CNT_PER_DEPTNO DEPTNO_PERCENT
---------- -------------- --------------
         0             10            .01
```

```
        1           10          .01
        2           10          .01
        3           10          .01
        4           10          .01
        5           10          .01
        6           10          .01
        7           10          .01
        8           10          .01
        9           10          .01
       10        99900         99.9

11 rows selected.

SQL>

--------------------------------------------------------------

set echo on

select deptno, count(*) cnt_per_deptno, (count(*)*100)/nr
deptno_percent
from emp, (select max(empno) nr
          from emp)
group by deptno, nr
order by deptno;
```

4) Determine if there are histograms available on any column of the `EMP` table. What do you observe?

   a) Currently, there are no histograms defined on any column of the `EMP` table.

```
SQL> @check_emp_histogram
SQL> set echo on
SQL>
SQL> select column_name, histogram, num_buckets
  2  from user_tab_columns
  3  where table_name='EMP';

COLUMN_NAME                    HISTOGRAM       NUM_BUCKETS
------------------------------ --------------- -----------
EMPNO                          NONE
ENAME                          NONE
PHONE                          NONE
DEPTNO                         NONE

SQL>

--------------------------------------------------------------

set echo on

select column_name, histogram, num_buckets
from user_tab_columns
where table_name='EMP';
```

Practice 7-2: Automatic Statistics Gathering (continued)

5) Determine if you currently have statistics gathered on your EMP table. What do you observe?

   a) Currently, there are no statistics gathered on your EMP table.

```
SQL> @check_table_last_analyzed
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT
last_analyzed,sample_size,monitoring,num_rows,blocks,table_name
  2  FROM user_tables
  3  WHERE table_name = 'EMP';

LAST_ANAL SAMPLE_SIZE MON   NUM_ROWS     BLOCKS TABLE_NAME
--------- ----------- --- ---------- ---------- --------------------
----------
                      YES                       EMP

SQL>

--------------------------------------------------------------

set echo on

set linesize 200 pagesize 1000

SELECT
last_analyzed,sample_size,monitoring,num_rows,blocks,table_name
FROM user_tables
WHERE table_name = 'EMP';
```

6) Determine if you currently have statistics gathered on the index created on top of the EMP table. What do you observe?

   a) Currently, EMP_I1 has no statistics gathered.

```
SQL> @check_index_last_analyzed
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT index_name name,num_rows,
  2         last_analyzed,distinct_keys,leaf_blocks,
  3         avg_leaf_blocks_per_key,join_index
  4  FROM user_indexes
  5  WHERE table_name = 'EMP';

NAME                              NUM_ROWS LAST_ANAL DISTINCT_KEYS
LEAF_BLOCKS AVG_LEAF_BLOCKS_PER_KEY JOI
------------------------------ ---------- --------- ------------- --
--------- ----------------------- ---
EMP_I1
NO
```

```
SQL>
--------------------------------------------------------------

set echo on

set linesize 200 pagesize 1000

SELECT index_name name,num_rows,
       last_analyzed,distinct_keys,leaf_blocks,
       avg_leaf_blocks_per_key,join_index
FROM user_indexes
WHERE table_name = 'EMP';
```

7) Execute the following two statements and determine their execution plans:
`select count(*), max(empno) from emp where deptno = 9;`
`select count(*), max(empno) from emp where deptno = 10;`

What do you observe and why?

a) You see that for literal 9, the optimizer decided to use the index whereas for literal 10, the optimizer decided to do `TABLE ACCESS FULL`. The optimizer determined the correct plans in both cases. This is because dynamic sampling was used, as there were no statistics gathered on both objects.

```
SQL> @check_exec_plans
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> set autotrace on
SQL>
SQL> select count(*), max(empno) from emp where deptno = 9;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99


Execution Plan
----------------------------------------------------------
Plan hash value: 3184478295

--------------------------------------------------------------------
-------------------
| Id  | Operation                    | Name   | Rows   | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT             |        |     1 |    26 |
2   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |        |     1 |    26 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP    |    10 |   260 |
2   (0)| 00:00:01 |
```

```
|*  3 |     INDEX RANGE SCAN          | EMP_I1 |    10 |        |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------

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

   3 - access("DEPTNO"=9)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          9  recursive calls
          0  db block gets
         64  consistent gets
          1  physical reads
          0  redo size
        481  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
SQL> select count(*), max(empno) from emp where deptno = 10;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000


Execution Plan
----------------------------------------------------------
Plan hash value: 2083865914

--------------------------------------------------------------------
-------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
--------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT   |      |     1 |    26 |   265   (1)|
00:00:03 |
|   1 |  SORT AGGREGATE    |      |     1 |    26 |            |
|
|*  2 |   TABLE ACCESS FULL| EMP  | 86262 |  2190K|   265   (1)|
00:00:03 |
--------------------------------------------------------------------
-------

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

```
---------------------------------------------------

   2 - filter("DEPTNO"=10)

Note
-----
   - dynamic sampling used for this statement


Statistics
----------------------------------------------------------
          7  recursive calls
          0  db block gets
        871  consistent gets
          5  physical reads
          0  redo size
        482  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
SQL> set autotrace off
SQL>

--------------------------------------------------------------

set echo on

set linesize 200 pagesize 1000

set autotrace on

select count(*), max(empno) from emp where deptno = 9;

select count(*), max(empno) from emp where deptno = 10;

set autotrace off
```

8) Confirm your assumption from the previous step.

```
SQL> show parameter optimizer_dynamic_sampling

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------------------
-----------
optimizer_dynamic_sampling           integer     2
SQL>
```

9) Make sure you disable the mechanism found in the previous step.

```
SQL> alter session set optimizer_dynamic_sampling=0;

Session altered.
SQL>
```

Practice 7-2: Automatic Statistics Gathering (continued)

10) Perform step 7 again. What do you observe and why?

    a) Because dynamic sampling is not used, the optimizer cannot use any statistics. It uses default statistics that are not a good choice for the second statement.

```
SQL> @check_exec_plans
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> set autotrace on
SQL>
SQL> select count(*), max(empno) from emp where deptno = 9;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99


Execution Plan
----------------------------------------------------------
Plan hash value: 3184478295

--------------------------------------------------------------------
-------------------
| Id  | Operation                   | Name   | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT            |        |     1 |    26 |
5   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE             |        |     1 |    26 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP    |   714 | 18564 |
5   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN         | EMP_I1 |   286 |       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------

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

   3 - access("DEPTNO"=9)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
          3  consistent gets
          0  physical reads
          0  redo size
        481  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
```

```
          0  sorts (disk)
          1  rows processed

SQL>
SQL> select count(*), max(empno) from emp where deptno = 10;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000


Execution Plan
----------------------------------------------------------
Plan hash value: 3184478295

---------------------------------------------------------------------
-------------------
| Id  | Operation                    | Name   | Rows  | Bytes | Cost
(%CPU)| Time     |
---------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT             |        |     1 |    26 |
5   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |        |     1 |    26 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP    |   714 | 18564 |
5   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | EMP_I1 |   286 |       |
1   (0)| 00:00:01 |
---------------------------------------------------------------------
-------------------

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

   3 - access("DEPTNO"=10)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        954  consistent gets
        190  physical reads
          0  redo size
        482  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
SQL> set autotrace off
SQL>

--------------------------------------------------------------
```

Practice 7-2: Automatic Statistics Gathering (continued)

```
set echo on

set linesize 200 pagesize 1000

set autotrace on

select count(*), max(empno) from emp where deptno = 9;

select count(*), max(empno) from emp where deptno = 10;

set autotrace off
```

11) Reset dynamic sampling as it was at the beginning of this lab.

```
SQL> alter session set optimizer_dynamic_sampling=2;

Session altered.

SQL>
```

12) Set the following wrong statistic values for your EMP table:
- Set its number of rows to 10.
- Set its number of blocks to 5.

```
SQL> @set_wrong_stats
SQL> set echo on
SQL>
SQL> exec
dbms_stats.set_table_stats('AGS','EMP',null,null,null,10,5);

PL/SQL procedure successfully completed.

SQL>

--------------------------------------------------------------

set echo on

exec dbms_stats.set_table_stats('AGS','EMP',null,null,null,10,5);
```

13) Check that you modified correctly the statistics of the EMP table.

```
SQL> @check_table_last_analyzed
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT
last_analyzed,sample_size,monitoring,num_rows,blocks,table_name
  2  FROM user_tables
  3  WHERE table_name = 'EMP';

LAST_ANAL SAMPLE_SIZE MON   NUM_ROWS     BLOCKS TABLE_NAME
--------- ----------- --- ---------- ---------- --------------------
----------
```

```
07-APR-08        2000 YES         10          5 EMP

SQL>

--------------------------------------------------------------

set echo on

set linesize 200 pagesize 1000

SELECT
last_analyzed,sample_size,monitoring,num_rows,blocks,table_name
FROM user_tables
WHERE table_name = 'EMP';
```

14) Perform step 7 again. What do you observe and why?

a) Because there are statistics defined on the `EMP` table, the optimizer uses them, and not dynamic sampling. However, because the statistics are incorrect, the generated plans are also incorrect, at least for the second statement.

```
SQL> @check_exec_plans
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> set autotrace on
SQL>
SQL> select count(*), max(empno) from emp where deptno = 9;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99


Execution Plan
----------------------------------------------------------
Plan hash value: 3184478295

--------------------------------------------------------------------
-------------------
| Id  | Operation                   | Name    | Rows   | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT            |         |      1 |    26 |
2   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE             |         |      1 |    26 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP     |      1 |    26 |
2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN         | EMP_I1 |      1 |       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------
```

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

   3 - access("DEPTNO"=9)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          3  consistent gets
          0  physical reads
          0  redo size
        481  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
SQL> select count(*), max(empno) from emp where deptno = 10;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000


Execution Plan
----------------------------------------------------------
Plan hash value: 3184478295

--------------------------------------------------------------------
-------------------
| Id  | Operation                    | Name   | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT             |        |     1 |    26 |
2   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |        |     1 |    26 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP    |     1 |    26 |
2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | EMP_I1 |     1 |       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------

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

   3 - access("DEPTNO"=10)


Statistics
----------------------------------------------------------
```

```
          0  recursive calls
          0  db block gets
        805  consistent gets
          0  physical reads
          0  redo size
        482  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
SQL> set autotrace off
SQL>

--------------------------------------------------------------

set echo on

set linesize 200 pagesize 1000

set autotrace on

select count(*), max(empno) from emp where deptno = 9;

select count(*), max(empno) from emp where deptno = 10;

set autotrace off
```

15) Make sure you manually gather statistics on your `EMP` table and its corresponding index.

```
SQL> @gather_stats_manually
SQL> set echo on
SQL>
SQL> exec dbms_stats.gather_table_stats('AGS', 'EMP', cascade =>
true);

PL/SQL procedure successfully completed.

SQL>

--------------------------------------------------------------

set echo on

exec dbms_stats.gather_table_stats('AGS', 'EMP', cascade => true);
```

16) Make sure statistics were gathered on your objects.

```
SQL> @check_table_last_analyzed
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
```

Practice 7-2: Automatic Statistics Gathering (continued)

```
SQL>
SQL> SELECT
last_analyzed,sample_size,monitoring,num_rows,blocks,table_name
  2  FROM user_tables
  3  WHERE table_name = 'EMP';

LAST_ANAL SAMPLE_SIZE MON   NUM_ROWS     BLOCKS TABLE_NAME
--------- ----------- --- ---------- ---------- --------------------
----------
07-APR-08      100000 YES     100000        874 EMP

SQL>
SQL> @check_index_last_analyzed
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT index_name name,num_rows,
  2         last_analyzed,distinct_keys,leaf_blocks,
  3         avg_leaf_blocks_per_key,join_index
  4  FROM user_indexes
  5  WHERE table_name = 'EMP';

NAME                             NUM_ROWS LAST_ANAL DISTINCT_KEYS
LEAF_BLOCKS AVG_LEAF_BLOCKS_PER_KEY JOI
------------------------------ ---------- --------- ------------- --
--------- ----------------------- ---
EMP_I1                             100000 07-APR-08            11
196                      17 NO

SQL>
SQL> @check_emp_histogram
SQL> set echo on
SQL>
SQL> select column_name, histogram, num_buckets
  2  from user_tab_columns
  3  where table_name='EMP';

COLUMN_NAME                    HISTOGRAM       NUM_BUCKETS
------------------------------ --------------- -----------
EMPNO                          NONE                      1
ENAME                          NONE                      1
PHONE                          NONE                      1
DEPTNO                         FREQUENCY                 7

SQL>
```

17) Perform step 7 again. What do you observe and why?

a) Because statistics were correctly gathered on both objects, the optimizer is able to use correct execution plans for both statements.

```
SQL> @check_exec_plans
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
```

Practice 7-2: Automatic Statistics Gathering (continued)

```
SQL>
SQL> set autotrace on
SQL>
SQL> select count(*), max(empno) from emp where deptno = 9;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99


Execution Plan
----------------------------------------------------------
Plan hash value: 3184478295

--------------------------------------------------------------------
-------------------
| Id  | Operation                    | Name   | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT             |        |     1 |     8 |
2   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE              |        |     1 |     8 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP    |    18 |   144 |
2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | EMP_I1 |    18 |       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------

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

   3 - access("DEPTNO"=9)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          3  consistent gets
          0  physical reads
          0  redo size
        481  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
SQL> select count(*), max(empno) from emp where deptno = 10;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000
```

```

Execution Plan
----------------------------------------------------------
Plan hash value: 2083865914

--------------------------------------------------------------------
-------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
--------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT   |      |     1 |     8 |   265   (1)|
00:00:03 |
|   1 |  SORT AGGREGATE    |      |     1 |     8 |            |
|
|*  2 |   TABLE ACCESS FULL| EMP  | 99864 |   780K|   265   (1)|
00:00:03 |
--------------------------------------------------------------------
-------

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

   2 - filter("DEPTNO"=10)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
        805  consistent gets
          0  physical reads
          0  redo size
        482  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

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

18) Ensure you delete all statistics previously generated on your objects.

```
SQL> @delete_stats
SQL> set echo on
SQL>
SQL> exec dbms_stats.delete_schema_stats('AGS');

PL/SQL procedure successfully completed.

SQL>

--------------------------------------------------------------
set echo on
```

Practice 7-2: Automatic Statistics Gathering (continued)

```
exec dbms_stats.delete_schema_stats('AGS');
```

19) Verify that you no longer have statistics gathered on both objects.

```
SQL> @check_table_last_analyzed
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT
last_analyzed,sample_size,monitoring,num_rows,blocks,table_name
  2  FROM user_tables
  3  WHERE table_name = 'EMP';

LAST_ANAL SAMPLE_SIZE MON   NUM_ROWS     BLOCKS TABLE_NAME
--------- ----------- --- ---------- ---------- --------------------
----------
                      YES                       EMP

SQL>
SQL> @check_index_last_analyzed
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT index_name name,num_rows,
  2         last_analyzed,distinct_keys,leaf_blocks,
  3         avg_leaf_blocks_per_key,join_index
  4  FROM user_indexes
  5  WHERE table_name = 'EMP';

NAME                             NUM_ROWS LAST_ANAL DISTINCT_KEYS
LEAF_BLOCKS AVG_LEAF_BLOCKS_PER_KEY JOI
------------------------------ ---------- --------- ------------- --
--------- ----------------------- ---
EMP_I1
NO

SQL>
SQL> @check_emp_histogram
SQL> set echo on
SQL>
SQL> select column_name, histogram, num_buckets
  2  from user_tab_columns
  3  where table_name='EMP';

COLUMN_NAME                    HISTOGRAM       NUM_BUCKETS
------------------------------ --------------- -----------
EMPNO                          NONE
ENAME                          NONE
PHONE                          NONE
DEPTNO                         NONE

SQL>
```

Practice 7-2: Automatic Statistics Gathering (continued)

20) How would you determine the list of automated tasks that exist on your database?

   a) You can use Enterprise Manager Database Control by navigating to the Automated Maintenance Tasks page (Home > Server> Automated Maintenance Tasks). On the Automated Maintenance Tasks page, you can see the three automated tasks implemented by default on your database.

   b) Another possibility is to use the `DBA_AUTOTASK_TASK` table as shown by the following statement:

```
SQL> select task_name from dba_autotask_task;

TASK_NAME
----------------------------------------------------------------
gather_stats_prog
auto_space_advisor_prog
AUTO_SQL_TUNING_PROG

SQL>
```

21) Exit from your SQL*Plus session.

```
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Automatic_Gather_Stats]$
```

22) You now want to observe the effects of the automatic statistics-gathering feature of your database. However, you do not want to wait until the database automatically opens the next maintenance window. So from your terminal session, execute the `run_ags.sh` script. This script forces the execution of the automatic statistics-gathering task.

```
[oracle@edrsr33p1-orcl Automatic_Gather_Stats]$ ./run_ags.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Mon Apr 7 16:00:03 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> exec dbms_workload_repository.create_snapshot;

PL/SQL procedure successfully completed.

SQL>
SQL> variable window varchar2(20);
SQL>
SQL> begin
```

Practice 7-2: Automatic Statistics Gathering (continued)

```
  2   select upper(to_char(sysdate,'fmday'))||'_WINDOW' into :window
from dual;
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL>
SQL> print window;

WINDOW
--------------------------------
MONDAY_WINDOW

SQL>
SQL> --
SQL> -- Open the corresponding maintenance window, but with other
clients disabled
SQL> --
SQL>
SQL> alter system set "_enable_automatic_maintenance"=1
  2  /

System altered.

SQL>
SQL> exec dbms_auto_task_admin.disable( -
>   'auto space advisor', null, :window);

PL/SQL procedure successfully completed.

SQL>
SQL> exec dbms_auto_task_admin.disable( -
>   'sql tuning advisor', null, :window);

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> exec dbms_scheduler.open_window(:window, null, true);

PL/SQL procedure successfully completed.

SQL>
SQL> --
SQL> -- Close the maintenance window when auto optimizer stats
collection is done
SQL> --
SQL>
SQL>
SQL> exec dbms_lock.sleep(120);

PL/SQL procedure successfully completed.

SQL>
SQL> exec dbms_scheduler.close_window(:window);
```

```
PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> alter system set "_enable_automatic_maintenance"=0
  2  /

System altered.

SQL>
SQL> --
SQL> -- Re-enable the other guys so they look like they are enabled
in EM.
SQL> -- Still they will be disabled because we have set the
underscore.
SQL> --
SQL>
SQL> exec dbms_auto_task_admin.enable( -
>   'auto space advisor', null, :window);

PL/SQL procedure successfully completed.

SQL>
SQL> exec dbms_auto_task_admin.enable( -
>   'sql tuning advisor', null, :window);

PL/SQL procedure successfully completed.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Automatic_Gather_Stats]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Automatic_Gather_Stats

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @run_ags.sql

--------------------------------------------------------------

set echo on

exec dbms_workload_repository.create_snapshot;
```

Practice 7-2: Automatic Statistics Gathering (continued)

```
variable window varchar2(20);

begin
 select upper(to_char(sysdate,'fmday'))||'_WINDOW' into :window from
dual;
end;
/

print window;

--
-- Open the corresponding maintenance window, but with other clients
disabled
--

alter system set "_enable_automatic_maintenance"=1
/

exec dbms_auto_task_admin.disable( -
  'auto space advisor', null, :window);

exec dbms_auto_task_admin.disable( -
  'sql tuning advisor', null, :window);


exec dbms_scheduler.open_window(:window, null, true);

--
-- Close the maintenance window when auto optimizer stats collection
is done
--


exec dbms_lock.sleep(120);

exec dbms_scheduler.close_window(:window);


alter system set "_enable_automatic_maintenance"=0
/

--
-- Re-enable the other guys so they look like they are enabled in
EM.
-- Still they will be disabled because we have set the underscore.
--

exec dbms_auto_task_admin.enable( -
  'auto space advisor', null, :window);

exec dbms_auto_task_admin.enable( -
  'sql tuning advisor', null, :window);

exit;
```

Practice 7-2: Automatic Statistics Gathering (continued)

23) Connect again as the AGS user from a SQL*Plus session.

```
[oracle@edrsr33p1-orcl Automatic_Gather_Stats]$ sqlplus ags/ags

SQL*Plus: Release 11.1.0.6.0 - Production on Mon Apr 7 16:02:44 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
```

24) View the statistics of your objects. What do you observe and why?

a) The statistics were automatically gathered by the database during the maintenance window. You can also see this directly from the Automated Maintenance Tasks page in Enterprise Manager. The important thing is that the database automatically gathered the right statistics and histograms. Depending on your environment, you may see different sample sizes.

```
SQL> @check_table_last_analyzed
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT
last_analyzed,sample_size,monitoring,num_rows,blocks,table_name
  2  FROM user_tables
  3  WHERE table_name = 'EMP';

LAST_ANAL SAMPLE_SIZE MON   NUM_ROWS     BLOCKS TABLE_NAME
--------- ----------- --- ---------- ---------- --------------------
----------
07-APR-08      100000 YES     100000        874 EMP

SQL>
SQL> @check_index_last_analyzed
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> SELECT index_name name,num_rows,
  2         last_analyzed,distinct_keys,leaf_blocks,
  3         avg_leaf_blocks_per_key,join_index
  4  FROM user_indexes
  5  WHERE table_name = 'EMP';

NAME                             NUM_ROWS LAST_ANAL DISTINCT_KEYS
LEAF_BLOCKS AVG_LEAF_BLOCKS_PER_KEY JOI
------------------------------ ---------- --------- ------------- --
--------- ----------------------- ---
EMP_I1                             100000 07-APR-08            11
196                     17 NO
```

Practice 7-2: Automatic Statistics Gathering (continued)

```
SQL>
SQL> @check_emp_histogram
SQL> set echo on
SQL>
SQL> select column_name, histogram, num_buckets
  2  from user_tab_columns
  3  where table_name='EMP';

COLUMN_NAME                    HISTOGRAM       NUM_BUCKETS
------------------------------ --------------- -----------
EMPNO                          NONE                      1
ENAME                          NONE                      1
PHONE                          NONE                      1
DEPTNO                         FREQUENCY                 5

SQL>
```

25) Perform step 7 again. What do you observe and why?

a) The optimizer can make the right decisions for both statements. This is because of the statistics that were automatically gathered by the database previously.

```
SQL> @check_exec_plans
SQL> set echo on
SQL>
SQL> set linesize 200 pagesize 1000
SQL>
SQL> set autotrace on
SQL>
SQL> select count(*), max(empno) from emp where deptno = 9;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99


Execution Plan
----------------------------------------------------------
Plan hash value: 3184478295

--------------------------------------------------------------------
-------------------
| Id  | Operation                   | Name   | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT            |        |     1 |     8 |
2   (0)| 00:00:01 |
|   1 |  SORT AGGREGATE             |        |     1 |     8 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP    |     9 |    72 |
2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN         | EMP_I1 |     9 |       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------
```

Practice 7-2: Automatic Statistics Gathering (continued)

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

   3 - access("DEPTNO"=9)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
          3  consistent gets
          0  physical reads
          0  redo size
        481  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

SQL>
SQL> select count(*), max(empno) from emp where deptno = 10;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000


Execution Plan
----------------------------------------------------------
Plan hash value: 2083865914

--------------------------------------------------------------------
-------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
--------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT   |      |     1 |     8 |   265   (1)|
00:00:03 |
|   1 |  SORT AGGREGATE    |      |     1 |     8 |            |
|
|*  2 |   TABLE ACCESS FULL| EMP  | 99863 |   780K|   265   (1)|
00:00:03 |
--------------------------------------------------------------------
-------

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

   2 - filter("DEPTNO"=10)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
```

```
        805  consistent gets
          0  physical reads
          0  redo size
        482  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed

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

26) Exit from your SQL*Plus session.

```
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Automatic_Gather_Stats]$
```

27) From your terminal window, clean up your environment by executing the `ags_cleanup.sh` script.

```
[oracle@edrsr33p1-orcl Automatic_Gather_Stats]$ ./ags_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Mon Apr 7 16:03:37 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user ags cascade;

User dropped.

SQL>
SQL> alter system set "_enable_automatic_maintenance"=1;

System altered.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Automatic_Gather_Stats]$

--------------------------------------------------------------
```

Practice 7-2: Automatic Statistics Gathering (continued)

```
#!/bin/bash

cd /home/oracle/solutions/Automatic_Gather_Stats

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin

sqlplus / as sysdba @ags_cleanup.sql

--------------------------------------------------------------

set echo on

drop user ags cascade;

alter system set "_enable_automatic_maintenance"=1;

exit;
```

Practices for Lesson 8

Practice 8-1: Understanding Adaptive Cusrsor Sharing

In this practice, you experiment with bind variable peeking and adaptive cursor sharing.

1) Connected to a terminal session as the `oracle` user, execute the `acs_setup.sh` script to set up the environment used for this lab. You can locate this script in your `$HOME/solutions/Adaptive_Cursor_Sharing` directory.

```
[oracle@edrsr33p1-orcl ~]$ cd solutions/Adaptive_Cursor_Sharing
[oracle@edrsr33p1-orcl Adaptive_Cursor_Sharing]$ ./acs_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Mar 28 16:54:36
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user acs cascade;
drop user acs cascade
          *
ERROR at line 1:
ORA-01918: user 'ACS' does not exist


SQL>
SQL> create user acs identified by acs default tablespace users
temporary tablespace temp;

User created.

SQL>
SQL> grant dba, connect to acs;

Grant succeeded.

SQL>
SQL> connect acs/acs
Connected.
SQL>
SQL> drop table emp purge;
drop table emp purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
SQL> create table emp
  2  (
  3   empno      number,
```

Practice 8-1: Understanding Adaptive Cusrsor Sharing (continued)

```
  4   ename      varchar2(20),
  5   phone      varchar2(20),
  6   deptno  number
  7  );

Table created.

SQL>
SQL>
SQL> insert into emp
  2    with tdata as
  3         (select rownum empno
  4           from all_objects
  5           where rownum <= 1000)
  6    select rownum,
  7           dbms_random.string ('u', 20),
  8           dbms_random.string ('u', 20),
  9           case
 10                  when rownum/100000 <= 0.001 then mod(rownum,
10)
 11                  else 10
 12           end
 13      from tdata a, tdata b
 14     where rownum <= 100000;

100000 rows created.

SQL>
SQL> create index emp_i1 on emp(deptno);

Index created.

SQL>
SQL> exec dbms_stats.gather_table_stats(null, 'EMP', METHOD_OPT =>
'FOR COLUMNS DEPTNO SIZE 10', CASCADE => TRUE);

PL/SQL procedure successfully completed.

SQL>
SQL> alter system flush shared_pool;

System altered.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Adaptive_Cursor_Sharing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Adaptive_Cursor_Sharing
```

Practice 8-1: Understanding Adaptive Cusrsor Sharing (continued)

```
export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @acs_setup.sql

--------------------------------------------------------------

set echo on

drop user acs cascade;

create user acs identified by acs default tablespace users temporary
tablespace temp;

grant dba, connect to acs;

connect acs/acs

drop table emp purge;

create table emp
(
 empno   number,
 ename   varchar2(20),
 phone   varchar2(20),
 deptno  number
);


insert into emp
  with tdata as
       (select rownum empno
         from all_objects
         where rownum <= 1000)
  select rownum,
         dbms_random.string ('u', 20),
         dbms_random.string ('u', 20),
         case
                when rownum/100000 <= 0.001 then mod(rownum, 10)
                else 10
         end
    from tdata a, tdata b
   where rownum <= 100000;

create index emp_i1 on emp(deptno);

exec dbms_stats.gather_table_stats(null, 'EMP', METHOD_OPT => 'FOR
COLUMNS DEPTNO SIZE 10', CASCADE => TRUE);

alter system flush shared_pool;
exit;
```

2) In your terminal session, connect to the SQL*Plus session as the ACS user. Ensure that you stay connected to the same SQL*Plus session until the end of this lab. After you get connected, identify the columns of the EMP table that have histograms.

   a) Only the DEPTNO column has a 10 buckets histogram.

```
[oracle@edrsr33p1-orcl Adaptive_Cursor_Sharing]$ sqlplus acs/acs

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Mar 28 16:54:49
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> @check_emp_histogram
SQL>
SQL> select column_name, histogram, num_buckets
  2  from user_tab_columns
  3  where table_name='EMP';

COLUMN_NAME                    HISTOGRAM       NUM_BUCKETS
------------------------------ --------------- -----------
EMPNO                          NONE
ENAME                          NONE
PHONE                          NONE
DEPTNO                         HEIGHT BALANCED          10

SQL>

--------------------------------------------------------------

set echo on

select column_name, histogram, num_buckets
from user_tab_columns
where table_name='EMP';
```

3) Determine the distribution of all the distinct values found in the DEPTNO column of the EMP table. What do you find?

   a) Values distribution is uniform for all of them (0.01%) except for value 10 (99.9%). This is typical of what is called data skew.

```
SQL> @show_deptno_distribution
SQL> set echo on
SQL>
SQL> select deptno, count(*) cnt_per_deptno, (count(*)*100)/nr
deptno_percent
```

```
  2  from emp, (select max(empno) nr
  3            from emp)
  4  group by deptno, nr
  5  order by deptno;

    DEPTNO CNT_PER_DEPTNO DEPTNO_PERCENT
---------- -------------- --------------
         0             10            .01
         1             10            .01
         2             10            .01
         3             10            .01
         4             10            .01
         5             10            .01
         6             10            .01
         7             10            .01
         8             10            .01
         9             10            .01
        10          99900           99.9

11 rows selected.

SQL>

--------------------------------------------------------------

set echo on

select deptno, count(*) cnt_per_deptno, (count(*)*100)/nr
deptno_percent
from emp, (select max(empno) nr
          from emp)
group by deptno, nr
order by deptno;
```

4) Before you study the adaptive cursor sharing feature, disable its functionality by setting the `OPTIMIZER_FEATURES_ENABLE` session parameter back to `10.2.0.1`. After this is done, ensure that you execute the following command in your SQL*Plus session: `set lines 200 pages 10000`. This is used in the lab to print execution plans correctly.

```
SQL> alter session set optimizer_features_enable="10.2.0.1";

Session altered.

SQL> set lines 200 pages 10000
```

5) Determine the execution plan for the following statement:
```
select /*ACS_L9*/ count(*), max(empno)
from emp
where deptno = 9;
```

What do you notice and why?

a) The optimizer uses an index range scan because value 9 is very selective.

Practice 8-1: Understanding Adaptive Cusrsor Sharing (continued)

```
SQL> @select_deptno_literal_9
SQL> set echo on
SQL>
SQL> select /*ACS_L9*/ count(*), max(empno)
  2  from emp
  3  where deptno = 9;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99

SQL>
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from table(dbms_xplan.display_cursor);

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL_ID  64ngy4j55d1z5, child number 0
-------------------------------------
select /*ACS_L9*/ count(*), max(empno) from emp where deptno = 9

Plan hash value: 3184478295

--------------------------------------------------------------------
-------------------
| Id  | Operation                     | Name   | Rows   | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT              |        |        |       |
2 (100)|          |
|   1 |  SORT AGGREGATE               |        |     1 |    16 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP    |    12 |   192 |
2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN           | EMP_I1 |    12 |       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------

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

   3 - access("DEPTNO"=9)


20 rows selected.

SQL>

--------------------------------------------------------------
```

```
set echo on

select /*ACS_L9*/ count(*), max(empno)
from emp
where deptno = 9;
```

6) Determine the execution plan for the following statement:

```
select /*ACS_L10*/ count(*), max(empno)
from emp
where deptno = 10;
```

What do you notice and why?

a) The optimizer uses a full table scan because value 10 is not a selective value.

```
SQL> @select_deptno_literal_10
SQL> set echo on
SQL>
SQL> select /*ACS_L10*/ count(*), max(empno)
  2  from emp
  3  where deptno = 10;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000

SQL>
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from table(dbms_xplan.display_cursor);

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------
---------------------------------------------------------------------
-----------------------------------------------------------------
SQL_ID  3232j5gkp2u5h, child number 0
-------------------------------------
select /*ACS_L10*/ count(*), max(empno) from emp where deptno = 10

Plan hash value: 2083865914

---------------------------------------------------------------------
-------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
---------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT    |      |       |       |   240 (100)|
|
|   1 |  SORT AGGREGATE     |      |     1 |    16 |            |
|
|*  2 |   TABLE ACCESS FULL| EMP  | 95000 |  1484K|   240   (1)|
00:00:03 |
```

Practice 8-1: Understanding Adaptive Cusrsor Sharing (continued)

```
--------------------------------------------------------------------
-------

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

   2 - filter("DEPTNO"=10)


19 rows selected.

SQL>

--------------------------------------------------------------

set echo on

select /*ACS_L10*/ count(*), max(empno)
from emp
where deptno = 10;
```

7) Define a bind variable called `DEPTNO` in your SQL*Plus session, set it to value 9, and execute the following query, and determine its execution plan:

```
select /*ACS_1*/ count(*), max(empno)
from emp
where deptno = :deptno;
```

What do you notice and why?

a) Because the optimizer uses bind peeking the first time you execute a statement with a bind variable, and because for this first execution, value 9 is used, the execution plan with index access is used.

```
SQL> variable deptno number;
SQL> exec :deptno := 9

PL/SQL procedure successfully completed.

SQL> @select_deptno_bind
SQL> set echo on
SQL>
SQL> select /*ACS_1*/ count(*), max(empno)
  2  from emp
  3  where deptno = :deptno;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99

SQL>
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from table(dbms_xplan.display_cursor);
```

Practice 8-1: Understanding Adaptive Cusrsor Sharing (continued)

```
PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL_ID  272gr4hapc9w1, child number 0
-------------------------------------
select /*ACS_1*/ count(*), max(empno) from emp where deptno =
:deptno

Plan hash value: 3184478295

--------------------------------------------------------------------
-------------------
| Id  | Operation                    | Name    | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT             |         |       |       |
2 (100)|          |
|   1 |  SORT AGGREGATE              |         |     1 |    16 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP     |    12 |   192 |
2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | EMP_I1 |    12 |       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------

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

   3 - access("DEPTNO"=:DEPTNO)


20 rows selected.

SQL>

--------------------------------------------------------------

set echo on

select /*ACS_1*/ count(*), max(empno)
from emp
where deptno = :deptno;

--------------------------------------------------------------

set echo on

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

Practice 8-1: Understanding Adaptive Cusrsor Sharing (continued)

8) Determine the execution statistics in terms of child cursors, executions, and buffer gets for the previously executed statement. What do you observe?

   a) In V$SQL, only one child cursor exists, and it has been executed only once (first time ever in this case). Also, the number of buffer gets is small due to the efficient access path that was used.

```
SQL> @show_latest_exec_stats
SQL> set echo on
SQL>
SQL> select child_number, executions, buffer_gets
  2  from v$sql
  3  where sql_text like 'select /*ACS_1%';

CHILD_NUMBER EXECUTIONS BUFFER_GETS
------------ ---------- -----------
           0          1           3

SQL>

--------------------------------------------------------------

set echo on

select child_number, executions, buffer_gets
from v$sql
where sql_text like 'select /*ACS_1%';
```

9) Perform steps 7 and 8 again, but this time using 10 as the bind value for DEPTNO. What do you observe and why?

   a) The execution plan is identical. The index path is used although value 10 is not selective. This is because bind peeking only operates the first time you execute your statement. Looking at V$SQL, you can clearly see that there is still only one child cursor associated with your statement. However, this time, the number of buffer gets was raised significantly due to the full table scan.

```
SQL> exec :deptno := 10

PL/SQL procedure successfully completed.

SQL> @select_deptno_bind
SQL> set echo on
SQL>
SQL> select /*ACS_1*/ count(*), max(empno)
  2  from emp
  3  where deptno = :deptno;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000

SQL>
SQL> @show_latest_exec_plan
```

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

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------
---------------------------------------------------------------------
-----------------------------------------------------------------
SQL_ID  272gr4hapc9w1, child number 0
-------------------------------------
select /*ACS_1*/ count(*), max(empno) from emp where deptno =
:deptno

Plan hash value: 3184478295

---------------------------------------------------------------------
-------------------
| Id  | Operation                   | Name   | Rows  | Bytes | Cost
(%CPU)| Time     |
---------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT            |        |       |       |
2 (100)|          |
|   1 |  SORT AGGREGATE             |        |     1 |    16 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP    |    12 |   192 |
2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | EMP_I1 |    12 |       |
1   (0)| 00:00:01 |
---------------------------------------------------------------------
-------------------

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

   3 - access("DEPTNO"=:DEPTNO)


20 rows selected.

SQL>
SQL> @show_latest_exec_stats
SQL> set echo on
SQL>
SQL> select child_number, executions, buffer_gets
  2  from v$sql
  3  where sql_text like 'select /*ACS_1%';

CHILD_NUMBER EXECUTIONS BUFFER_GETS
------------ ---------- -----------
           0          2         957

SQL>
```

10) Before the next step, flush your shared pool to make sure you wipe out all cursor's information.

Practice 8-1: Understanding Adaptive Cusrsor Sharing (continued)

```
SQL> alter system flush shared_pool;

System altered.

SQL>
```

11) Perform step 9 again. What do you observe and why?

   a) The execution plan is a full table scan because you used value 10 as your first bind value. There is only one child cursor that is created so far to handle your statement.

```
SQL> exec :deptno := 10

PL/SQL procedure successfully completed.

SQL> @select_deptno_bind
SQL> set echo on
SQL>
SQL> select /*ACS_1*/ count(*), max(empno)
  2  from emp
  3  where deptno = :deptno;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000

SQL>
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from table(dbms_xplan.display_cursor);

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------
---------------------------------------------------------------------
-----------------------------------------------------------------
SQL_ID  272gr4hapc9w1, child number 0
-------------------------------------
select /*ACS_1*/ count(*), max(empno) from emp where deptno =
:deptno

Plan hash value: 2083865914

---------------------------------------------------------------------
-------
| Id  | Operation           | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
---------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT    |      |       |       |   240 (100)|
|
|   1 |  SORT AGGREGATE     |      |     1 |    16 |            |
|
|*  2 |   TABLE ACCESS FULL| EMP  | 95000 |  1484K|   240   (1)|
00:00:03 |
```

```
---------------------------------------------------------------------
-------

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

   2 - filter("DEPTNO"=:DEPTNO)


19 rows selected.

SQL>
SQL> @show_latest_exec_stats
SQL> set echo on
SQL>
SQL> select child_number, executions, buffer_gets
  2  from v$sql
  3  where sql_text like 'select /*ACS_1%';

CHILD_NUMBER EXECUTIONS BUFFER_GETS
------------ ---------- -----------
           0          1         872

SQL>
```

12) Perform step 9 again, but this time use 9 as your bind value. What do you observe and why?

   a) Although value 9 is very selective, a full table scan is still used. This is because the second time you execute your statement, bind peeking is not done. So you continue to use the same child cursor.

```
SQL> exec :deptno := 9

PL/SQL procedure successfully completed.

SQL> @select_deptno_bind
SQL> set echo on
SQL>
SQL> select /*ACS_1*/ count(*), max(empno)
  2  from emp
  3  where deptno = :deptno;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99

SQL>
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from table(dbms_xplan.display_cursor);

PLAN_TABLE_OUTPUT
```

```
----------------------------------------------------------------------
----------------------------------------------------------------------
------------------------------------------------------------------
SQL_ID  272gr4hapc9w1, child number 0
-------------------------------------
select /*ACS_1*/ count(*), max(empno) from emp where deptno =
:deptno

Plan hash value: 2083865914

----------------------------------------------------------------------
-------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
----------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT   |      |       |       |   240 (100)|
|
|   1 |  SORT AGGREGATE    |      |     1 |    16 |            |
|
|*  2 |   TABLE ACCESS FULL| EMP  | 95000 |  1484K|   240   (1)|
00:00:03 |
----------------------------------------------------------------------
-------

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

   2 - filter("DEPTNO"=:DEPTNO)


19 rows selected.

SQL>
SQL> @show_latest_exec_stats
SQL> set echo on
SQL>
SQL> select child_number, executions, buffer_gets
  2  from v$sql
  3  where sql_text like 'select /*ACS_1%';

CHILD_NUMBER EXECUTIONS BUFFER_GETS
------------ ---------- -----------
           0          2        1693

SQL>
```

13) Before the next step, reset your session to use adaptive cursor sharing, and ensure that you flush your shared pool again.

```
SQL> alter session set optimizer_features_enable="11.1.0.6";

Session altered.

SQL> alter system flush shared_pool;
```

```
System altered.

SQL>
```

14) Perform step 12 again. What do you observe, and why?

a) Because this is the first time you execute the statement, bind peeking is used, and because value 9 is very selective, the index path is used. Only one child cursor is used to handle this statement.

```
SQL> exec :deptno := 9

PL/SQL procedure successfully completed.

SQL> @select_deptno_bind
SQL> set echo on
SQL>
SQL> select /*ACS_1*/ count(*), max(empno)
  2  from emp
  3  where deptno = :deptno;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99

SQL>
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from table(dbms_xplan.display_cursor);

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------
---------------------------------------------------------------------
-----------------------------------------------------------------
SQL_ID  272gr4hapc9w1, child number 0
-------------------------------------
select /*ACS_1*/ count(*), max(empno) from emp where deptno =
:deptno

Plan hash value: 3184478295

---------------------------------------------------------------------
-------------------
| Id  | Operation                    | Name   | Rows  | Bytes | Cost
(%CPU)| Time     |
---------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT             |        |       |       |
2 (100)|          |
|   1 |  SORT AGGREGATE              |        |     1 |    16 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP    |     1 |    16 |
2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN          | EMP_I1 |     1 |       |
1   (0)| 00:00:01 |
```

Practice 8-1: Understanding Adaptive Cusrsor Sharing (continued)

```
---------------------------------------------------------------------
-------------------

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

   3 - access("DEPTNO"=:DEPTNO)


20 rows selected.

SQL>
SQL> @show_latest_exec_stats
SQL> set echo on
SQL>
SQL> select child_number, executions, buffer_gets
  2  from v$sql
  3  where sql_text like 'select /*ACS_1%';

CHILD_NUMBER EXECUTIONS BUFFER_GETS
------------ ---------- -----------
           0          1          54

SQL>
```

15) Perform step 14 again, but this time using value 10 as your bind value. What do you observe and why?

   a) Although value 10 is not selective, the same index path as in the previous step is used. Only one child cursor is currently needed to represent your statement.

```
SQL> exec :deptno := 10

PL/SQL procedure successfully completed.

SQL> @select_deptno_bind
SQL> set echo on
SQL>
SQL> select /*ACS_1*/ count(*), max(empno)
  2  from emp
  3  where deptno = :deptno;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000

SQL>
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from table(dbms_xplan.display_cursor);

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------
---------------------------------------------------------------------
-----------------------------------------------------------------
```

Practice 8-1: Understanding Adaptive Cusrsor Sharing (continued)

```
SQL_ID  272gr4hapc9w1, child number 0
-------------------------------------
select /*ACS_1*/ count(*), max(empno) from emp where deptno =
:deptno

Plan hash value: 3184478295

--------------------------------------------------------------------
-------------------
| Id  | Operation                   | Name   | Rows  | Bytes | Cost
(%CPU)| Time     |
--------------------------------------------------------------------
-------------------
|   0 | SELECT STATEMENT            |        |       |       |
2 (100)|          |
|   1 |  SORT AGGREGATE             |        |     1 |    16 |
|          |
|   2 |   TABLE ACCESS BY INDEX ROWID| EMP    |     1 |    16 |
2   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN         | EMP_I1 |     1 |       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------

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

   3 - access("DEPTNO"=:DEPTNO)


20 rows selected.

SQL>
SQL> @show_latest_exec_stats
SQL> set echo on
SQL>
SQL> select child_number, executions, buffer_gets
  2  from v$sql
  3  where sql_text like 'select /*ACS_1%';

CHILD_NUMBER EXECUTIONS BUFFER_GETS
------------ ---------- -----------
           0          2        1008

SQL>
```

16) Perform step 15 again. What do you observe and why?

   a) Because you now use adaptive cursor sharing, the system realizes that you benefit from another child cursor for handling your statement. This time, a full table access path is used to better handle your statement.

```
SQL> exec :deptno := 10

PL/SQL procedure successfully completed.
```

Practice 8-1: Understanding Adaptive Cusrsor Sharing (continued)

```
SQL> @select_deptno_bind
SQL> set echo on
SQL>
SQL> select /*ACS_1*/ count(*), max(empno)
  2  from emp
  3  where deptno = :deptno;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000

SQL>
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from table(dbms_xplan.display_cursor);

PLAN_TABLE_OUTPUT
---------------------------------------------------------------------
---------------------------------------------------------------------
-----------------------------------------------------------------
SQL_ID  272gr4hapc9w1, child number 1
-------------------------------------
select /*ACS_1*/ count(*), max(empno) from emp where deptno =
:deptno

Plan hash value: 2083865914

---------------------------------------------------------------------
-------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)|
Time     |
---------------------------------------------------------------------
-------
|   0 | SELECT STATEMENT   |      |       |       |   240 (100)|
|
|   1 |  SORT AGGREGATE    |      |     1 |    16 |            |
|
|*  2 |   TABLE ACCESS FULL| EMP  | 95000 |  1484K|   240   (1)|
00:00:03 |
---------------------------------------------------------------------
-------

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

   2 - filter("DEPTNO"=:DEPTNO)


19 rows selected.

SQL>
SQL> @show_latest_exec_stats
SQL> set echo on
SQL>
SQL> select child_number, executions, buffer_gets
```

Practice 8-1: Understanding Adaptive Cusrsor Sharing (continued)

```
  2  from v$sql
  3  where sql_text like 'select /*ACS_1%';

CHILD_NUMBER EXECUTIONS BUFFER_GETS
------------ ---------- -----------
           0          2        1008
           1          1         821

SQL>
```

17) Exit your SQL*Plus session, and execute the `acs_cleanup.sh` script to clean up your environment.

```
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Adaptive_Cursor_Sharing]$ ./acs_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Mar 28 16:59:18
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user acs cascade;

User dropped.

SQL>
SQL> alter system flush shared_pool;

System altered.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Adaptive_Cursor_Sharing]$
```

Practice 8-2: Understanding `CURSOR_SHARING`

In this practice, you investigate the use of the `CURSOR_SHARING` initialization parameter.

1) You can find all the necessary scripts for this lab in your `$HOME/solutions/Cursor_Sharing` directory. First, you must set up the environment for this lab by executing the `cs_setup.sh` script from a terminal session connected as the `oracle` user. This script creates a new user called `CS` and the `EMP` table used throughout this lab.

```
[oracle@edrsr33p1-orcl ~]$ cd solutions/Cursor_Sharing
[oracle@edrsr33p1-orcl Cursor_Sharing]$
[oracle@edrsr33p1-orcl Cursor_Sharing]$ ./cs_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Mon Mar 31 14:10:59
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user cs cascade;

User dropped.

SQL>
SQL> create user cs identified by cs default tablespace users
temporary tablespace temp;

User created.

SQL>
SQL> grant dba, connect to cs;

Grant succeeded.

SQL>
SQL> connect cs/cs
Connected.
SQL>
SQL> drop table emp purge;
drop table emp purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
SQL> create table emp
```

Practice 8-2: Understanding CURSOR_SHARING (continued)

```
  2  (
  3   empno      number,
  4   ename      varchar2(20),
  5   phone      varchar2(20),
  6   deptno  number
  7  );

Table created.

SQL>
SQL>
SQL> insert into emp
  2    with tdata as
  3         (select rownum empno
  4           from all_objects
  5           where rownum <= 1000)
  6    select rownum,
  7           dbms_random.string ('u', 20),
  8           dbms_random.string ('u', 20),
  9           case
 10                  when rownum/100000 <= 0.001 then mod(rownum,
10)
 11                  else 10
 12           end
 13      from tdata a, tdata b
 14     where rownum <= 100000;

100000 rows created.

SQL>
SQL> create index emp_i1 on emp(deptno);

Index created.

SQL>
SQL> execute dbms_stats.gather_table_stats(null, 'EMP', cascade =>
true);

PL/SQL procedure successfully completed.

SQL>
SQL> alter system flush shared_pool;

System altered.

SQL>
SQL> connect / as sysdba
Connected.
SQL>
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
SQL> startup;
ORACLE instance started.
```

```
Total System Global Area  845348864 bytes
Fixed Size                  1303188 bytes
Variable Size             541068652 bytes
Database Buffers          297795584 bytes
Redo Buffers                5181440 bytes
Database mounted.
Database opened.
SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Cursor_Sharing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Cursor_Sharing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @cs_setup.sql

--------------------------------------------------------------

set echo on

drop user cs cascade;

create user cs identified by cs default tablespace users temporary
tablespace temp;

grant dba, connect to cs;

connect cs/cs

drop table emp purge;

create table emp
(
 empno   number,
 ename   varchar2(20),
 phone   varchar2(20),
 deptno  number
);


insert into emp
  with tdata as
```

Practice 8-2: Understanding CURSOR_SHARING (continued)

```
        (select rownum empno
          from all_objects
          where rownum <= 1000)
  select rownum,
          dbms_random.string ('u', 20),
          dbms_random.string ('u', 20),
          case
                 when rownum/100000 <= 0.001 then mod(rownum, 10)
                 else 10
          end
    from tdata a, tdata b
   where rownum <= 100000;

create index emp_i1 on emp(deptno);

execute dbms_stats.gather_table_stats(null, 'EMP', cascade => true);

alter system flush shared_pool;

connect / as sysdba

shutdown immediate;

startup;

exit;
```

2) From the same terminal session, connect as the `CS` user in the SQL*Plus session, and stay connected to that session until the end of this lab. For formatting reasons, after you have connected in the SQL*Plus session, execute the following command:
`set linesize 200 pagesize 1000`

```
[oracle@edrsr33p1-orcl Cursor_Sharing]$ sqlplus cs/cs

SQL*Plus: Release 11.1.0.6.0 - Production on Mon Mar 31 14:11:38
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options


SQL> set linesize 200 pagesize 1000
SQL>
```

3) Check the existence of histograms on the columns of the `EMP` table, and then determine the data distribution in the `DEPTNO` column of the `EMP` table. What do you observe?

Practice 8-2: Understanding CURSOR_SHARING (continued)

a) Currently, there are no histograms created on the columns of the EMP table. Also, it is clear that you have data skew in the DEPTNO column. Value 10 repeats most of the time (99.9%) whereas all other values only repeat 0.01%.

```
SQL> @check_emp_histogram
SQL>
SQL> select column_name, histogram, num_buckets
  2  from user_tab_columns
  3  where table_name='EMP';

COLUMN_NAME                    HISTOGRAM       NUM_BUCKETS
------------------------------ --------------- -----------
EMPNO                          NONE                      1
ENAME                          NONE                      1
PHONE                          NONE                      1
DEPTNO                         NONE                      1

SQL>
SQL> @show_deptno_distribution
SQL> set echo on
SQL>
SQL> select deptno, count(*) cnt_per_deptno, (count(*)*100)/nr
deptno_percent
  2  from emp, (select max(empno) nr
  3            from emp)
  4  group by deptno, nr
  5  order by deptno;

    DEPTNO CNT_PER_DEPTNO DEPTNO_PERCENT
---------- -------------- --------------
         0             10            .01
         1             10            .01
         2             10            .01
         3             10            .01
         4             10            .01
         5             10            .01
         6             10            .01
         7             10            .01
         8             10            .01
         9             10            .01
        10          99900           99.9

11 rows selected.

SQL>

--------------------------------------------------------------

set echo on

select column_name, histogram, num_buckets
from user_tab_columns
where table_name='EMP';

--------------------------------------------------------------

set echo on
```

Practice 8-2: Understanding CURSOR_SHARING (continued)

```
select deptno, count(*) cnt_per_deptno, (count(*)*100)/nr
deptno_percent
from emp, (select max(empno) nr
          from emp)
group by deptno, nr
order by deptno;
```

4) Before you continue, ensure that you flush your shared pool.

```
SQL> alter system flush shared_pool;

System altered.

SQL>
```

5) How would you force your SQL*Plus session to automatically replace statement literals with bind variables to make sure the same cursor is used independently of the literal values?

```
SQL> alter session set cursor_sharing = force;

Session altered.

SQL>
```

6) From the same SQL*Plus session, execute the following two queries, and then determine how many cursors are generated to handle these two statements, and what execution plans were used. What do you observe and why?
`select /*CS*/ count(*), max(empno) from emp where deptno = 9;`
`select /*CS*/ count(*), max(empno) from emp where deptno = 10;`

   a) Because of the previous step, literal values are replaced with bind variables. The `FORCE` option forces the system to share only one child cursor in this case and use the exact same execution plan (index range scan).

```
SQL> @select_deptno_literal_9
SQL> set echo on
SQL>
SQL> select /*CS*/ count(*), max(empno) from emp where deptno = 9;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99

SQL>
SQL> @select_deptno_literal_10
SQL> set echo on
SQL>
SQL> select /*CS*/ count(*), max(empno) from emp where deptno = 10;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000
```

Practice 8-2: Understanding CURSOR_SHARING (continued)

```
SQL>
SQL> @show_latest_cursors
SQL> set echo on
SQL>
SQL> col sql_text format a70
SQL>
SQL> select sql_text,hash_value
  2  from v$sql
  3  where sql_text like '%select /*CS%';

SQL_TEXT
HASH_VALUE
--------------------------------------------------------------------
-- ----------
select /*CS*/ count(*), max(empno) from emp where deptno =
:"SYS_B_0"  3434097775

SQL> @show_latest_exec_plans
SQL> set echo on
SQL>
SQL> col object_name format a5
SQL> col operation format a16
SQL> col options format a15
SQL>
SQL> select address,hash_value,child_number,
operation,options,object_name
  2  from v$sql_plan
  3  where (address,hash_value) in
  4    (select address,hash_value
  5     from v$sql
  6     where sql_text like '%select /*CS%');

ADDRESS  HASH_VALUE CHILD_NUMBER OPERATION        OPTIONS
OBJEC
-------- ---------- ------------ ---------------- --------------- --
---
4C9C53D4 3434097775            0 SELECT STATEMENT
4C9C53D4 3434097775            0 SORT             AGGREGATE
4C9C53D4 3434097775            0 TABLE ACCESS     BY INDEX ROWID
EMP
4C9C53D4 3434097775            0 INDEX            RANGE SCAN
EMP_I
                                                                   1


SQL>

--------------------------------------------------------------

set echo on

select /*CS*/ count(*), max(empno) from emp where deptno = 9;

--------------------------------------------------------------

set echo on
```

Practice 8-2: Understanding CURSOR_SHARING (continued)

```
select /*CS*/ count(*), max(empno) from emp where deptno = 10;

--------------------------------------------------------------

set echo on

col sql_text format a70

select sql_text,hash_value
from v$sql
where sql_text like '%select /*CS%';

--------------------------------------------------------------

set echo on

col object_name format a5
col operation format a16
col options format a15

select address,hash_value,child_number,
operation,options,object_name
from v$sql_plan
where (address,hash_value) in
  (select address,hash_value
   from v$sql
   where sql_text like '%select /*CS%');
```

7) Ensure that you create a 10 buckets histogram on the DEPTNO column of the EMP table.

```
SQL> exec dbms_stats.gather_table_stats(null, 'EMP', METHOD_OPT =>
'FOR COLUMNS DEPTNO SIZE 10', CASCADE => TRUE);

PL/SQL procedure successfully completed.

SQL> @check_emp_histogram
SQL> set echo on
SQL>
SQL> select column_name, histogram, num_buckets
  2  from user_tab_columns
  3  where table_name='EMP';

COLUMN_NAME                    HISTOGRAM       NUM_BUCKETS
------------------------------ --------------- -----------
EMPNO                          NONE                      1
ENAME                          NONE                      1
PHONE                          NONE                      1
DEPTNO                         HEIGHT BALANCED          10

SQL>

--------------------------------------------------------------

set echo on
```

```
select column_name, histogram, num_buckets
from user_tab_columns
where table_name='EMP';
```

8) Before you continue, ensure that you flush your shared pool.

```
SQL> alter system flush shared_pool;

System altered.

SQL>
```

9) Perform step 6 again. What do you observe and why?

   a) Although you captured histogram for the DEPTNO column that shows data skew, the system continues to share only one child cursor to handle both statements. This behavior is due to the FORCE option for the CURSOR_SHARING initialization parameter.

```
SQL> @select_deptno_literal_9
SQL> set echo on
SQL>
SQL> select /*CS*/ count(*), max(empno) from emp where deptno = 9;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99

SQL>
SQL> @select_deptno_literal_10
SQL> set echo on
SQL>
SQL> select /*CS*/ count(*), max(empno) from emp where deptno = 10;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000

SQL>
SQL> @show_latest_cursors
SQL> set echo on
SQL>
SQL> col sql_text format a70
SQL>
SQL> select sql_text,hash_value
  2  from v$sql
  3  where sql_text like '%select /*CS%';

SQL_TEXT
HASH_VALUE
--------------------------------------------------------------------
-- ----------
```

Practice 8-2: Understanding CURSOR_SHARING (continued)

```
select /*CS*/ count(*), max(empno) from emp where deptno =
:"SYS_B_0"  3434097775

SQL>

---------------------------------------------------------------

set echo on

select /*CS*/ count(*), max(empno) from emp where deptno = 9;

---------------------------------------------------------------

set echo on

select /*CS*/ count(*), max(empno) from emp where deptno = 10;

---------------------------------------------------------------

set echo on

col sql_text format a70

select sql_text,hash_value
from v$sql
where sql_text like '%select /*CS%';
```

10) Before you continue, ensure that you flush your shared pool.

```
SQL> alter system flush shared_pool;

System altered.

SQL>
```

11) How would you ensure that you now use more than one child cursor to handle both statements? Implement your solution, and check it.

a) By setting `CURSOR_SHARING` to `SIMILAR` for your session, the system is able to see that you benefit from using two different child cursors to handle both statements because they lend themselves to very different execution plans.

```
SQL> alter session set cursor_sharing = similar;

Session altered.

SQL> @select_deptno_literal_9
SQL> set echo on
SQL>
SQL> select /*CS*/ count(*), max(empno) from emp where deptno = 9;

  COUNT(*) MAX(EMPNO)
---------- ----------
        10         99
```

Practice 8-2: Understanding CURSOR_SHARING (continued)

```
SQL>
SQL> @select_deptno_literal_10
SQL> set echo on
SQL>
SQL> select /*CS*/ count(*), max(empno) from emp where deptno = 10;

  COUNT(*) MAX(EMPNO)
---------- ----------
     99900     100000

SQL>
SQL> @show_latest_cursors
SQL> set echo on
SQL>
SQL> col sql_text format a70
SQL>
SQL> select sql_text,hash_value
  2  from v$sql
  3  where sql_text like '%select /*CS%';

SQL_TEXT
HASH_VALUE
--------------------------------------------------------------------
-- ----------
select /*CS*/ count(*), max(empno) from emp where deptno =
:"SYS_B_0"  3434097775
select /*CS*/ count(*), max(empno) from emp where deptno =
:"SYS_B_0"  3434097775

SQL> @show_latest_exec_plans
SQL> set echo on
SQL>
SQL> col object_name format a5
SQL> col operation format a16
SQL> col options format a15
SQL>
SQL> select address,hash_value,child_number,
operation,options,object_name
  2  from v$sql_plan
  3  where (address,hash_value) in
  4    (select address,hash_value
  5     from v$sql
  6     where sql_text like '%select /*CS%');

ADDRESS  HASH_VALUE CHILD_NUMBER OPERATION        OPTIONS
OBJEC
-------- ---------- ------------ ---------------- --------------- --
---
4C9C53D4 3434097775            1 SELECT STATEMENT
4C9C53D4 3434097775            1 SORT             AGGREGATE
4C9C53D4 3434097775            1 TABLE ACCESS     FULL
EMP
4C9C53D4 3434097775            0 SELECT STATEMENT
4C9C53D4 3434097775            0 SORT             AGGREGATE
4C9C53D4 3434097775            0 TABLE ACCESS     BY INDEX ROWID
EMP
```

Practice 8-2: Understanding CURSOR_SHARING (continued)

```
4C9C53D4 3434097775            0 INDEX           RANGE SCAN
EMP_I
                                                                  1


7 rows selected.

SQL>

---------------------------------------------------------------

set echo on

select /*CS*/ count(*), max(empno) from emp where deptno = 9;

---------------------------------------------------------------

set echo on

select /*CS*/ count(*), max(empno) from emp where deptno = 10;

---------------------------------------------------------------

set echo on

col sql_text format a70

select sql_text,hash_value
from v$sql
where sql_text like '%select /*CS%';

---------------------------------------------------------------

set echo on

col object_name format a5
col operation format a16
col options format a15

select address,hash_value,child_number,
operation,options,object_name
from v$sql_plan
where (address,hash_value) in
  (select address,hash_value
   from v$sql
   where sql_text like '%select /*CS%');
```

12) Exit your SQL*Plus session and execute the `cs_cleanup.sh` script to clean up your environment for this lab.

```
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Cursor_Sharing]$
[oracle@edrsr33p1-orcl Cursor_Sharing]$ ./cs_cleanup.sh
```

```
SQL*Plus: Release 11.1.0.6.0 - Production on Mon Mar 31 14:15:49
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user cs cascade;

User dropped.

SQL>
SQL> alter system flush shared_pool;

System altered.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Cursor_Sharing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Cursor_Sharing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @cs_cleanup.sql

--------------------------------------------------------------

set echo on

drop user cs cascade;

alter system flush shared_pool;

exit;
```

Practices for Lesson 9

Practice 9-1: Using Hints

In this practice, you study five different hint cases. They are all independent from each other. **Note:** You can find all the necessary scripts used for this lab in your `$HOME/solutions/Hints` directory. You should be using a terminal session connected as the `oracle` user.

1) From your terminal session, execute the `iot_setup.sh` script. This script creates an index-organized table.

```
[oracle@edrsr33p1-orcl Hints]$ ./iot_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 15:17:24 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user iot cascade;
drop user iot cascade
          *
ERROR at line 1:
ORA-01918: user 'IOT' does not exist


SQL>
SQL> create user iot identified by iot default tablespace users
temporary tablespace temp;

User created.

SQL>
SQL> grant connect, resource, dba to iot;

Grant succeeded.

SQL>
SQL> connect iot/iot
Connected.
SQL>
SQL> drop table iottab purge;
drop table iottab purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
SQL> CREATE TABLE IOTTAB (
  2     OBJECT_ID      NUMBER(14, 0) NOT NULL ENABLE
  3   , OBJECT_ID_ATT  NUMBER(14, 0) NOT NULL ENABLE
```

Practice 9-1: Using Hints (continued)

```
  4   , OBJECT_ID_CAT  NUMBER(14, 0) NOT NULL ENABLE
  5   , BEGIN          DATE NOT NULL ENABLE
  6   , END            DATE NOT NULL ENABLE
  7   , STATUS         NUMBER
  8   , COMM           VARCHAR2(32) NOT NULL ENABLE
  9   , CONSTRAINT IOTTAB_PK
 10       PRIMARY KEY (OBJECT_ID
 11                  , OBJECT_ID_ATT
 12                  , OBJECT_ID_CAT
 13                  , BEGIN
 14                  , END) ENABLE )
 15  ORGANIZATION INDEX PCTTHRESHOLD 50 ;

Table created.

SQL>
SQL> CREATE INDEX OBJECT_ID_ATT_INDX ON IOTTAB (OBJECT_ID_ATT);

Index created.

SQL>
SQL> -- load data
SQL>
SQL> begin
  2  for i in 400001..500000 loop
  3    insert into iottab values(i,mod(i,428),mod(i,20),sysdate-
mod(i,100),sysdate+mod(i,100),mod(i,3),'aaaaaaaaaaaaaaaaaaaaaaaaa'|
|i);
  4  end loop;
  5  commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> begin
  2  for i in 100001..200000 loop
  3    insert into iottab values(i,mod(i,428),mod(i,20),sysdate-
mod(i,100),sysdate+mod(i,100),mod(i,3),'aaaaaaaaaaaaaaaaaaaaaaaaa'|
|i);
  4  end loop;
  5  commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> begin
  2  for i in 300001..400000 loop
  3    insert into iottab values(i,mod(i,428),mod(i,20),sysdate-
mod(i,100),sysdate+mod(i,100),mod(i,3),'aaaaaaaaaaaaaaaaaaaaaaaaa'|
|i);
  4  end loop;
```

Practice 9-1: Using Hints (continued)

```
  5  commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> begin
  2  for i in 500001..600000 loop
  3    insert into iottab values(i,mod(i,428),mod(i,20),sysdate-
mod(i,100),sysdate+mod(i,100),mod(i,3),'aaaaaaaaaaaaaaaaaaaaaaaaaa'|
|i);
  4  end loop;
  5  commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> begin
  2  for i in 1..100000 loop
  3    insert into iottab values(i,mod(i,428),mod(i,20),sysdate-
mod(i,100),sysdate+mod(i,100),mod(i,3),'aaaaaaaaaaaaaaaaaaaaaaaaaa'|
|i);
  4  end loop;
  5  commit;
  6  end;
  7  /

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL>
SQL> alter system flush shared_pool;

System altered.

SQL>
SQL> alter system flush buffer_cache;

System altered.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Hints]$

---------------------------------------------------------------

#!/bin/bash
```

Practice 9-1: Using Hints (continued)

```
cd /home/oracle/solutions/Hints

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @iot_setup.sql

--------------------------------------------------------------

set echo on

drop user iot cascade;

create user iot identified by iot default tablespace users temporary
tablespace temp;

grant connect, resource, dba to iot;

connect iot/iot

drop table iottab purge;

CREATE TABLE IOTTAB (
   OBJECT_ID      NUMBER(14, 0) NOT NULL ENABLE
 , OBJECT_ID_ATT  NUMBER(14, 0) NOT NULL ENABLE
 , OBJECT_ID_CAT  NUMBER(14, 0) NOT NULL ENABLE
 , BEGIN          DATE NOT NULL ENABLE
 , END            DATE NOT NULL ENABLE
 , STATUS         NUMBER
 , COMM           VARCHAR2(32) NOT NULL ENABLE
 , CONSTRAINT IOTTAB_PK
      PRIMARY KEY (OBJECT_ID
                 , OBJECT_ID_ATT
                 , OBJECT_ID_CAT
                 , BEGIN
                 , END) ENABLE )
ORGANIZATION INDEX PCTTHRESHOLD 50 ;

CREATE INDEX OBJECT_ID_ATT_INDX ON IOTTAB (OBJECT_ID_ATT);

-- load data

begin
for i in 400001..500000 loop
  insert into iottab values(i,mod(i,428),mod(i,20),sysdate-
mod(i,100),sysdate+mod(i,100),mod(i,3),'aaaaaaaaaaaaaaaaaaaaaaaaa'|
|i);
end loop;
commit;
end;
/
```

```
begin
for i in 100001..200000 loop
  insert into iottab values(i,mod(i,428),mod(i,20),sysdate-
mod(i,100),sysdate+mod(i,100),mod(i,3),'aaaaaaaaaaaaaaaaaaaaaaaaaa'|
|i);
end loop;
commit;
end;
/


begin
for i in 300001..400000 loop
  insert into iottab values(i,mod(i,428),mod(i,20),sysdate-
mod(i,100),sysdate+mod(i,100),mod(i,3),'aaaaaaaaaaaaaaaaaaaaaaaaaa'|
|i);
end loop;
commit;
end;
/


begin
for i in 500001..600000 loop
  insert into iottab values(i,mod(i,428),mod(i,20),sysdate-
mod(i,100),sysdate+mod(i,100),mod(i,3),'aaaaaaaaaaaaaaaaaaaaaaaaaa'|
|i);
end loop;
commit;
end;
/


begin
for i in 1..100000 loop
  insert into iottab values(i,mod(i,428),mod(i,20),sysdate-
mod(i,100),sysdate+mod(i,100),mod(i,3),'aaaaaaaaaaaaaaaaaaaaaaaaaa'|
|i);
end loop;
commit;
end;
/



alter system flush shared_pool;

alter system flush buffer_cache;

exit;
```

Practice 9-1: Using Hints (continued)

2) From your terminal session, connect as the `IOT` user in the SQL*Plus session. From your SQL*Plus session, execute the `set_session.sql` script. This script sets a number of parameters for the duration of this case. Ensure that you do not exit from your SQL*Plus session until asked to do so.

```
[oracle@edrsr33p1-orcl Hints]$ sqlplus iot/iot

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 15:18:22 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> @set_session

Session altered.

SQL>

--------------------------------------------------------------

ALTER SESSION SET OPTIMIZER_INDEX_CACHING=2;

set timing on
set linesize 200 pagesize 1000
```

3) From your SQL*Plus session, execute the following query and note down the time it takes to execute:

```
SELECT comm.
  FROM iottab
 WHERE object_id = 1
   AND object_id_cat = 0
   AND object_id_att = 426 ;
```

```
SQL> @select_iot
SQL>
SQL> SELECT comm
  2    FROM iottab
  3   WHERE object_id = 1
  4     AND object_id_cat = 0
  5     AND object_id_att = 426 ;

no rows selected

Elapsed: 00:00:00.07
SQL>

--------------------------------------------------------------

set echo on
```

Practice 9-1: Using Hints (continued)

```
SELECT comm
  FROM iottab
 WHERE object_id = 1
   AND object_id_cat = 0
   AND object_id_att = 426 ;
```

4) Use the DBMS_XPLAN package to display the execution plan associated with the statement you executed in the previous step. What do you observe?

   a) It is strange to see that the optimizer chooses a plan that accesses the secondary index while the query references all columns of the primary key.

```
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from
table(dbms_xplan.display_cursor(null,null,'TYPICAL'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL_ID  6u4tsfpprgvzn, child number 0
-------------------------------------
SELECT comm   FROM iottab  WHERE object_id = 1    AND object_id_cat
= 0
   AND object_id_att = 426

Plan hash value: 2544181447

--------------------------------------------------------------------
--------------------
| Id  | Operation         | Name                 | Rows  | Bytes |
Cost (%CPU)| Time     |
--------------------------------------------------------------------
--------------------
|   0 | SELECT STATEMENT  |                      |       |       |
1 (100)|          |
|*  1 |  INDEX UNIQUE SCAN| IOTTAB_PK            |     1 |    57 |
1   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN| OBJECT_ID_ATT_INDX |    50 |       |
1   (0)| 00:00:01 |
--------------------------------------------------------------------
-------------------

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

   1 - access("OBJECT_ID_ATT"=426)
       filter(("OBJECT_ID"=1 AND "OBJECT_ID_CAT"=0))
   2 - access("OBJECT_ID_ATT"=426)

Note
-----
   - dynamic sampling used for this statement
```

```
26 rows selected.

Elapsed: 00:00:00.13
SQL>

--------------------------------------------------------------

set echo on

select * from table(dbms_xplan.display_cursor(null,null,'TYPICAL'));
```

5) Before trying to fix this issue, make sure you flush the important content of your SGA by executing the `flush_sga.sql` script.

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

System altered.

Elapsed: 00:00:00.11
SQL>
SQL> alter system flush buffer_cache;

System altered.

Elapsed: 00:00:00.43
SQL>

--------------------------------------------------------------

set echo on

alter system flush shared_pool;

alter system flush buffer_cache;
```

6) Using hints only, how would you fix the issue raised at step 4? Implement your solution and check if it works.

   a) The idea is to use a hint to prevent the optimizer from using the secondary index. You will not use the `NO_INDEX` hint.

```
SQL> @select_iot_hint
SQL> set echo on
SQL>
SQL> SELECT /*+ NO_INDEX(t OBJECT_ID_ATT_INDX) */ comm
  2    FROM iottab t
  3   WHERE object_id = 1
  4     AND object_id_cat = 0
  5     AND object_id_att = 426 ;
```

Practice 9-1: Using Hints (continued)

```
no rows selected

Elapsed: 00:00:00.03
SQL>
SQL> @show_latest_exec_plan
SQL> set echo on
SQL>
SQL> select * from
table(dbms_xplan.display_cursor(null,null,'TYPICAL'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL_ID  4zcxy7z1cg8da, child number 0
-------------------------------------
SELECT /*+ NO_INDEX(t OBJECT_ID_ATT_INDX) */ comm   FROM iottab t
WHERE object_id = 1    AND object_id_cat = 0    AND object_id_att =
426

Plan hash value: 181430399

--------------------------------------------------------------------
----------
| Id  | Operation         | Name      | Rows  | Bytes | Cost (%CPU)|
Time     |
--------------------------------------------------------------------
----------
|   0 | SELECT STATEMENT |           |       |       |     1 (100)|
|
|*  1 |  INDEX RANGE SCAN| IOTTAB_PK |     1 |    57 |     1   (0)|
00:00:01 |
--------------------------------------------------------------------
----------

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

   1 - access("OBJECT_ID"=1 AND "OBJECT_ID_ATT"=426 AND
              "OBJECT_ID_CAT"=0)

Note
-----
   - dynamic sampling used for this statement


24 rows selected.

Elapsed: 00:00:00.13
SQL>

--------------------------------------------------------------

set echo on

SELECT /*+ NO_INDEX(t OBJECT_ID_ATT_INDX) */ comm
  FROM iottab t
```

Practice 9-1: Using Hints (continued)

```
 WHERE object_id = 1
   AND object_id_cat = 0
   AND object_id_att = 426 ;

--------------------------------------------------------------

set echo on

select * from table(dbms_xplan.display_cursor(null,null,'TYPICAL'));
```

7) Exit from your SQL*Plus session, and clean up your environment by executing the `iot_cleanup.sh` script.

```
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Hints]$
[oracle@edrsr33p1-orcl Hints]$ ./iot_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 15:21:09 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user iot cascade;

User dropped.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Hints]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Hints

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1
```

Practice 9-1: Using Hints (continued)

```
export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @iot_cleanup.sql

--------------------------------------------------------------

set echo on

drop user iot cascade;

exit;
```

8) You study a second case of hint utilization to specify hints in lower query blocks. From your terminal session connected as the `oracle` user, execute the `hr_hint_setup.sh` script.

```
[oracle@edrsr33p1-orcl Hints]$ ./hr_hint_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 16:08:51 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> alter user hr identified by hr account unlock;

User altered.

SQL>
SQL> grant dba to hr
  2
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Hints]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Hints

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1
```

Practice 9-1: Using Hints (continued)

```
export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @hr_hint_setup.sql

--------------------------------------------------------------

set echo on

alter user hr identified by hr account unlock;

grant dba to hr

exit;
```

9) From your terminal session, connect as the `HR` user in the SQL*Plus session. After you are connected, execute the `create_hr_view1.sql` script that creates a view called `V1` on top of the `EMPLOYEES` table. After this is done, execute the `create_hr_view2.sql` script that creates a view `V2` on top of `V1`.

```
[oracle@edrsr33p1-orcl Hints]$ sqlplus hr/hr

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 16:09:02 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> @create_hr_view1
SQL>
SQL> CREATE OR REPLACE VIEW v1 AS
  2    SELECT *
  3    FROM      employees
  4    WHERE  employee_id < 150;

View created.

SQL>
SQL> @create_hr_view2
SQL> set echo on
SQL>
SQL> CREATE OR REPLACE VIEW v2 AS
  2    SELECT v1.employee_id employee_id, departments.department_id
department_id
  3    FROM      v1, departments
  4    WHERE  v1.department_id = departments.department_id;

View created.
```

Practice 9-1: Using Hints (continued)

```
SQL>

--------------------------------------------------------------

set echo on

CREATE OR REPLACE VIEW v1 AS
  SELECT *
  FROM   employees
  WHERE  employee_id < 150;

--------------------------------------------------------------

set echo on

CREATE OR REPLACE VIEW v2 AS
  SELECT v1.employee_id employee_id, departments.department_id
department_id
  FROM   v1, departments
  WHERE  v1.department_id = departments.department_id;
```

10) Determine the execution plan used to process the following query:
`SELECT * FROM v2 WHERE department_id = 30;`

```
SQL> set linesize 200 pagesize 1000
SQL> @show_exec_plan_view2
SQL> set echo on
SQL>
SQL> explain plan for
  2  SELECT *
  3  FROM   v2
  4  WHERE  department_id = 30;

Explained.

SQL>
SQL> select * from table(DBMS_XPLAN.DISPLAY(null,null,'ALL'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
Plan hash value: 389887213

--------------------------------------------------------------------
------------------------------
| Id  | Operation                   | Name                | Rows  |
Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
------------------------------
|   0 | SELECT STATEMENT            |                     |     3 |
33 |     1   (0)| 00:00:01 |
|   1 |  NESTED LOOPS               |                     |     3 |
33 |     1   (0)| 00:00:01 |
|*  2 |   INDEX UNIQUE SCAN         | DEPT_ID_PK          |     1 |
4 |     0   (0)| 00:00:01 |
```

Practice 9-1: Using Hints (continued)

```
|*  3 |    TABLE ACCESS BY INDEX ROWID| EMPLOYEES         |      3 |
21 |      1   (0)| 00:00:01 |
|*  4 |     INDEX RANGE SCAN          | EMP_DEPARTMENT_IX |      6 |
|     0   (0)| 00:00:01 |
--------------------------------------------------------------------
------------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$5C160134
   2 - SEL$5C160134 / DEPARTMENTS@SEL$2
   3 - SEL$5C160134 / EMPLOYEES@SEL$3
   4 - SEL$5C160134 / EMPLOYEES@SEL$3

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

   2 - access("DEPARTMENTS"."DEPARTMENT_ID"=30)
   3 - filter("EMPLOYEE_ID"<150)
   4 - access("DEPARTMENT_ID"=30)

Column Projection Information (identified by operation id):
-----------------------------------------------------------

   1 - (#keys=0) "DEPARTMENTS"."DEPARTMENT_ID"[NUMBER,22],
"EMPLOYEE_ID"[NUMBER,22]
   2 - "DEPARTMENTS"."DEPARTMENT_ID"[NUMBER,22]
   3 - "EMPLOYEE_ID"[NUMBER,22]
   4 - "EMPLOYEES".ROWID[ROWID,10]

34 rows selected.

SQL>

--------------------------------------------------------------

set echo on

explain plan for
SELECT *
FROM   v2
WHERE  department_id = 30;

select * from table(DBMS_XPLAN.DISPLAY(null,null,'ALL'));
```

11) How do you force the query from step 10 to do a full table scan of the `departments` table and a range scan of the `emp_emp_id_pk` index?

   a) You have to use extended hint syntax to be able to specify hints that apply to tables and indexes that appear in views. The `NO_MERGE` hint is used to make sure that the view is not merged into the surrounding query blocks.

```
SQL> @show_exec_plan_view2_hints
SQL> set echo on
SQL>
```

Practice 9-1: Using Hints (continued)

```
SQL> explain plan for
  2  SELECT /*+ NO_MERGE(v2) INDEX(v2.v1.employees emp_emp_id_pk)
FULL(v2.departments) */ *
  3  FROM   v2
  4  WHERE  department_id = 30;

Explained.

SQL>
SQL> select * from table(DBMS_XPLAN.DISPLAY(null,null,'ALL'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
Plan hash value: 1511767168

--------------------------------------------------------------------
---------------------------
| Id  | Operation                    | Name          | Rows  |
Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
---------------------------
|   0 | SELECT STATEMENT             |               |     3 |
78 |     5   (0)| 00:00:01 |
|   1 |  VIEW                        | V2            |     3 |
78 |     5   (0)| 00:00:01 |
|   2 |   NESTED LOOPS               |               |       |
|            |          |
|   3 |    NESTED LOOPS              |               |     3 |
57 |     5   (0)| 00:00:01 |
|*  4 |     TABLE ACCESS FULL        | DEPARTMENTS   |     1 |
8 |     3   (0)| 00:00:01 |
|*  5 |     INDEX RANGE SCAN         | EMP_EMP_ID_PK |    50 |
|     1   (0)| 00:00:01 |
|*  6 |    TABLE ACCESS BY INDEX ROWID| EMPLOYEES     |     3 |
33 |     2   (0)| 00:00:01 |
--------------------------------------------------------------------
---------------------------

Query Block Name / Object Alias (identified by operation id):
-------------------------------------------------------------

   1 - SEL$335DD26A / V2@SEL$1
   2 - SEL$335DD26A
   4 - SEL$335DD26A / DEPARTMENTS@SEL$2
   5 - SEL$335DD26A / EMPLOYEES@SEL$3
   6 - SEL$335DD26A / EMPLOYEES@SEL$3

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

   4 - filter("DEPARTMENTS"."DEPARTMENT_ID"=30)
   5 - access("EMPLOYEE_ID"<150)
   6 - filter("DEPARTMENT_ID"=30)

Column Projection Information (identified by operation id):
```

Practice 9-1: Using Hints (continued)

```
------------------------------------------------------------

   1 - "V2"."EMPLOYEE_ID"[NUMBER,22],
"V2"."DEPARTMENT_ID"[NUMBER,22]
   2 - (#keys=0) "DEPARTMENTS"."DEPARTMENT_ID"[NUMBER,22],
"EMPLOYEE_ID"[NUMBER,22]
   3 - (#keys=0) "DEPARTMENTS"."DEPARTMENT_ID"[NUMBER,22],
       "EMPLOYEES".ROWID[ROWID,10], "EMPLOYEE_ID"[NUMBER,22]
   4 - "DEPARTMENTS"."DEPARTMENT_ID"[NUMBER,22]
   5 - "EMPLOYEES".ROWID[ROWID,10], "EMPLOYEE_ID"[NUMBER,22]

39 rows selected.

SQL>

---------------------------------------------------------------

set echo on

explain plan for
SELECT /*+ NO_MERGE(v2) INDEX(v2.v1.employees emp_emp_id_pk)
FULL(v2.departments) */ *
FROM   v2
WHERE  department_id = 30;

select * from table(DBMS_XPLAN.DISPLAY(null,null,'ALL'));
```

12) Exit from your SQL*Plus session and clean up your environment by executing the `hr_hint_cleanup.sh` script.

```
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Hints]$ ./hr_hint_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 16:10:09 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> revoke dba from hr;
revoke dba from hr
*
ERROR at line 1:
ORA-01951: ROLE 'DBA' not granted to 'HR'
```

```
SQL>
SQL> drop view hr.v1;

View dropped.

SQL>
SQL> drop view hr.v2;

View dropped.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Hints]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Hints

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @hr_hint_cleanup.sql

--------------------------------------------------------------

set echo on

revoke dba from hr;

drop view hr.v1;

drop view hr.v2;

exit;
```

13) In this third case, you investigate how to influence optimizer's joins. From your terminal session, execute the `sh_hint_setup.sh` script.

```
[oracle@edrsr33p1-orcl Hints]$ ./sh_hint_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 19:38:18 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.
```

Practice 9-1: Using Hints (continued)

```
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> alter user sh identified by sh acccount unlock;
alter user sh identified by sh acccount unlock
                               *
ERROR at line 1:
ORA-00922: missing or invalid option


SQL>
SQL> grant dba to sh;

Grant succeeded.

SQL>
SQL> connect sh/sh
Connected.
SQL>
SQL> exec dbms_stats.delete_schema_stats('SH');

PL/SQL procedure successfully completed.

SQL>
SQL> exec
dbms_stats.set_table_stats('SH','SALES',null,null,null,10,5);

PL/SQL procedure successfully completed.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Hints]$
[oracle@edrsr33p1-orcl Hints]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Hints

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @sh_hint_setup.sql
```

Practice 9-1: Using Hints (continued)

```
--------------------------------------------------------------

set echo on

alter user sh identified by sh acccount unlock;

grant dba to sh;

connect sh/sh

exec dbms_stats.delete_schema_stats('SH');

exec dbms_stats.set_table_stats('SH','SALES',null,null,null,10,5);

exit;
```

14) From your terminal session, connect as the `SH` user in the SQL*Plus session. After you are connected, ensure that you flush your SGA content using the `flush_sga.sql` script, and set some important session parameters using the `set_sh_session.sql` script. Ensure that you do not disconnect from your established SQL*Plus session.

```
[oracle@edrsr33p1-orcl Hints]$ sqlplus sh/sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 19:38:35 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> @flush_sga
SQL>
SQL> alter system flush shared_pool;

System altered.

SQL>
SQL> alter system flush buffer_cache;

System altered.

SQL>
SQL> @set_sh_session
SQL>
SQL> set linesize 200
SQL>
SQL> set pagesize 200
SQL>
SQL> set timing on
```

Practice 9-1: Using Hints (continued)

```
SQL>
SQL> alter session set optimizer_dynamic_sampling=0;

Session altered.

Elapsed: 00:00:00.00
SQL>

--------------------------------------------------------------

set echo on

alter system flush shared_pool;

alter system flush buffer_cache;

--------------------------------------------------------------

set linesize 200

set pagesize 200

set timing on

alter session set optimizer_dynamic_sampling=0;
```

15) From your SQL*Plus session, execute the following query and determine its execution plan:
`select count(*) from sales s, customers c where s.cust_id=c.cust_id;`

```
SQL> @exec_and_show_sh_exec_plan
SQL> set echo on
SQL>
SQL> select count(*) from sales s, customers c where
s.cust_id=c.cust_id;

  COUNT(*)
----------
    918843

Elapsed: 00:00:03.59
SQL>
SQL> select * from
table(dbms_xplan.display_cursor(null,null,'TYPICAL'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL_ID  88r79fy8nphrc, child number 0
-------------------------------------
select count(*) from sales s, customers c where s.cust_id=c.cust_id

Plan hash value: 2841872969
```

```
-----------------------------------------------------------------------
-----------------------------------
| Id  | Operation              | Name          | Rows   | Bytes | Cost
(%CPU)| Time      | Pstart| Pstop |
-----------------------------------------------------------------------
-----------------------------------
|   0 | SELECT STATEMENT       |               |        |       |     3
(100)|          |       |       |
|   1 |  SORT AGGREGATE        |               |     1  |    26 |
|          |       |       |
|   2 |   NESTED LOOPS         |               |   121K|  3081K|     3
(0)| 00:00:01 |       |       |
|   3 |    PARTITION RANGE ALL|               |    10  |   130 |     3
(0)| 00:00:01 |     1 |    28 |
|   4 |     TABLE ACCESS FULL | SALES         |    10  |   130 |     3
(0)| 00:00:01 |     1 |    28 |
|*  5 |    INDEX UNIQUE SCAN  | CUSTOMERS_PK | 12138 |   154K|     0
(0)|          |       |       |
-----------------------------------------------------------------------
-----------------------------------

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

   5 - access("S"."CUST_ID"="C"."CUST_ID")


22 rows selected.

Elapsed: 00:00:00.15
SQL>

---------------------------------------------------------------

set echo on

select count(*) from sales s, customers c where s.cust_id=c.cust_id;

select * from table(dbms_xplan.display_cursor(null,null,'TYPICAL'));
```

16) Before you continue, ensure that you flush the content of your SGA to avoid caching issues later. Use the `flush_sga.sql` script.

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

System altered.

Elapsed: 00:00:00.12
SQL>
SQL> alter system flush buffer_cache;

System altered.
```

Practice 9-1: Using Hints (continued)

```
Elapsed: 00:00:00.18
SQL>

---------------------------------------------------------------

set echo on

alter system flush shared_pool;

alter system flush buffer_cache;
```

17) Using hints only, how would you enhance the performance of the same query you executed in step 15? Make sure you verify your implementation.

   a) In step 15, the optimizer chose a `NESTED LOOPS` join. You can try to force a hash join instead, using the `LEADING` and `USE_HASH` hints.

```
SQL> @exec_and_show_sh_hint_exec_plan
SQL> set echo on
SQL>
SQL> select /*+ LEADING(c s) USE_HASH(s) */ count(*) from sales s,
customers c where s.cust_id=c.cust_id;

  COUNT(*)
----------
    918843

Elapsed: 00:00:00.40
SQL>
SQL> select * from
table(dbms_xplan.display_cursor(null,null,'TYPICAL'));

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------
--------------------------------------------------------------------
----------------------------------------------------------------
SQL_ID  9k8wcn7ckb7c1, child number 0
-------------------------------------
select /*+ LEADING(c s) USE_HASH(s) */ count(*) from sales s,
customers
c where s.cust_id=c.cust_id

Plan hash value: 3568173901

--------------------------------------------------------------------
-------------------------------------------
| Id  | Operation                | Name          | Rows   | Bytes
|TempSpc| Cost (%CPU)| Time      | Pstart| Pstop |
--------------------------------------------------------------------
-------------------------------------------
|   0 | SELECT STATEMENT         |               |        |        |
|   182 (100)|           |       |       |
|   1 |  SORT AGGREGATE          |               |      1 |     26 |
|            |           |       |       |
|*  2 |   HASH JOIN              |               |   121K|  3081K|
2968K|   182   (2)| 00:00:02 |       |       |
```

Practice 9-1: Using Hints (continued)

```
|   3 |     INDEX FAST FULL SCAN| CUSTOMERS_PK |   121K|  1540K|
|     9   (0)| 00:00:01 |       |       |
|   4 |     PARTITION RANGE ALL |              |    10 |   130 |
|     3   (0)| 00:00:01 |     1 |    28 |
|   5 |      TABLE ACCESS FULL  | SALES        |    10 |   130 |
|     3   (0)| 00:00:01 |     1 |    28 |
--------------------------------------------------------------------
-------------------------------------------

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

   2 - access("S"."CUST_ID"="C"."CUST_ID")


23 rows selected.

Elapsed: 00:00:00.13
SQL>

--------------------------------------------------------------

set echo on

select /*+ LEADING(c s) USE_HASH(s) */ count(*) from sales s,
customers c where s.cust_id=c.cust_id;

select * from table(dbms_xplan.display_cursor(null,null,'TYPICAL'));
```

18) Exit from your SQL*Plus session and clean up your environment by executing the `sh_hint_cleanup.sh` script.

```
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Hints]$
[oracle@edrsr33p1-orcl Hints]$ ./sh_hint_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 19:43:04 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> Connected.
SQL> SQL>
User altered.

SQL> SQL> revoke dba from sh
```

Practice 9-1: Using Hints (continued)

```
*
ERROR at line 1:
ORA-01951: ROLE 'DBA' not granted to 'SH'


SQL> SQL> SQL> Rem
SQL> Rem $Header: sh_main.sql 06-mar-2008.15:00:45 cbauwens Exp $
SQL> Rem
SQL> Rem sh_main.sql
SQL> Rem
...
SQL> Rem
SQL>
SQL> SET ECHO OFF

specify password for SH as parameter 1:

specify default tablespace for SH as parameter 2:

specify temporary tablespace for SH as parameter 3:

specify password for SYS as parameter 4:

specify directory path for the data files as parameter 5:

writeable directory path for the log files as parameter 6:

specify version as parameter 7:


Session altered.


User dropped.

old   1: CREATE USER sh IDENTIFIED BY &pass
new   1: CREATE USER sh IDENTIFIED BY sh

User created.

old   1: ALTER USER sh DEFAULT TABLESPACE &tbs
new   1: ALTER USER sh DEFAULT TABLESPACE example
old   2:  QUOTA UNLIMITED ON &tbs
new   2:  QUOTA UNLIMITED ON example

User altered.

old   1: ALTER USER sh TEMPORARY TABLESPACE &ttbs
new   1: ALTER USER sh TEMPORARY TABLESPACE temp

User altered.


Grant succeeded.


...
```

Practice 9-1: Using Hints (continued)

```
Grant succeeded.


PL/SQL procedure successfully completed.

Connected.

Grant succeeded.

old   1: CREATE OR REPLACE DIRECTORY data_file_dir AS '&data_dir'
new   1: CREATE OR REPLACE DIRECTORY data_file_dir AS
'/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/'

Directory created.

old   1: CREATE OR REPLACE DIRECTORY log_file_dir AS '&log_dir'
new   1: CREATE OR REPLACE DIRECTORY log_file_dir AS '/home/oracle/'

Directory created.


Grant succeeded.


Grant succeeded.


Grant succeeded.

Connected.

Session altered.


Session altered.


Table created.


...


Comment created.


Creating OLAP metadata ...
<<<<< CREATE CWMLite Metadata for the Sales History Schema >>>>>
-
…
-
<<<<< FINAL PROCESSING >>>>>
        - Changes have been committed

PL/SQL procedure successfully completed.
```

Practice 9-1: Using Hints (continued)

```
Commit complete.


gathering statistics ...

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Hints]$
```

19) With this fourth case, you study the influence of the `INDEX` and `AND_EQUAL` hints. From your terminal session, execute the `sh_hint_index_setup.sh` script to set up the environment for this lab.

```
[oracle@edrsr33p1-orcl Hints]$ ./sh_hint_index_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 20:16:22 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> alter user sh identified by sh acccount unlock;
alter user sh identified by sh acccount unlock
                               *
ERROR at line 1:
ORA-00922: missing or invalid option


SQL>
SQL> grant dba to sh;

Grant succeeded.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Hints]$

--------------------------------------------------------------
```

Practice 9-1: Using Hints (continued)

```
#!/bin/bash

cd /home/oracle/solutions/Hints

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @sh_hint_index_setup.sql

--------------------------------------------------------------

set echo on

alter user sh identified by sh acccount unlock;

grant dba to sh;

exit;
```

20) From your terminal session, connect to the SQL*Plus session as the SH user. After you are connected, execute the following scripts to further set up your environment for this lab. Ensure that you stay connected to your session throughout this case.

```
[oracle@edrsr33p1-orcl Hints]$ sqlplus sh/sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 20:16:28 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> @drop_index_customers
SQL>
SQL> @dait
SQL>
SQL> drop index CUSTOMERS_YOB_BIX;

Index dropped.

SQL> drop index CUSTOMERS_MARITAL_BIX;

Index dropped.

SQL> drop index CUSTOMERS_GENDER_BIX;
```

Practice 9-1: Using Hints (continued)

```
Index dropped.

SQL>
SQL>
SQL> @create_cust_indexes
SQL> set echo on
SQL>
SQL> CREATE INDEX CUST_CUST_GENDER_idx
  2  ON CUSTOMERS(CUST_GENDER)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL> CREATE INDEX CUST_CUST_POSTAL_CODE_idx
  2  ON CUSTOMERS(CUST_POSTAL_CODE)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>
SQL> CREATE INDEX CUST_CUST_CREDIT_LIMIT_idx
  2  ON CUSTOMERS(CUST_CREDIT_LIMIT)
  3  NOLOGGING COMPUTE STATISTICS;

Index created.

SQL>

---------------------------------------------------------------

REM    drop all indexes on CUSTOMERS table
REM    does not touch indexes associated with constraints
REM    ==================================================

set    termout off
store  set sqlplus_settings replace
save   buffer.sql replace
set    heading off verify off autotrace off feedback off

spool  dait.sql

SELECT 'drop index '||i.index_name||';'
FROM   user_indexes i
WHERE  i.table_name = 'CUSTOMERS'
AND    NOT EXISTS
      (SELECT 'x'
       FROM   user_constraints c
       WHERE  c.index_name = i.index_name
       AND    c.table_name = i.table_name
       AND    c.status = 'ENABLED');

spool  off

get    buffer.sql nolist
@sqlplus_settings
set    termout on
```

```
set echo on

@dait

--------------------------------------------------------------

set echo on

CREATE INDEX CUST_CUST_GENDER_idx
ON CUSTOMERS(CUST_GENDER)
NOLOGGING COMPUTE STATISTICS;

CREATE INDEX CUST_CUST_POSTAL_CODE_idx
ON CUSTOMERS(CUST_POSTAL_CODE)
NOLOGGING COMPUTE STATISTICS;

CREATE INDEX CUST_CUST_CREDIT_LIMIT_idx
ON CUSTOMERS(CUST_CREDIT_LIMIT)
NOLOGGING COMPUTE STATISTICS;
```

21) Determine the execution plan for the following query:

```
SELECT
c.*
FROM   customers c
WHERE  cust_gender   = 'M'
AND    cust_postal_code = 40804
AND   cust_credit_limit = 10000;
```

```
SQL> set autotrace traceonly
SQL> set linesize 200 pagesize 1000
SQL> @index
SQL> set echo on
SQL>
SQL> SELECT
  2  c.*
  3  FROM   customers c
  4  WHERE  cust_gender   = 'M'
  5  AND    cust_postal_code = 40804
  6  AND   cust_credit_limit = 10000
  7  /

6 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2008213504

--------------------------------------------------------------------
-----------
| Id  | Operation         | Name       | Rows   | Bytes | Cost (%CPU)|
Time      |
--------------------------------------------------------------------
-----------
```

```
|   0 | SELECT STATEMENT  |           |     6 |  1080 |   448   (1)|
00:00:05 |
|*  1 |  TABLE ACCESS FULL| CUSTOMERS |     6 |  1080 |   448   (1)|
00:00:05 |
--------------------------------------------------------------------
-----------

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

   1 - filter(TO_NUMBER("CUST_POSTAL_CODE")=40804 AND
              "CUST_CREDIT_LIMIT"=10000 AND "CUST_GENDER"='M')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       1460  consistent gets
          0  physical reads
          0  redo size
       2570  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed

SQL>

--------------------------------------------------------------

set echo on

SELECT
c.*
FROM   customers c
WHERE  cust_gender   = 'M'
AND    cust_postal_code = 40804
AND   cust_credit_limit = 10000
/
```

22) Try to get a better execution plan using the INDEX hint for the same query you investigated in step 21. Which index is best suited?

a) The CUST_CUST_GENDER_IDX index is the best one for this query. Note that using the INDEX hint without specifying any index leads the optimizer to use the CUST_CUST_GENDER_IDX index.

```
SQL> @index_hint
SQL> set echo on
SQL>
SQL> SELECT /*+ INDEX (c &indexname) */
  2  c.*
  3  FROM   customers c
  4  WHERE  cust_gender   = 'M'
```

```
  5  AND    cust_postal_code = 40804
  6  AND   cust_credit_limit = 10000
  7  /
Enter value for indexname: CUST_CUST_CREDIT_LIMIT_IDX
old   1: SELECT /*+ INDEX (c &indexname) */
new   1: SELECT /*+ INDEX (c CUST_CUST_CREDIT_LIMIT_IDX) */

6 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1407552528

--------------------------------------------------------------------
--------------------------------------
| Id  | Operation                   | Name                       |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
--------------------------------------
|   0 | SELECT STATEMENT            |                            |
6 |  1080 |  1074   (1)| 00:00:11 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                  |
6 |  1080 |  1074   (1)| 00:00:11 |
|*  2 |   INDEX RANGE SCAN          | CUST_CUST_CREDIT_LIMIT_IDX |
6938 |       |    14   (0)| 00:00:01 |
--------------------------------------------------------------------
--------------------------------------

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

   1 - filter(TO_NUMBER("CUST_POSTAL_CODE")=40804 AND
"CUST_GENDER"='M')
   2 - access("CUST_CREDIT_LIMIT"=10000)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       1053  consistent gets
         13  physical reads
          0  redo size
       2570  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed

SQL>
SQL> @index_hint
SQL> set echo on
SQL>
SQL> SELECT /*+ INDEX (c &indexname) */
  2  c.*
```

Practice 9-1: Using Hints (continued)

```
  3  FROM    customers c
  4  WHERE   cust_gender    = 'M'
  5  AND     cust_postal_code = 40804
  6  AND    cust_credit_limit = 10000
  7  /
Enter value for indexname: CUST_CUST_GENDER_IDX
old   1: SELECT /*+ INDEX (c &indexname) */
new   1: SELECT /*+ INDEX (c CUST_CUST_GENDER_IDX) */

6 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 3629874189

--------------------------------------------------------------------
--------------------------------
| Id  | Operation                   | Name                 | Rows  |
Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
--------------------------------
|   0 | SELECT STATEMENT            |                      |     6 |
1080 |  1164   (1)| 00:00:12 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS            |     6 |
1080 |  1164   (1)| 00:00:12 |
|*  2 |   INDEX RANGE SCAN          | CUST_CUST_GENDER_IDX | 27750 |
|    51   (0)| 00:00:01 |
--------------------------------------------------------------------
--------------------------------

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

   1 - filter(TO_NUMBER("CUST_POSTAL_CODE")=40804 AND
"CUST_CREDIT_LIMIT"=10000)
   2 - access("CUST_GENDER"='M')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       1399  consistent gets
         68  physical reads
          0  redo size
       2570  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed

SQL>
SQL> @index_hint
SQL> set echo on
SQL>
```

Practice 9-1: Using Hints (continued)

```
SQL> SELECT /*+ INDEX (c &indexname) */
  2  c.*
  3  FROM   customers c
  4  WHERE  cust_gender   = 'M'
  5  AND    cust_postal_code = 40804
  6  AND   cust_credit_limit = 10000
  7  /
Enter value for indexname: CUST_CUST_POSTAL_CODE_IDX
old   1: SELECT /*+ INDEX (c &indexname) */
new   1: SELECT /*+ INDEX (c CUST_CUST_POSTAL_CODE_IDX) */

6 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1928091631

--------------------------------------------------------------------
-------------------------------------
| Id  | Operation                   | Name                        |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
-------------------------------------
|   0 | SELECT STATEMENT            |                             |
6 |  1080 |   218   (1)| 00:00:03 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                   |
6 |  1080 |   218   (1)| 00:00:03 |
|*  2 |   INDEX FULL SCAN           | CUST_CUST_POSTAL_CODE_IDX |
89 |       |   134   (1)| 00:00:02 |
--------------------------------------------------------------------
-------------------------------------

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

   1 - filter("CUST_CREDIT_LIMIT"=10000 AND "CUST_GENDER"='M')
   2 - filter(TO_NUMBER("CUST_POSTAL_CODE")=40804)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        251  consistent gets
        132  physical reads
          0  redo size
       2570  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed

SQL>
SQL> @index_hint
SQL> set echo on
```

Practice 9-1: Using Hints (continued)

```
SQL>
SQL> SELECT /*+ INDEX (c &indexname) */
  2  c.*
  3  FROM   customers c
  4  WHERE  cust_gender   = 'M'
  5  AND    cust_postal_code = 40804
  6  AND   cust_credit_limit = 10000
  7  /
Enter value for indexname:
old   1: SELECT /*+ INDEX (c &indexname) */
new   1: SELECT /*+ INDEX (c ) */

6 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1928091631

--------------------------------------------------------------------
-------------------------------------
| Id  | Operation                   | Name                        |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
-------------------------------------
|   0 | SELECT STATEMENT            |                             |
6 |  1080 |   218   (1)| 00:00:03 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                   |
6 |  1080 |   218   (1)| 00:00:03 |
|*  2 |   INDEX FULL SCAN           | CUST_CUST_POSTAL_CODE_IDX |
89 |       |   134   (1)| 00:00:02 |
--------------------------------------------------------------------
-------------------------------------

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

   1 - filter("CUST_CREDIT_LIMIT"=10000 AND "CUST_GENDER"='M')
   2 - filter(TO_NUMBER("CUST_POSTAL_CODE")=40804)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        251  consistent gets
          0  physical reads
          0  redo size
       2570  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed

SQL>
```

Practice 9-1: Using Hints (continued)

```
--------------------------------------------------------------

set echo on

SELECT /*+ INDEX (c &indexname) */
c.*
FROM   customers c
WHERE  cust_gender   = 'M'
AND    cust_postal_code = 40804
AND   cust_credit_limit = 10000
/
```

23) Investigate the same query, but this time using the `AND_EQUAL` hint. Can you find a better execution plan with this hint?

   a) Using the `AND_EQUAL` hint allows the optimizer to explicitly choose an execution plan that uses an access path that merges the scans on several single-column indexes. However, no combination is better than the `CUST_CUST_GENDER_IDX` index.

```
SQL> @and_equal_hint
SQL> set echo on
SQL>
SQL> SELECT /*+ AND_EQUAL (c &index_name1, &index_name2) */
  2  c.*
  3  FROM   customers c
  4  WHERE  cust_gender   = 'M'
  5  AND    cust_postal_code = 40804
  6  AND   cust_credit_limit = 10000
  7  /
Enter value for index_name1: CUST_CUST_CREDIT_LIMIT_IDX
Enter value for index_name2: CUST_CUST_GENDER_IDX
old   1: SELECT /*+ AND_EQUAL (c &index_name1, &index_name2) */
new   1: SELECT /*+ AND_EQUAL (c CUST_CUST_CREDIT_LIMIT_IDX,
CUST_CUST_GENDER_IDX) */

6 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1121089724

--------------------------------------------------------------------
--------------------------------------
| Id  | Operation                   | Name                        |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
--------------------------------------
|   0 | SELECT STATEMENT            |                             |
6 |  1080 |  1416   (1)| 00:00:15 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                   |
6 |  1080 |  1416   (1)| 00:00:15 |
|   2 |   AND-EQUAL                 |                             |
|       |           |          |
```

```
|*  3 |     INDEX RANGE SCAN         | CUST_CUST_CREDIT_LIMIT_IDX |
6938 |       |    14   (0)| 00:00:01 |
|*  4 |     INDEX RANGE SCAN         | CUST_CUST_GENDER_IDX       |
27750 |       |    51   (0)| 00:00:01 |
--------------------------------------------------------------------
--------------------------------------

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

   1 - filter(TO_NUMBER("CUST_POSTAL_CODE")=40804 AND
"CUST_CREDIT_LIMIT"=10000 AND
              "CUST_GENDER"='M')
   3 - access("CUST_CREDIT_LIMIT"=10000)
   4 - access("CUST_GENDER"='M')


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
       9160  consistent gets
          0  physical reads
          0  redo size
       2570  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed

SQL>
SQL> @and_equal_hint
SQL> set echo on
SQL>
SQL> SELECT /*+ AND_EQUAL (c &index_name1, &index_name2) */
  2  c.*
  3  FROM   customers c
  4  WHERE  cust_gender   = 'M'
  5  AND    cust_postal_code = 40804
  6  AND   cust_credit_limit = 10000
  7  /
Enter value for index_name1: CUST_CUST_CREDIT_LIMIT_IDX
Enter value for index_name2: CUST_CUST_POSTAL_CODE_IDX
old   1: SELECT /*+ AND_EQUAL (c &index_name1, &index_name2) */
new   1: SELECT /*+ AND_EQUAL (c CUST_CUST_CREDIT_LIMIT_IDX,
CUST_CUST_POSTAL_CODE_IDX) */

6 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 1928091631

--------------------------------------------------------------------
-------------------------------------
```

Practice 9-1: Using Hints (continued)

```
| Id  | Operation                   | Name                       |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------
-------------------------------------
|   0 | SELECT STATEMENT            |                            |
6 |  1080 |   218   (1)| 00:00:03 |
|*  1 |  TABLE ACCESS BY INDEX ROWID| CUSTOMERS                  |
6 |  1080 |   218   (1)| 00:00:03 |
|*  2 |   INDEX FULL SCAN           | CUST_CUST_POSTAL_CODE_IDX |
89 |       |   134   (1)| 00:00:02 |
--------------------------------------------------------------------
-------------------------------------

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

   1 - filter("CUST_CREDIT_LIMIT"=10000 AND "CUST_GENDER"='M')
   2 - filter(TO_NUMBER("CUST_POSTAL_CODE")=40804)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        251  consistent gets
          0  physical reads
          0  redo size
       2570  bytes sent via SQL*Net to client
        420  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          6  rows processed

SQL>

--------------------------------------------------------------

set echo on

SELECT /*+ AND_EQUAL (c &index_name1, &index_name2) */
c.*
FROM   customers c
WHERE  cust_gender   = 'M'
AND    cust_postal_code = 40804
AND   cust_credit_limit = 10000
/
```

24) Exit from your SQL*Plus session and clean up your environment by executing the `sh_hint_cleanup.sh` script.

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

Practice 9-1: Using Hints (continued)

```
[oracle@edrsr33p1-orcl Hints]$
[oracle@edrsr33p1-orcl Hints]$ ./sh_hint_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 1 19:43:04 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> Connected.
SQL> SQL>
User altered.

SQL> SQL> revoke dba from sh
*
ERROR at line 1:
ORA-01951: ROLE 'DBA' not granted to 'SH'


SQL> SQL> SQL> Rem
SQL> Rem $Header: sh_main.sql 06-mar-2008.15:00:45 cbauwens Exp $
SQL> Rem
SQL> Rem sh_main.sql
SQL> Rem
...
SQL> Rem
SQL>
SQL> SET ECHO OFF

specify password for SH as parameter 1:

specify default tablespace for SH as parameter 2:

specify temporary tablespace for SH as parameter 3:

specify password for SYS as parameter 4:

specify directory path for the data files as parameter 5:

writeable directory path for the log files as parameter 6:

specify version as parameter 7:


Session altered.


User dropped.

old   1: CREATE USER sh IDENTIFIED BY &pass
new   1: CREATE USER sh IDENTIFIED BY sh
```

Practice 9-1: Using Hints (continued)

```
User created.

old   1: ALTER USER sh DEFAULT TABLESPACE &tbs
new   1: ALTER USER sh DEFAULT TABLESPACE example
old   2:  QUOTA UNLIMITED ON &tbs
new   2:  QUOTA UNLIMITED ON example

User altered.

old   1: ALTER USER sh TEMPORARY TABLESPACE &ttbs
new   1: ALTER USER sh TEMPORARY TABLESPACE temp

User altered.


Grant succeeded.


...


Grant succeeded.


PL/SQL procedure successfully completed.

Connected.

Grant succeeded.

old   1: CREATE OR REPLACE DIRECTORY data_file_dir AS '&data_dir'
new   1: CREATE OR REPLACE DIRECTORY data_file_dir AS
'/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/'

Directory created.

old   1: CREATE OR REPLACE DIRECTORY log_file_dir AS '&log_dir'
new   1: CREATE OR REPLACE DIRECTORY log_file_dir AS '/home/oracle/'

Directory created.


Grant succeeded.


Grant succeeded.


Grant succeeded.

Connected.

Session altered.


Session altered.
```

```
Table created.


...


Comment created.


Creating OLAP metadata ...
<<<<< CREATE CWMLite Metadata for the Sales History Schema >>>>>
…
<<<<< FINAL PROCESSING >>>>>
        - Changes have been committed

PL/SQL procedure successfully completed.


Commit complete.


gathering statistics ...

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Hints]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/SQL_Access_Advisor/sh

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

cp * $ORACLE_HOME/demo/schema/sales_history

sqlplus / as sysdba <<FIN!

SET ECHO ON
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 8000
```

Practice 9-1: Using Hints (continued)

```
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET LONG 1000

CONNECT / AS SYSDBA

alter user sh identified by sh account unlock;

revoke dba from sh;

@sh_main sh example temp oracle
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/
/home/oracle/ v3

exit;

FIN!
```

25) In the fifth case, you retrieve the first rows of your query as fast as possible. Connect to the SQL*Plus session as the `SYS` user and ensure that you set the following SQL*Plus environment variables:

```
timing on
autotrace on
pagesize 200
linesize 200
```

```
[oracle@edrsr33p1-orcl Hints]$ sqlplus / as sysdba

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Apr 29
21:02:19 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data
Mining
and Real Application Testing options

SQL> set timing on autotrace on pagesize 200 linesize 200
SQL>
```

26) From the same SQL*Plus session, execute the following query. Based on the fact that you want to retrieve the first 10 rows as fast as possible, what do you observe?

```
SELECT employee_id, department_name
FROM hr.employees e, hr.departments d
WHERE e.department_id = d.department_id;
```

a) It takes a bit of time before the result starts to be printed on the screen. This is because a merge join is used. It needs to sort all data before producing rows.

Practice 9-1: Using Hints (continued)

```
SQL> SELECT employee_id, department_name
FROM hr.employees e, hr.departments d
WHERE e.department_id = d.department_id;
  2    3
EMPLOYEE_ID DEPARTMENT_NAME
----------- ------------------------------
        200 Administration
        201 Marketing
        202 Marketing
        114 Purchasing
        115 Purchasing
        116 Purchasing
        117 Purchasing
…
        113 Finance
        205 Accounting
        206 Accounting

106 rows selected.

Elapsed: 00:00:00.09

Execution Plan
----------------------------------------------------------
Plan hash value: 1343509718

--------------------------------------------------------------
------------------------------
| Id  | Operation                    | Name          | Rows  |
Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------
------------------------------
|   0 | SELECT STATEMENT             |               |   106 |
2438 |     6  (17)| 00:00:01 |
|   1 |  MERGE JOIN                  |               |   106 |
2438 |     6  (17)| 00:00:01 |
|   2 |   TABLE ACCESS BY INDEX ROWID| DEPARTMENTS |    27 |
432 |     2   (0)| 00:00:01 |
|   3 |    INDEX FULL SCAN           | DEPT_ID_PK  |    27 |
|     1   (0)| 00:00:01 |
|*  4 |   SORT JOIN                  |               |   107 |
749 |     4  (25)| 00:00:01 |
|   5 |    TABLE ACCESS FULL         | EMPLOYEES   |   107 |
749 |     3   (0)| 00:00:01 |
--------------------------------------------------------------
------------------------------

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


   4 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
       filter("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")
```

Practice 9-1: Using Hints (continued)

```
Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
         19  consistent gets
          0  physical reads
          0  redo size
       2435  bytes sent via SQL*Net to client
        497  bytes received via SQL*Net from client
          9  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
        106  rows processed

SQL>
```

27) Using a hint, how can you ensure that the previous query starts fetching rows faster? Test your solution.

   a) Using the `FIRST_ROWS(10)` hint, a nested loop is chosen by the optimizer. It is faster to retrieve the first rows.

```
SQL> SELECT /*+ FIRST_ROWS(10) */ employee_id, department_name
FROM hr.employees e, hr.departments d
WHERE e.department_id = d.department_id;
  2    3
EMPLOYEE_ID DEPARTMENT_NAME
----------- ------------------------------
        200 Administration
        201 Marketing
        202 Marketing
        114 Purchasing
        115 Purchasing
        116 Purchasing
 …
        112 Finance
        113 Finance
        205 Accounting
        206 Accounting

106 rows selected.

Elapsed: 00:00:00.02

Execution Plan
----------------------------------------------------------
Plan hash value: 1021246405

--------------------------------------------------------------
------------------------------------
```

Practice 9-1: Using Hints (continued)

```
| Id  | Operation                    | Name              |
Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------
------------------------------------
|   0 | SELECT STATEMENT             |                   |
10 |   230 |     3   (0)| 00:00:01 |
|   1 |  NESTED LOOPS                |                   |
|       |            |          |
|   2 |   NESTED LOOPS               |                   |
10 |   230 |     3   (0)| 00:00:01 |
|   3 |    TABLE ACCESS FULL         | DEPARTMENTS       |
25 |   400 |     2   (0)| 00:00:01 |
|*  4 |    INDEX RANGE SCAN          | EMP_DEPARTMENT_IX |
8 |       |     0   (0)| 00:00:01 |
|   5 |   TABLE ACCESS BY INDEX ROWID| EMPLOYEES         |
3 |    21 |     1   (0)| 00:00:01 |
--------------------------------------------------------------
------------------------------------

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

   4 - access("E"."DEPARTMENT_ID"="D"."DEPARTMENT_ID")


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
         40  consistent gets
          0  physical reads
          0  redo size
       2435  bytes sent via SQL*Net to client
        497  bytes received via SQL*Net from client
          9  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        106  rows processed

SQL>
```

Practices for Lesson 10

Practice 10-1: Tracing Applications

In this practice, you define a service and use it to generate traces. You then interpret generated trace files. You can find all needed script files for this lab in your `$HOME/solutions/Application_Tracing` directory.

1) Initialize your environment be executing the `at_setup.sh` script from a terminal session connected as the `oracle` user.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./at_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:25:55 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user trace cascade;
drop user trace cascade
          *
ERROR at line 1:
ORA-01918: user 'TRACE' does not exist


SQL>
SQL> create user trace identified by trace default tablespace users
temporary tablespace temp;

User created.

SQL>
SQL> grant connect, resource, dba to trace;

Grant succeeded.

SQL>
SQL>
SQL> drop tablespace tracetbs including contents and datafiles;
drop tablespace tracetbs including contents and datafiles
*
ERROR at line 1:
ORA-00959: tablespace 'TRACETBS' does not exist


SQL>
SQL> drop tablespace tracetbs3 including contents and datafiles;
drop tablespace tracetbs3 including contents and datafiles
*
ERROR at line 1:
ORA-00959: tablespace 'TRACETBS3' does not exist
```

Practice 10-1: Tracing Applications (continued)

```
SQL>
SQL> create tablespace tracetbs
  2  datafile '/u01/app/oracle/oradata/orcl/tracetbs.dbf' size 100m
  3  extent management local uniform size 40k;

Tablespace created.

SQL>
SQL> create tablespace tracetbs3
  2  datafile '/u01/app/oracle/oradata/orcl/tracetbs3.dbf' size 100m
  3  extent management local uniform size 10m;

Tablespace created.

SQL>
SQL>
SQL> connect trace/trace
Connected.
SQL>
SQL> drop table sales purge;
drop table sales purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
SQL> create table sales as select * from sh.sales;

Table created.

SQL>
SQL>
SQL> drop table sales2 purge;
drop table sales2 purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
SQL> create table sales2 tablespace tracetbs as select * from
sh.sales where 1=2;

Table created.

SQL>
SQL> drop table sales3 purge;
drop table sales3 purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
```

Practice 10-1: Tracing Applications (continued)

```
SQL> create table sales3 tablespace tracetbs3 as select * from
sh.sales where 1=2;

Table created.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @at_setup.sql

--------------------------------------------------------------

set echo on

drop user trace cascade;

create user trace identified by trace default tablespace users
temporary tablespace temp;

grant connect, resource, dba to trace;


drop tablespace tracetbs including contents and datafiles;

drop tablespace tracetbs3 including contents and datafiles;

create tablespace tracetbs
datafile '/u01/app/oracle/oradata/orcl/tracetbs.dbf' size 100m
extent management local uniform size 40k;

create tablespace tracetbs3
datafile '/u01/app/oracle/oradata/orcl/tracetbs3.dbf' size 100m
extent management local uniform size 10m;


connect trace/trace

drop table sales purge;
```

```
create table sales as select * from sh.sales;


drop table sales2 purge;

create table sales2 tablespace tracetbs as select * from sh.sales
where 1=2;

drop table sales3 purge;

create table sales3 tablespace tracetbs3 as select * from sh.sales
where 1=2;

exit;
```

2) Before you can use a service in your applications, you have to make sure this service is available from the `tnsnames.ora` file you use to connect to your database. Modify this file to make sure it references a service called `TRACESERV`. You can use the `add_traceserv_tns.sh` script to help you in this task.

```
TRACESERV =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = node)(PORT = 1521))
      (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = TRACESERV)
    )
  )
```

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./add_traceserv_tns.sh
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/ksh

y=`hostname`

DBNAME=orcl

sed 's/NODE/'$y'/'
/home/oracle/solutions/Application_Tracing/wrong_tnstraceserv.ora >
/home/oracle/solutions/Application_Tracing/tnstraceserv.ora

cp /u01/app/oracle/product/11.1.0/db_1/network/admin/tnsnames.ora
/u01/app/oracle/product/11.1.0/db_1/network/admin/tnsnames.ora.bak1
cat /home/oracle/solutions/Application_Tracing/tnstraceserv.ora >>
/u01/app/oracle/product/11.1.0/db_1/network/admin/tnsnames.ora

--------------------------------------------------------------

TRACESERV =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = NODE)(PORT = 1521))
    (CONNECT_DATA =
```

Practice 10-1: Tracing Applications (continued)

```
      (SERVER = DEDICATED)
      (SERVICE_NAME = TRACESERV)
    )
  )
```

3) You now need to declare the TRACESERV service in your database. So connect to your database instance as the SYS user using a SQL*Plus session.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ sqlplus / as sysdba

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:26:15 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
```

4) In your SQL*Plus session, create a new service called TRACESERV with the same network name.

```
SQL> @add_traceserv_db
SQL>
SQL> select name from dba_services;

NAME
----------------------------------------------------------------
SYS$BACKGROUND
SYS$USERS
seeddataXDB
seeddata
orclXDB
orcl

6 rows selected.

SQL>
SQL> exec DBMS_SERVICE.CREATE_SERVICE('TRACESERV','TRACESERV');

PL/SQL procedure successfully completed.

SQL>
SQL> select name from dba_services;

NAME
----------------------------------------------------------------
SYS$BACKGROUND
SYS$USERS
seeddataXDB
seeddata
orclXDB
```

Practice 10-1: Tracing Applications (continued)

```
orcl
TRACESERV

7 rows selected.

SQL>
SQL>

--------------------------------------------------------------

set echo on

select name from dba_services;

exec DBMS_SERVICE.CREATE_SERVICE('TRACESERV','TRACESERV');

select name from dba_services;
```

5) From the same SQL*Plus session, start the `TRACESERV` service.

```
SQL> @start_traceserv
SQL> set echo on
SQL>
SQL> show parameter service_names

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------------------
-----------
service_names                        string
SQL>
SQL> exec DBMS_SERVICE.START_SERVICE('TRACESERV');

PL/SQL procedure successfully completed.

SQL>
SQL> show parameter service_names

NAME                                 TYPE        VALUE
------------------------------------ ----------- -------------------
-----------
service_names                        string      TRACESERV
SQL>
SQL>

--------------------------------------------------------------

set echo on

show parameter service_names

exec DBMS_SERVICE.START_SERVICE('TRACESERV');

show parameter service_names
```

Practice 10-1: Tracing Applications (continued)

6) Exit from your SQL*Plus session.

```
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Application_Tracing]$
```

7) At this point, open a browser window and connect to Enterprise Manager Database Control as the `SYS` user. Ensure that you navigate to the Top Services page.
   a) Log in to Enterprise Manager as the `SYS` user.
   b) On the Database Home page, click the Performance tab.
   c) On the Performance page, click the Top Consumers link in the Additional Monitoring links section of the page.
   d) On the Top Consumers page, click the Top Services tab. This takes you to the Top Services page.

8) You now execute seven workload scripts that are traced. All workload scripts run under the `TRACESERV` service. Your goal is to analyze the generated trace files to interpret what happens in the seven cases. From your terminal session, execute the `run_tracep0.sh` script. This script is used to trigger the generation of statistics for your `TRACESERV` service so it can be viewed from within Enterprise Manager. As soon as you start the execution of the `run_tracep0.sh` script, move to the next step of this lab.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./run_tracep0.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:28:40 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceP0';

Session altered.

SQL>
SQL> set termout off
SQL>
SQL> exit;
```

Practice 10-1: Tracing Applications (continued)

```
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @run_tracep0.sql

--------------------------------------------------------------

set echo on

connect trace/trace@TRACESERV

alter session set tracefile_identifier='mytraceP0';

set termout off

select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;
select count(*) from dba_objects;

exec dbms_lock.sleep(60);

set termout on

exit;
```

Practice 10-1: Tracing Applications (continued)

9) Go back to the Top Consumers page in your Enterprise Manager session. Wait until you see the TRACESERV service in the Active Services table, and enable tracing for that service.

   a) When you see TRACESERV in the Active Services table on the Top Services page, select it, and click the Enable SQL Trace button.

   b) On the Enable SQL Trace page, make sure Waits is set to TRUE, and Binds is set to FALSE. Click OK.

   c) Back to the Top Services page, you should see a confirmation message near the top of the Top Consumers page.

10) When tracing for TRACESERV is enabled, execute the `run_tracep1.sh` script from your terminal session. Observe your screen.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./run_tracep1.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:29:48 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set workarea_size_policy=manual;

Session altered.

SQL>
SQL> alter session set sort_area_size=50000;

Session altered.

SQL>
SQL> alter session set hash_area_size=5000;

Session altered.

SQL>
SQL>
SQL> alter session set tracefile_identifier='mytraceP1';

Session altered.

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

Practice 10-1: Tracing Applications (continued)

```
SQL>
SQL> select /*+ ORDERED USE_HASH(s2) */ count(*) from sales s1,
sales s2 where s1.cust_id=s2.cust_id;

  COUNT(*)
----------
 172878975

Elapsed: 00:01:19.25
SQL>
SQL>
SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceS1';

Session altered.

Elapsed: 00:00:00.00
SQL>
SQL> set timing on
SQL>
SQL> select /*+ ORDERED USE_HASH(s2) S1 */ count(*) from sales s1,
sales s2 where s1.cust_id=s2.cust_id;

  COUNT(*)
----------
 172878975

Elapsed: 00:00:40.19
SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @run_tracep1.sql

--------------------------------------------------------------
```

Practice 10-1: Tracing Applications (continued)

```
[oracle@edrsr33p1-orcl Application_Tracing]$ cat run_tracep1.sql
set echo on

connect trace/trace@TRACESERV

alter session set workarea_size_policy=manual;

alter session set sort_area_size=50000;

alter session set hash_area_size=5000;


alter session set tracefile_identifier='mytraceP1';


set timing on

select /*+ ORDERED USE_HASH(s2) */ count(*) from sales s1, sales s2
where s1.cust_id=s2.cust_id;



connect trace/trace@TRACESERV

alter session set tracefile_identifier='mytraceS1';

set timing on

select /*+ ORDERED USE_HASH(s2) S1 */ count(*) from sales s1, sales
s2 where s1.cust_id=s2.cust_id;

exit;
```

11) Execute the `run_tracep2.sh` script from your terminal session. Observe your screen.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./run_tracep2.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:31:57 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceP2';

Session altered.
```

Practice 10-1: Tracing Applications (continued)

```
SQL>
SQL> set timing on
SQL>
SQL> declare
  2     c      number := dbms_sql.open_cursor;
  3     oname  varchar2(50);
  4     ignore integer;
  5  begin
  6    for i in 1 .. 5000 loop
  7     dbms_sql.parse(c,'select object_name from dba_objects where
object_id = '||i , dbms_sql.native);  -- use literal
  8     dbms_sql.define_column(c, 1, oname, 50);
  9     ignore := dbms_sql.execute(c);
 10     if dbms_sql.fetch_rows(c)>0 then
 11       dbms_sql.column_value(c, 1, oname);
 12     end if;
 13    end loop;
 14    dbms_sql.close_cursor(c);
 15  end;
 16  /

PL/SQL procedure successfully completed.

Elapsed: 00:00:49.36
SQL>
SQL>
SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceS2';

Session altered.

Elapsed: 00:00:00.00
SQL>
SQL> declare
  2     c number := dbms_sql.open_cursor;
  3     oname  varchar2(50);
  4     ignore integer;
  5  begin
  6     dbms_sql.parse(c,'select object_name from dba_objects where
object_id = :y' , dbms_sql.native); -- use bind var
  7     for i in 1 .. 5000 loop
  8       dbms_sql.bind_variable(c,':y',i);
  9       dbms_sql.define_column(c, 1, oname, 50);
 10       ignore := dbms_sql.execute(c);
 11       if dbms_sql.fetch_rows(c)>0 then
 12         dbms_sql.column_value(c, 1, oname);
 13      end if;
 14     end loop;
 15     dbms_sql.close_cursor(c);
 16  end;
 17  /

PL/SQL procedure successfully completed.
```

Practice 10-1: Tracing Applications (continued)

```
Elapsed: 00:00:00.86
SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @run_tracep2.sql

--------------------------------------------------------------

set echo on

connect trace/trace@TRACESERV

alter session set tracefile_identifier='mytraceP2';

set timing on

declare
   c      number := dbms_sql.open_cursor;
   oname  varchar2(50);
   ignore integer;
begin
  for i in 1 .. 5000 loop
    dbms_sql.parse(c,'select object_name from dba_objects where
object_id = '||i , dbms_sql.native);  -- use literal
    dbms_sql.define_column(c, 1, oname, 50);
    ignore := dbms_sql.execute(c);
    if dbms_sql.fetch_rows(c)>0 then
      dbms_sql.column_value(c, 1, oname);
    end if;
  end loop;
  dbms_sql.close_cursor(c);
end;
/



connect trace/trace@TRACESERV
```

Practice 10-1: Tracing Applications (continued)

```
alter session set tracefile_identifier='mytraceS2';

declare
   c number := dbms_sql.open_cursor;
   oname  varchar2(50);
   ignore integer;
begin
   dbms_sql.parse(c,'select object_name from dba_objects where
object_id = :y' , dbms_sql.native); -- use bind var
   for i in 1 .. 5000 loop
     dbms_sql.bind_variable(c,':y',i);
     dbms_sql.define_column(c, 1, oname, 50);
     ignore := dbms_sql.execute(c);
     if dbms_sql.fetch_rows(c)>0 then
       dbms_sql.column_value(c, 1, oname);
    end if;
   end loop;
   dbms_sql.close_cursor(c);
end;
/

exit;
```

12) Execute the `run_tracep3.sh` script from your terminal session. Observe your screen.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./run_tracep3.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:32:53 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceP3';

Session altered.

SQL>
SQL> update sales set amount_sold=20000 where prod_id=13 and
cust_id=987;

2 rows updated.

SQL>
SQL> commit;
```

Practice 10-1: Tracing Applications (continued)

```
Commit complete.

SQL>
SQL>
SQL> connect trace/trace
Connected.
SQL>
SQL> create index sales_prod_cust_indx on sales(prod_id,cust_id);

Index created.

SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceS3';

Session altered.

SQL>
SQL> update sales set amount_sold=30000 where prod_id=13 and
cust_id=987;

2 rows updated.

SQL>
SQL> commit;

Commit complete.

SQL>
SQL> connect trace/trace
Connected.
SQL>
SQL> drop index sales_prod_cust_indx;

Index dropped.

SQL>
SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1
```

Practice 10-1: Tracing Applications (continued)

```
export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @run_tracep3.sql

--------------------------------------------------------------

set echo on

connect trace/trace@TRACESERV

alter session set tracefile_identifier='mytraceP3';

update sales set amount_sold=20000 where prod_id=13 and cust_id=987;

commit;


connect trace/trace

create index sales_prod_cust_indx on sales(prod_id,cust_id);

connect trace/trace@TRACESERV

alter session set tracefile_identifier='mytraceS3';

update sales set amount_sold=30000 where prod_id=13 and cust_id=987;

commit;

connect trace/trace

drop index sales_prod_cust_indx;


exit;
```

13) Execute the `run_tracep4.sh` script from your terminal session. Observe your screen.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./run_tracep4.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:33:08 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
```

Practice 10-1: Tracing Applications (continued)

```
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceP4';

Session altered.

SQL>
SQL> set timing on
SQL>
SQL> DECLARE
       TYPE SalesCurTyp  IS REF CURSOR;
       v_sales_cursor    SalesCurTyp;
       sales_record      sh.sales%ROWTYPE;
       v_stmt_str         VARCHAR2(200);
     BEGIN
       -- Dynamic SQL statement with placeholder:
       v_stmt_str := 'select * from sh.sales where amount_sold>0';
       -- Open cursor and specify bind argument in USING clause:
       OPEN v_sales_cursor FOR v_stmt_str;
       -- Fetch rows from result set one at a time:
       LOOP
         FETCH v_sales_cursor INTO sales_record;
         EXIT WHEN v_sales_cursor%NOTFOUND;
       END LOOP;
       -- Close cursor:
       CLOSE v_sales_cursor;
     END;
     /

PL/SQL procedure successfully completed.

Elapsed: 00:00:26.84
SQL>
SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceS4';

Session altered.

Elapsed: 00:00:00.00
SQL>
SQL> set timing on
SQL>
SQL> DECLARE
       TYPE SalesCurTyp  IS REF CURSOR;
       TYPE SalesList IS TABLE OF sh.sales%ROWTYPE;
       v_sales_cursor    SalesCurTyp;
       sales_List        SalesList;
       v_stmt_str         VARCHAR2(200);
     BEGIN
       -- Dynamic SQL statement with placeholder:
```

```
   v_stmt_str := 'select /* S4 */ * from sh.sales where
amount_sold>0';

   -- Open cursor:
   OPEN v_sales_cursor FOR v_stmt_str;

   -- Fetch rows from result set one at a time:
   LOOP
     FETCH v_sales_cursor BULK COLLECT INTO Sales_List LIMIT
10000;
     EXIT WHEN v_sales_cursor%NOTFOUND;
   END LOOP;

   -- Close cursor:
   CLOSE v_sales_cursor;
 END;
 /

PL/SQL procedure successfully completed.

Elapsed: 00:00:02.09
SQL>
SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @run_tracep4.sql

--------------------------------------------------------------
set echo on

connect trace/trace@TRACESERV

alter session set tracefile_identifier='mytraceP4';

set timing on

DECLARE
  TYPE SalesCurTyp  IS REF CURSOR;
```

Practice 10-1: Tracing Applications (continued)

```
  v_sales_cursor    SalesCurTyp;
  sales_record      sh.sales%ROWTYPE;
  v_stmt_str      VARCHAR2(200);
BEGIN
  -- Dynamic SQL statement with placeholder:
  v_stmt_str := 'select * from sh.sales where amount_sold>0';

  -- Open cursor and specify bind argument in USING clause:
  OPEN v_sales_cursor FOR v_stmt_str;

  -- Fetch rows from result set one at a time:
  LOOP
    FETCH v_sales_cursor INTO sales_record;
    EXIT WHEN v_sales_cursor%NOTFOUND;
  END LOOP;

  -- Close cursor:
  CLOSE v_sales_cursor;
END;
/


connect trace/trace@TRACESERV

alter session set tracefile_identifier='mytraceS4';

set timing on

DECLARE
  TYPE SalesCurTyp  IS REF CURSOR;
  TYPE SalesList IS TABLE OF sh.sales%ROWTYPE;
  v_sales_cursor    SalesCurTyp;
  sales_List        SalesList;
  v_stmt_str      VARCHAR2(200);
BEGIN
  -- Dynamic SQL statement with placeholder:
  v_stmt_str := 'select /* S4 */ * from sh.sales where
amount_sold>0';

  -- Open cursor:
  OPEN v_sales_cursor FOR v_stmt_str;

  -- Fetch rows from result set one at a time:
  LOOP
    FETCH v_sales_cursor BULK COLLECT INTO Sales_List LIMIT 10000;
    EXIT WHEN v_sales_cursor%NOTFOUND;
  END LOOP;

  -- Close cursor:
  CLOSE v_sales_cursor;
END;
/


exit;
```

Practice 10-1: Tracing Applications (continued)

14) Execute the `run_tracep5.sh` script from your terminal session. Observe your screen.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./run_tracep5.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:33:59 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceP5';

Session altered.

SQL>
SQL> insert into sales2 select * from sh.sales union all select *
from sales;

1837686 rows created.

SQL> commit;

Commit complete.

SQL>
SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceS5';

Session altered.

SQL>
SQL> insert into sales3 select * from sh.sales union all select *
from sales;

1837686 rows created.

SQL> commit;

Commit complete.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
```

```
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @run_tracep5.sql

--------------------------------------------------------------

set echo on

connect trace/trace@TRACESERV

alter session set tracefile_identifier='mytraceP5';

insert into sales2 select * from sh.sales union all select * from
sales;
commit;


connect trace/trace@TRACESERV

alter session set tracefile_identifier='mytraceS5';

insert into sales3 select * from sh.sales union all select * from
sales;
commit;

exit;
```

15) Execute the `run_tracep6.sh` script from your terminal session. Observe your screen.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./run_tracep6.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:34:26 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:34:26 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.
```

Practice 10-1: Tracing Applications (continued)

```
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> connect trace/trace
SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> update sales set amount_sold=amount_sold+1;
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceP6';

Session altered.

SQL>
SQL> exec dbms_lock.sleep(30);

918843 rows updated.

SQL>
SQL> exec dbms_lock.sleep(60);

PL/SQL procedure successfully completed.

SQL>
SQL> set termout off
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Application_Tracing]$
PL/SQL procedure successfully completed.

SQL>
SQL> rollback;

Rollback complete.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
```

Practice 10-1: Tracing Applications (continued)

```
and Real Application Testing options

[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @run_tracep6a.sql &

sqlplus / as sysdba @run_tracep6b.sql

--------------------------------------------------------------


set echo on

connect trace/trace

update sales set amount_sold=amount_sold+1;

exec dbms_lock.sleep(60);

rollback;

exit;

--------------------------------------------------------------

set echo on

connect trace/trace@TRACESERV

alter session set tracefile_identifier='mytraceP6';

exec dbms_lock.sleep(30);

set termout off

select cust_id, sum(amount_sold) from sales group by cust_id order
by cust_id;

set tournout on

exit;
```

Practice 10-1: Tracing Applications (continued)

16) Execute the `run_tracep7.sh` script from your terminal session. Observe your screen.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./run_tracep7.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:36:10 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> connect trace/trace@TRACESERV
Connected.
SQL>
SQL> alter session set tracefile_identifier='mytraceP7';

Session altered.

SQL>
SQL> declare
  2     c number := dbms_sql.open_cursor;
  3     custid  number;
  4     amount  number;
  5     ignore integer;
  6  begin
  7     dbms_sql.parse(c,'select cust_id, sum(amount_sold) from
sales where cust_id=2 group by cust_id order by cust_id' ,
dbms_sql.native); -- use bind var
  8     dbms_sql.define_column(c, 1, custid);
  9     dbms_sql.define_column(c, 2, amount);
 10     ignore := dbms_sql.execute(c);
 11     if dbms_sql.fetch_rows(c)>0 then
 12       dbms_sql.column_value(c, 1, custid);
 13       dbms_sql.column_value(c, 2, amount);
 14     end if;
 15  end;
 16  /

PL/SQL procedure successfully completed.

SQL>
SQL> connect trace/trace
Connected.
SQL>
SQL> create index sales_cust_indx on sales(cust_id);

Index created.

SQL>
SQL> exit;
```

```
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba @run_tracep7.sql

--------------------------------------------------------------

set echo on

connect trace/trace@TRACESERV

alter session set tracefile_identifier='mytraceP7';

declare
   c number := dbms_sql.open_cursor;
   custid  number;
   amount  number;
   ignore integer;
begin
   dbms_sql.parse(c,'select cust_id, sum(amount_sold) from sales
where cust_id=2 group by cust_id order by cust_id' ,
dbms_sql.native); -- use bind var
   dbms_sql.define_column(c, 1, custid);
   dbms_sql.define_column(c, 2, amount);
   ignore := dbms_sql.execute(c);
   if dbms_sql.fetch_rows(c)>0 then
     dbms_sql.column_value(c, 1, custid);
     dbms_sql.column_value(c, 2, amount);
   end if;
end;
/

connect trace/trace

create index sales_cust_indx on sales(cust_id);

exit;
```

17) Disable tracing for your database.

Practice 10-1: Tracing Applications (continued)

a) Go back to the Top Services page in your Enterprise Manager session.

b) Ensure that TRACESERV is selected, and click the Disable SQL trace button.

c) Back to the Top Services page, you should see a confirmation message near the top of the Top Consumers page.

18) Try to find out all the trace files that were generated to handle the previous seven cases.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./show_mytraces.sh
orcl_ora_19237_mytraceP0.trc
orcl_ora_19237_mytraceP0.trm
orcl_ora_19313_mytraceP1.trc
orcl_ora_19313_mytraceP1.trm
orcl_ora_19355_mytraceS1.trc
orcl_ora_19355_mytraceS1.trm
orcl_ora_19382_mytraceP2.trc
orcl_ora_19382_mytraceP2.trm
orcl_ora_19467_mytraceS2.trc
orcl_ora_19467_mytraceS2.trm
orcl_ora_19474_mytraceP3.trc
orcl_ora_19474_mytraceP3.trm
orcl_ora_19503_mytraceS3.trc
orcl_ora_19503_mytraceS3.trm
orcl_ora_19534_mytraceP4.trc
orcl_ora_19534_mytraceP4.trm
orcl_ora_19549_mytraceS4.trc
orcl_ora_19549_mytraceS4.trm
orcl_ora_19558_mytraceP5.trc
orcl_ora_19558_mytraceP5.trm
orcl_ora_19568_mytraceS5.trc
orcl_ora_19568_mytraceS5.trm
orcl_ora_19583_mytraceP6.trc
orcl_ora_19583_mytraceP6.trm
orcl_ora_19634_mytraceP7.trc
orcl_ora_19634_mytraceP7.trm
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

ls /u01/app/oracle/diag/rdbms/orcl/orcl/trace | grep mytrace
```

19) After you identify the location of those trace files, merge their content into one file called `mytrace.trc` located in your `$HOME/solutions/Application_Tracing` directory.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./merge_traces.sh
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------
```

```
#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

trcsess output=mytrace.trc service=TRACESERV
/u01/app/oracle/diag/rdbms/orcl/orcl/trace/*.trc
```

20) Use `tkprof` over the `mytrace.trc` file to generate a compiled trace output called `myreport.txt` located into your `$HOME/solutions/Application_Tracing` directory.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./tkprof_traces.sh

TKPROF: Release 11.1.0.6.0 - Production on Fri Apr 4 20:37:27 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


[oracle@edrsr33p1-orcl Application_Tracing]$
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

tkprof mytrace.trc myreport.txt
```

21) In addition, run `TKPROF` over the trace file that was generated for case 7 (step 16) with the `EXPLAIN` option set to your `TRACE` account.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ tkprof
/u01/app/oracle/diag/rdbms/orcl/orcl/trace/*mytraceP7.trc
myreport2.txt explain=trace/trace

TKPROF: Release 11.1.0.6.0 - Production on Fri Apr 4 20:40:22 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.
```

```
[oracle@edrsr33p1-orcl Application_Tracing]$
[oracle@edrsr33p1-orcl Application_Tracing]$
```

22) After this is done, interpret the trace generated for case 1 (step 10). What do you observe, and what are your conclusions?

   a) This case tries to illustrate the consequences of using manually sized SQL area parameters, such as `HASH_AREA_SIZE`. The first statement was executed using a very small `HASH_AREA_SIZE` value. The immediate consequence was the number of temporary segments needed to execute the statement. Later, the same statement was executed, but this time using the default SQL area parameters, which were sized automatically by the system to handle the needs. You can see a high disk value as well as a high number of waits for temporary segments for the first execution, compared to the second one.

```
select /*+ ORDERED USE_HASH(s2) */ count(*)
from
 sales s1, sales s2 where s1.cust_id=s2.cust_id


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.00       0.00          0          2          0
0
Execute      1      0.00       0.00          0          0          0
0
Fetch        2     78.23      79.16     806500       8874          0
1
------- ------  -------- ---------- ---------- ---------- ----------
----------
total        4     78.23      79.16     806500       8876          0
1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=8874 pr=806500 pw=806500 time=0 us)
172878975   HASH JOIN  (cr=8874 pr=806500 pw=806500 time=1723284 us
cost=1392520 size=5737160286 card=220660011)
 918843    TABLE ACCESS FULL SALES (cr=4437 pr=4433 pw=4433
time=7413 us cost=1363 size=14132716 card=1087132)
 918843    TABLE ACCESS FULL SALES (cr=4437 pr=4433 pw=4433
time=7485 us cost=1363 size=14132716 card=1087132)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait
Total Waited
```

Practice 10-1: Tracing Applications (continued)

```
  ----------------------------------------   Waited  ----------  ---
---------
  SQL*Net message to client                          2        0.00
0.00
  reliable message                                   1        0.00
0.00
  enq: KO - fast object checkpoint                   1        0.00
0.00
  direct path read                                  86        0.00
0.00
  direct path write temp                          2524        0.00
0.00
  direct path read temp                         797634        0.00
1.70
  SQL*Net message from client                        2        0.00
0.00
********************************************************************
************




select /*+ ORDERED USE_HASH(s2) S1 */ count(*)
from
 sales s1, sales s2 where s1.cust_id=s2.cust_id


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.00       0.00          0          2          0
0
Execute      1      0.00       0.00          0          0          0
0
Fetch        2     40.02      40.14      10028       8874          0
1
------- ------  -------- ---------- ---------- ---------- ----------
----------
total        4     40.03      40.14      10028       8876          0
1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT AGGREGATE (cr=8874 pr=10028 pw=10028 time=0 us)
172878975   HASH JOIN  (cr=8874 pr=10028 pw=10028 time=1430806 us
cost=6891 size=5737160286 card=220660011)
 918843    TABLE ACCESS FULL SALES (cr=4437 pr=4433 pw=4433
time=6837 us cost=1363 size=14132716 card=1087132)
 918843    TABLE ACCESS FULL SALES (cr=4437 pr=4433 pw=4433
time=7084 us cost=1363 size=14132716 card=1087132)
```

```
Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait
Total Waited
  ----------------------------------------   Waited  ----------  ---
---------
  SQL*Net message to client                       2        0.00
0.00
  direct path read                               86        0.00
0.00
  direct path write temp                        166        0.00
0.00
  direct path read temp                         166        0.00
0.00
  SQL*Net message from client                     2        0.00
0.00
*********************************************************************
************
```

23) Interpret the trace generated for case 2 (step 11). What do you observe, and what are your conclusions?

a) For this case, the almost same statement was run 5000 times, but each time a different literal was used to execute the query. This caused the system to parse almost the same statement 5000 times, which is extremely inefficient although the time precision is too low to give accurate information about the parse time of each statement. However, another statement was also executed 5000 times, but this time using a bind variable. The statement was parsed only once, and executed 5000 times. This behavior is much more efficient than the previous one.

```
select object_name
from
 dba_objects where object_id = 1


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.01       0.01          0          0          0
0
Execute      1      0.00       0.00          0          0          0
0
Fetch        1      0.00       0.00          0          3          0
0
------- ------  -------- ---------- ---------- ---------- ----------
----------
total        3      0.01       0.01          0          3          0
0

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182     (recursive depth: 1)

Rows     Row Source Operation
```

```
-------  ---------------------------------------------------
      0  VIEW  DBA_OBJECTS (cr=3 pr=0 pw=0 time=0 us cost=5 size=158
card=2)
      0   UNION-ALL  (cr=3 pr=0 pw=0 time=0 us)
      0    FILTER  (cr=3 pr=0 pw=0 time=0 us)
      0     NESTED LOOPS  (cr=3 pr=0 pw=0 time=0 us cost=5 size=109
card=1)
      0      NESTED LOOPS  (cr=3 pr=0 pw=0 time=0 us cost=4 size=105
card=1)
      0       TABLE ACCESS BY INDEX ROWID OBJ$ (cr=3 pr=0 pw=0
time=0 us cost=3 size=82 card=1)
      1        INDEX RANGE SCAN I_OBJ1 (cr=2 pr=0 pw=0 time=0 us
cost=2 size=0 card=1)(object id 36)
      0       INDEX RANGE SCAN I_USER2 (cr=0 pr=0 pw=0 time=0 us
cost=1 size=23 card=1)(object id 47)
      0      INDEX RANGE SCAN I_USER2 (cr=0 pr=0 pw=0 time=0 us
cost=1 size=4 card=1)(object id 47)
      0     TABLE ACCESS BY INDEX ROWID IND$ (cr=0 pr=0 pw=0 time=0
us cost=2 size=8 card=1)
      0      INDEX UNIQUE SCAN I_IND1 (cr=0 pr=0 pw=0 time=0 us
cost=1 size=0 card=1)(object id 41)
      0     NESTED LOOPS  (cr=0 pr=0 pw=0 time=0 us cost=2 size=28
card=1)
      0      INDEX FULL SCAN I_USER2 (cr=0 pr=0 pw=0 time=0 us
cost=1 size=20 card=1)(object id 47)
      0      INDEX RANGE SCAN I_OBJ4 (cr=0 pr=0 pw=0 time=0 us
cost=1 size=8 card=1)(object id 39)
      0    FILTER  (cr=0 pr=0 pw=0 time=0 us)
      0     NESTED LOOPS  (cr=0 pr=0 pw=0 time=0 us cost=1 size=83
card=1)
      0      INDEX FULL SCAN I_LINK1 (cr=0 pr=0 pw=0 time=0 us
cost=0 size=79 card=1)(object id 134)
      0      INDEX RANGE SCAN I_USER2 (cr=0 pr=0 pw=0 time=0 us
cost=1 size=4 card=1)(object id 47)

**********************************************************************
************


select object_name
from
 dba_objects where object_id = 2


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.00       0.00          0          0          0
0
Execute      1      0.00       0.00          0          0          0
0
Fetch        1      0.00       0.00          0          5          0
1
------- ------  -------- ---------- ---------- ---------- ----------
----------
```

Practice 10-1: Tracing Applications (continued)

```
total        3      0.00       0.00          0          5          0
1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182     (recursive depth: 1)


...

select object_name
from
 dba_objects where object_id = 5000


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.00       0.00          0          0          0
0
Execute      1      0.00       0.00          0          0          0
0
Fetch        1      0.00       0.00          0          5          0
1
------- ------  -------- ---------- ---------- ---------- ----------
----------
total        3      0.00       0.00          0          5          0
1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182     (recursive depth: 1)

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  VIEW  DBA_OBJECTS (cr=5 pr=0 pw=0 time=0 us cost=5 size=158
card=2)
      1   UNION-ALL  (cr=5 pr=0 pw=0 time=0 us)
      1    FILTER  (cr=5 pr=0 pw=0 time=0 us)
      1     NESTED LOOPS  (cr=5 pr=0 pw=0 time=0 us cost=5 size=109
card=1)
      1      NESTED LOOPS  (cr=4 pr=0 pw=0 time=0 us cost=4 size=105
card=1)
      1       TABLE ACCESS BY INDEX ROWID OBJ$ (cr=3 pr=0 pw=0
time=0 us cost=3 size=82 card=1)
      1        INDEX RANGE SCAN I_OBJ1 (cr=2 pr=0 pw=0 time=0 us
cost=2 size=0 card=1)(object id 36)
      1       INDEX RANGE SCAN I_USER2 (cr=1 pr=0 pw=0 time=0 us
cost=1 size=23 card=1)(object id 47)
      1      INDEX RANGE SCAN I_USER2 (cr=1 pr=0 pw=0 time=0 us
cost=1 size=4 card=1)(object id 47)
      0     TABLE ACCESS BY INDEX ROWID IND$ (cr=0 pr=0 pw=0 time=0
us cost=2 size=8 card=1)
      0      INDEX UNIQUE SCAN I_IND1 (cr=0 pr=0 pw=0 time=0 us
cost=1 size=0 card=1)(object id 41)
```

Practice 10-1: Tracing Applications (continued)

```
      0     NESTED LOOPS  (cr=0 pr=0 pw=0 time=0 us cost=2 size=28 
card=1)
      0      INDEX FULL SCAN I_USER2 (cr=0 pr=0 pw=0 time=0 us 
cost=1 size=20 card=1)(object id 47)
      0      INDEX RANGE SCAN I_OBJ4 (cr=0 pr=0 pw=0 time=0 us 
cost=1 size=8 card=1)(object id 39)
      0    FILTER  (cr=0 pr=0 pw=0 time=0 us)
      0     NESTED LOOPS  (cr=0 pr=0 pw=0 time=0 us cost=1 size=83 
card=1)
      0      INDEX FULL SCAN I_LINK1 (cr=0 pr=0 pw=0 time=0 us 
cost=0 size=79 card=1)(object id 134)
      0      INDEX RANGE SCAN I_USER2 (cr=0 pr=0 pw=0 time=0 us 
cost=1 size=4 card=1)(object id 47)

********************************************************************
************







select object_name
from
 dba_objects where object_id = :y


call     count       cpu    elapsed       disk      query    current 
rows
------- ------  -------- ---------- ---------- ---------- ---------- 
----------
Parse        1      0.00       0.00          0          0          0 
0
Execute   5000      0.20       0.18          0          0          0 
0
Fetch     5000      0.24       0.24          0      26837          0 
4928
------- ------  -------- ---------- ---------- ---------- ---------- 
----------
total    10001      0.45       0.43          0      26837          0 
4928

Misses in library cache during parse: 1
Misses in library cache during execute: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182     (recursive depth: 1)

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  VIEW  DBA_OBJECTS (cr=3 pr=0 pw=0 time=0 us cost=5 size=158 
card=2)
      0   UNION-ALL  (cr=3 pr=0 pw=0 time=0 us)
      0    FILTER  (cr=3 pr=0 pw=0 time=0 us)
      0     NESTED LOOPS  (cr=3 pr=0 pw=0 time=0 us cost=5 size=109 
card=1)
```

Practice 10-1: Tracing Applications (continued)

```
      0      NESTED LOOPS  (cr=3 pr=0 pw=0 time=0 us cost=4 size=105 
card=1)
      0       TABLE ACCESS BY INDEX ROWID OBJ$ (cr=3 pr=0 pw=0 
time=0 us cost=3 size=82 card=1)
      1        INDEX RANGE SCAN I_OBJ1 (cr=2 pr=0 pw=0 time=0 us 
cost=2 size=0 card=1)(object id 36)
      0       INDEX RANGE SCAN I_USER2 (cr=0 pr=0 pw=0 time=0 us 
cost=1 size=23 card=1)(object id 47)
      0      INDEX RANGE SCAN I_USER2 (cr=0 pr=0 pw=0 time=0 us 
cost=1 size=4 card=1)(object id 47)
      0     TABLE ACCESS BY INDEX ROWID IND$ (cr=0 pr=0 pw=0 time=0 
us cost=2 size=8 card=1)
      0      INDEX UNIQUE SCAN I_IND1 (cr=0 pr=0 pw=0 time=0 us 
cost=1 size=0 card=1)(object id 41)
      0     NESTED LOOPS  (cr=0 pr=0 pw=0 time=0 us cost=2 size=28 
card=1)
      0      INDEX FULL SCAN I_USER2 (cr=0 pr=0 pw=0 time=0 us 
cost=1 size=20 card=1)(object id 47)
      0      INDEX RANGE SCAN I_OBJ4 (cr=0 pr=0 pw=0 time=0 us 
cost=1 size=8 card=1)(object id 39)
      0    FILTER  (cr=0 pr=0 pw=0 time=0 us)
      0     NESTED LOOPS  (cr=0 pr=0 pw=0 time=0 us cost=1 size=83 
card=1)
      0      INDEX FULL SCAN I_LINK1 (cr=0 pr=0 pw=0 time=0 us 
cost=0 size=79 card=1)(object id 134)
      0      INDEX RANGE SCAN I_USER2 (cr=0 pr=0 pw=0 time=0 us 
cost=1 size=4 card=1)(object id 47)

********************************************************************
************
```

24) Interpret the trace generated for case 3 (step 12). What do you observe, and what are your conclusions?

a) If you look closely at this case, you see that you access the complete table to update only two rows. This is very inefficient and the alternate case is much better as it uses an index to speed up the retrieval of the rows that need to be updated.

```
update sales set amount_sold=20000
where
 prod_id=13 and cust_id=987


call     count       cpu    elapsed       disk      query    current 
rows
------- ------  -------- ---------- ---------- ---------- ---------- 
----------
Parse        1      0.00       0.00          0          1          0 
0
Execute      1      0.09       0.09       2442       4441          4 
2
Fetch        0      0.00       0.00          0          0          0 
0
------- ------  -------- ---------- ---------- ---------- ---------- 
----------
total        2      0.09       0.10       2442       4442          4 
2
```

Practice 10-1: Tracing Applications (continued)

```
Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  UPDATE  SALES (cr=4441 pr=2442 pw=2442 time=0 us)
      2   TABLE ACCESS FULL SALES (cr=4441 pr=2442 pw=2442 time=17
us cost=1366 size=4251 card=109)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait
Total Waited
  ----------------------------------------   Waited  ----------  ---
---------
  db file scattered read                         63        0.00
0.01
  db file sequential read                         3        0.00
0.00
  SQL*Net message to client                       1        0.00
0.00
  SQL*Net message from client                     1        0.00
0.00
********************************************************************
************





update sales set amount_sold=30000
where
 prod_id=13 and cust_id=987


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.00       0.00          0          2          0
0
Execute      1      0.00       0.00          0          3          3
2
Fetch        0      0.00       0.00          0          0          0
0
------- ------  -------- ---------- ---------- ---------- ----------
----------
total        2      0.00       0.00          0          5          3
2

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182

Rows     Row Source Operation
```

Practice 10-1: Tracing Applications (continued)

```
-------  ---------------------------------------------------
      0  UPDATE  SALES (cr=3 pr=0 pw=0 time=0 us)
      2   INDEX RANGE SCAN SALES_PROD_CUST_INDX (cr=3 pr=0 pw=0
time=2 us cost=3 size=78 card=2)(object id 75372)


Elapsed times include waiting on following events:
  Event waited on                            Times   Max. Wait
Total Waited
  ----------------------------------------   Waited  ----------  ---
---------
  SQL*Net message to client                       1        0.00
0.00
  SQL*Net message from client                     1        0.00
0.00
********************************************************************
************
```

25) Interpret the trace generated for case 4 (step 13). What do you observe, and what are your conclusions?

   a) In this case, the first statement does not use the fetching mechanism appropriately. One fetch operation is done for every single row retrieved. This is also very inefficient. The alternate case is doing 92 fetches to retrieve the same amount of rows. This technique is called array fetch.

```
select *
from
 sh.sales where amount_sold>0


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.02       0.02          0          0          0
0
Execute      1      0.00       0.00          0          0          0
0
Fetch   918844      5.89       5.93          0     918944          0
918843
------- ------  -------- ---------- ---------- ---------- ----------
----------
total   918846      5.92       5.95          0     918944          0
918843

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182     (recursive depth: 1)

Rows     Row Source Operation
-------  ---------------------------------------------------
 918843  PARTITION RANGE ALL PARTITION: 1 28 (cr=918944 pr=0 pw=0
time=41095 us cost=547 size=26646447 card=918843)
 918843   TABLE ACCESS FULL SALES PARTITION: 1 28 (cr=918944 pr=0
pw=0 time=27743 us cost=547 size=26646447 card=918843)
```

Practice 10-1: Tracing Applications (continued)

```
********************************************************************
************




select /* S4 */ *
from
 sh.sales where amount_sold>0


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.00       0.00          0          0          0
0
Execute      1      0.00       0.00          0          0          0
0
Fetch       92      2.06       2.06          0       1811          0
918843
------- ------  -------- ---------- ---------- ---------- ----------
----------
total       94      2.06       2.06          0       1811          0
918843

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182     (recursive depth: 1)

Rows     Row Source Operation
-------  ---------------------------------------------------
 918843  PARTITION RANGE ALL PARTITION: 1 28 (cr=1811 pr=0 pw=0
time=21132 us cost=547 size=26646447 card=918843)
 918843   TABLE ACCESS FULL SALES PARTITION: 1 28 (cr=1811 pr=0 pw=0
time=8617 us cost=547 size=26646447 card=918843)

********************************************************************
************
```

26) Interpret the trace generated for case 5 (step 14). What do you observe, and what are your conclusions?

a) Here, the first statement incurs too many recursive calls to allocate extents to the table. This is because the tablespace in which it is stored is not correctly set up for extent allocations. The alternate case is much better as you can see the number of disk values going way down compared to the first case. Also the number of recursive statements in the second case should be much lower than the first one.

```
insert into sales2 select * from sh.sales union all select * from
sales


call     count       cpu    elapsed       disk      query    current
rows
```

```
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.00       0.01          0          2          0
0
Execute      1      5.02       5.41       4432      28134     148866
1837686
Fetch        0      0.00       0.00          0          0          0
0
------- ------  -------- ---------- ---------- ---------- ----------
----------
total        2      5.02       5.42       4432      28136     148866
1837686

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182

Rows     Row Source Operation
-------  ---------------------------------------------------
      0  LOAD TABLE CONVENTIONAL  (cr=44323 pr=4433 pw=4433 time=0
us)
1837686   UNION-ALL  (cr=6160 pr=4432 pw=4432 time=58107 us)
 918843    PARTITION RANGE ALL PARTITION: 1 28 (cr=1719 pr=0 pw=0
time=21204 us cost=545 size=26646447 card=918843)
 918843     TABLE ACCESS FULL SALES PARTITION: 1 28 (cr=1719 pr=0
pw=0 time=8395 us cost=545 size=26646447 card=918843)
 918843    TABLE ACCESS FULL SALES (cr=4441 pr=4432 pw=4432
time=6976 us cost=1369 size=94580484 card=1087132)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait
Total Waited
  ----------------------------------------   Waited  ----------  ---
---------
  db file scattered read                         51        0.00
0.03
  log file switch completion                      2        0.00
0.00
  log buffer space                                8        0.09
0.23
  SQL*Net message to client                       1        0.00
0.00
  SQL*Net message from client                     1        0.00
0.00
********************************************************************
************

select file#
from
 file$ where ts#=:1


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
```

Practice 10-1: Tracing Applications (continued)

```
Parse     1812      0.02       0.02          0          0          0
0
Execute   1812      0.02       0.03          0          0          0
0
Fetch     3624      0.05       0.05          0       7242          0
1812
------- ------  -------- ---------- ---------- ---------- ----------
----------
total     7248      0.11       0.10          0       7242          0
1812

Misses in library cache during parse: 1
Misses in library cache during execute: 1
Optimizer mode: CHOOSE
Parsing user id: SYS   (recursive depth: 1)

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  TABLE ACCESS FULL FILE$ (cr=4 pr=0 pw=0 time=0 us cost=2
size=6 card=1)

********************************************************************
************










insert into sales3 select * from sh.sales union all select * from
sales


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.00       0.00          0          1          0
0
Execute      1      3.64       5.13       1087      22471      77748
1837686
Fetch        0      0.00       0.00          0          0          0
0
------- ------  -------- ---------- ---------- ---------- ----------
----------
total        2      3.65       5.14       1087      22472      77748
1837686

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182

Rows     Row Source Operation
```

Practice 10-1: Tracing Applications (continued)

```
-------  ---------------------------------------------------
      0  LOAD TABLE CONVENTIONAL  (cr=22544 pr=1087 pw=1087 time=0
us)
1837686   UNION-ALL  (cr=6160 pr=1087 pw=1087 time=58277 us)
 918843    PARTITION RANGE ALL PARTITION: 1 28 (cr=1719 pr=0 pw=0
time=21371 us cost=545 size=26646447 card=918843)
 918843     TABLE ACCESS FULL SALES PARTITION: 1 28 (cr=1719 pr=0
pw=0 time=8407 us cost=545 size=26646447 card=918843)
 918843    TABLE ACCESS FULL SALES (cr=4441 pr=1087 pw=1087
time=7141 us cost=1369 size=94580484 card=1087132)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait
Total Waited
  ----------------------------------------   Waited  ----------  ---
---------
  buffer busy waits                                3        0.00
0.00
  undo segment extension                           1        0.00
0.00
  log file switch completion                       5        0.08
0.12
  log buffer space                                30        0.15
1.24
  db file scattered read                          25        0.00
0.00
  log file switch (private strand flush incomplete)
                                                 313        0.05
0.10
  SQL*Net message to client                        1        0.00
0.00
  SQL*Net message from client                      1        0.00
0.00
********************************************************************
************
```

27) Interpret the trace generated for case 6 (step 15). What do you observe, and what are your conclusions?

   a) This case is more difficult to understand. Here, you select a table that is entirely locked by another transaction. This forces the query to generate consistent read blocks for almost the entire table causing undo segments to be accessed. This is materialized by an important disk value, and almost no current blocks.

```
select cust_id, sum(amount_sold)
from
 sales group by cust_id order by cust_id


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.00       0.00          0          1          0
0
```

```
Execute      1      0.00       0.00          0          0          0
0
Fetch      472      2.38       2.39       5333     978282          0
7059
------- ------  -------- ---------- ---------- ---------- ----------
----------
total      474      2.39       2.39       5333     978283          0
7059

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182

Rows     Row Source Operation
-------  ---------------------------------------------------
   7059  SORT GROUP BY (cr=978282 pr=5333 pw=5333 time=64 us
cost=1422 size=28265432 card=1087132)
 918843   TABLE ACCESS FULL SALES (cr=978282 pr=5333 pw=5333
time=17979 us cost=1369 size=28265432 card=1087132)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait
Total Waited
  ----------------------------------------   Waited  ----------  ---
---------
  SQL*Net message to client                     472        0.00
0.00
  db file sequential read                      3827        0.00
0.03
  db file scattered read                         37        0.00
0.01
  latch: cache buffers lru chain                  1        0.00
0.00
  SQL*Net message from client                   472        0.00
0.02
********************************************************************
************
```

28) Interpret the trace generated for case 7 (step 16). What do you observe, and what are your conclusions?

   a) For case 7, you should compare the content in `myreport.txt` with the content in `myreport2.txt`. You should see that an index was added after the first trace was generated. This situation can cause confusion especially if the trace does not contain an execution plan to begin with. This is what you can see from within `myreport.txt`:

```
select cust_id, sum(amount_sold)
from
 sales where cust_id=2 group by cust_id order by cust_id


call     count       cpu    elapsed       disk      query    current
rows
```

Practice 10-1: Tracing Applications (continued)

```
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.00       0.00          0          1          0
0
Execute      1      0.00       0.00          0          0          0
0
Fetch        1      0.05       0.05          0       4441          0
1
------- ------  -------- ---------- ---------- ---------- ----------
----------
total        3      0.06       0.06          0       4442          0
1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182     (recursive depth: 1)

Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT GROUP BY NOSORT (cr=4441 pr=0 pw=0 time=0 us cost=1366
size=1872 card=72)
    176   TABLE ACCESS FULL SALES (cr=4441 pr=0 pw=0 time=153 us
cost=1366 size=1872 card=72)




********************************************************************
************
```

b) This is what you see from `myreport2.txt`:

```
SQL ID : 51xxq509gnbbc
select cust_id, sum(amount_sold)
from
 sales where cust_id=2 group by cust_id order by cust_id


call     count       cpu    elapsed       disk      query    current
rows
------- ------  -------- ---------- ---------- ---------- ----------
----------
Parse        1      0.00       0.00          0          1          0
0
Execute      2      0.00       0.00          0          0          0
0
Fetch        1      0.05       0.05          0       4441          0
1
------- ------  -------- ---------- ---------- ---------- ----------
----------
total        4      0.06       0.06          0       4442          0
1

Misses in library cache during parse: 1
Optimizer mode: ALL_ROWS
Parsing user id: 182  (TRACE)   (recursive depth: 1)
```

```
Rows     Row Source Operation
-------  ---------------------------------------------------
      1  SORT GROUP BY NOSORT (cr=4441 pr=0 pw=0 time=0 us cost=1366
size=1872 card=72)
    176   TABLE ACCESS FULL SALES (cr=4441 pr=0 pw=0 time=153 us
cost=1366 size=1872 card=72)


Rows     Execution Plan
-------  ---------------------------------------------------
      0  SELECT STATEMENT   MODE: ALL_ROWS
      1   SORT (GROUP BY NOSORT)
    176    TABLE ACCESS (BY INDEX ROWID) OF 'SALES' (TABLE)
      0     INDEX   MODE: ANALYZED (RANGE SCAN) OF 'SALES_CUST_INDX'
                (INDEX)


Elapsed times include waiting on following events:
  Event waited on                             Times   Max. Wait
Total Waited
  ----------------------------------------   Waited  ----------  ---
---------
  SQL*Net message to client                       1        0.00
0.00
  SQL*Net message from client                     1        0.00
0.00
********************************************************************
************
```

29) You can now clean up your environment by executing the `at_cleanup.sh` script from your terminal window.

```
[oracle@edrsr33p1-orcl Application_Tracing]$ ./at_cleanup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Fri Apr 4 20:45:24 2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL>
SQL> drop user trace cascade;

User dropped.

SQL>
SQL> drop tablespace tracetbs including contents and datafiles;

Tablespace dropped.

SQL>
SQL> drop tablespace tracetbs3 including contents and datafiles;
```

Practice 10-1: Tracing Applications (continued)

```
Tablespace dropped.

SQL>
SQL> exit;
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Application_Tracing]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Application_Tracing

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

rm /u01/app/oracle/diag/rdbms/orcl/orcl/trace/*mytrace*.*

rm mytrace.trc

rm myreport.txt

rm myreport2.txt

rm $ORACLE_HOME/network/admin/tnsnames.ora

mv $ORACLE_HOME/network/admin/tnsnames.ora.bak1
$ORACLE_HOME/network/admin/tnsnames.ora

sqlplus / as sysdba @at_cleanup.sql

--------------------------------------------------------------

set echo on

drop user trace cascade;

drop tablespace tracetbs including contents and datafiles;

drop tablespace tracetbs3 including contents and datafiles;

exit;
```

Practices for Lesson 11

Practice 11-1: Proactively Tuning High-Load SQL Statements

In this practice, you use the SQL Tuning Advisor to help you tune problematic SQL statements.

1) Connect as `SYSDBA` through Database Control and navigate to the Performance tab of the Database Control Home page. On the Performance tabbed page, make sure that the View Data field is set to Real Time: 15 second Refresh. After this is done, open a terminal emulator window connected as the `oracle` user. When this is done, change your current directory to your lab directory: `cd $HOME/solutions/SQL_Tuning_Advisor`. Then enter the following command from the OS prompt: `./setup_dina.sh`

```
[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$ ./setup_dina.sh

[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$ ./setup_dina.sh

PL/SQL procedure successfully completed.


Grant succeeded.


Session altered.


PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.


User altered.


User altered.


Index dropped.

drop index sales_time_idx
           *
ERROR at line 1:
ORA-01418: specified index does not exist



Index created.

[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$


--------------------------------------------------------------

#!/bin/bash
```

```
cd /home/oracle/solutions/SQL_Tuning_Advisor

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus -s /NOLOG <<EOF

set echo on

connect / as sysdba

exec DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT();

grant dba to SH;

rem -- event to allow setting very short Flushing interval

alter session set events '13508 trace name context forever, level
1';


rem -- change INTERVAL setting to 2 minutes
rem -- change RETENTION setting to 6 hours (total of 180 snapshots)
execute dbms_workload_repository.modify_snapshot_settings(interval
=> 2,retention => 360);


rem -- play with ADDM sensitiveness
exec
dbms_advisor.set_default_task_parameter('ADDM','DB_ACTIVITY_MIN',30)
;


alter user sh account unlock;
alter user sh identified by sh;

connect sh/sh

drop index sales_time_bix;
drop index sales_time_idx;
create index sales_time_idx on sales(time_id) compute statistics;


EOF
```

2) After this is executed, execute the `start_dinas.sh` script by using the following command: `./start_dinas.sh`
   This script starts the workload used for this lab.

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

```
[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$ ./start_dinas.sh
Started stream with pid=30479
Started stream with pid=30480
Started stream with pid=30481
Started stream with pid=30482
Started stream with pid=30485
Started stream with pid=30486
[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/SQL_Tuning_Advisor

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin


STREAM_NUM=0
MAX_STREAM=6
PIDLST=""

while [ $STREAM_NUM -lt $MAX_STREAM ]; do

  # one more
  let STREAM_NUM="STREAM_NUM+1"

  # start one more stream
  sqlplus -S sh/sh @dina.sql &

  # remember PID
  PIDLST="$! $PIDLST"

  echo "Started stream with pid=$!"

done

#
# Save PID List
#
echo $PIDLST > /tmp/dina_pids

--------------------------------------------------------------

DECLARE
 n number;
BEGIN
for i in 1..1000 loop
 select /*+ ORDERED USE_NL(c) FULL(c) FULL(s)*/ count(*) into n
 from sales s, customers c
```

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

```
 where c.cust_id = s.cust_id and CUST_FIRST_NAME='Dina'
 order by time_id;
end loop;
END;
/
```

3) When the `start_dinas.sh` script completes, observe the Performance tabbed page for around 10 minutes (5 snapshot icons). What are your conclusions?
   a) You should see that the workload activity goes up very quickly. Because the CPU used by the workload is very close to the maximum CPU available on your system, there must be an issue with this workload. Because the most important area corresponding to a wait class is the Other wait class, the issue must be associated to that class. Note that the snapshot interval is now around two minutes.
4) Fix the problem.
   a) The fastest way to determine the problem is by looking at an Automatic Database Diagnostic Monitor (ADDM) report analysis executed during the problematic period. Then, by following its analysis, ADDM should guide you through the process of fixing the problem.
   b) Using the Database Control Home page, there are two different ways to identify the correct ADDM analysis task:
      i) If the time corresponding to the problematic time period corresponds with the latest ADDM run detected by Database Control, you should find the link corresponding to the correct performance analysis directly in the Diagnostic Summary section of the Database Control Home page. Note that you should wait around 8 to 10 minutes before the Diagnostic Summary section is refreshed with the correct ADDM analysis. If you are in this case, click the link corresponding to the number of findings right next to the ADDM Findings row. This takes you to the corresponding Automatic Database Diagnostic Monitor (ADDM) page.
      ii) If not, you should open the Advisor Central page and search for the correct ADDM task. This is how you can retrieve the task from the Advisor Central page:
         (1) On the Database Control Home page, click the Advisor Central link.
         (2) On the Advisor Central page, in the search section, select ADDM from the Advisory Type drop-down list, and Last 24 Hours from the Advisor Runs drop-down list.
         (3) After this is done, click Go.
         (4) Then select the ADDM task corresponding to the time of the problematic period.

(5) This takes you to the corresponding Automatic Database Diagnostic Monitor (ADDM) page.

c) On the Automatic Database Diagnostic Monitor (ADDM) page, you should see two main findings: CPU Usage and Top SQL by DB Time. Both should be close to 100%. If they are not, ensure that what you see is the latest ADDM analysis.

d) Click the "Top SQL by DB Time" link.

e) On the "Performance Finding Details: Top SQL by DB Time" page, click Show All Details link. You should see something similar to the following: Run SQL Tuning Advisor on the SQL statement with SQL_ID "5mxdwvuf9j3vp". Next to this recommendation, click the Run Advisor Now button.

f) Wait on the Processing: SQL Tuning Advisor Task SQL_TUNING_… page for a while.

g) You are automatically directed to the Recommendations for SQL ID:5mxdwvuf9j3vp" page from where you should see two recommendations-one for a SQL Profile and one for an index creation.

h) You can investigate the consequence of implementing the recommended profile by comparing execution plans before and after profile implementation. You can do so by clicking the "Compare explain plan" eyeglass icon for the corresponding SQL Profile row. Because the potential benefit of using the proposed SQL profile is very high (see both computed Costs), you implement the SQL profile.

i) On the Compare Explain Plans page, click the Recommendations for SQL ID:5mxdwvuf9j3vp locator link.

j) Back to the Recommendations for SQL ID:5mxdwvuf9j3vp page, ensure that the SQL Profile row is selected, and click Implement.

k) On Confirmation page, click Yes.

l) On Recommendations for SQL ID:5mxdwvuf9j3vp page, you should see the following message at its top: "Confirmation: The recommended SQL Profile has been created successfully".

m) Click Return.

n) On SQL Tuning Results:TASK, click the Database Instance: orcl locator link.

o) Back to the Database Home Page, click the Performance tab.

5) After following the SQL Tuning Advisor recommendation to implement a SQL Profile, how can you quickly verify that the problem is solved?

a) On the Performance tabbed page, you expect to see a dramatic drop for the Other wait class in the Average Active Sessions graph. However, when you view the graph you see that not has changed.

6) How do you interpret the result you see, and how would you ensure that the new profile is taken into account?

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

a) A profile is taken into account the next time you execute the corresponding statement. Because the SQL statement for which you created a profile takes a long time to execute, you should wait for a long time to see the benefit. To quickly see the benefit, stop and restart your workload. This executes the same SQL statements again, and the profile should be used automatically this time.

```
[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$ ./stop_dinas.sh
Killing stream with pid=30486
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


Killing stream with pid=30485
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


Killing stream with pid=30482
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


Killing stream with pid=30481
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


Killing stream with pid=30480
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


Killing stream with pid=30479
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed


[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$



[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$ ./start_dinas.sh
Started stream with pid=31731
```

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

```
Started stream with pid=31732
Started stream with pid=31733
Started stream with pid=31734
Started stream with pid=31735
Started stream with pid=31740
[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/SQL_Tuning_Advisor

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin


PIDLST=`cat /tmp/dina_pids`

#
# Kill all these processes
#
for PID in $PIDLST; do
  echo "Killing stream with pid=$PID"
  sqlplus / as sysdba @kill_dina.sql $PID >> /tmp/stop_dina.log 2>&1
  sqlplus /nolog @/tmp/drop_dina.sql >> /tmp/stop_dina.log 2>&1
  kill -9 $PID >> /tmp/stop_dina.log 2>&1
done

--------------------------------------------------------------

set head off
set timing off
set feedback off;
set pagesize 0
set verify off

spool /tmp/drop_dina.sql;

select 'connect / as sysdba;' from dual;

select 'alter system kill session ''' || sid || ',' || serial# ||
''';'
from v$session
where process=&1;

select 'exit;' from dual;

spool off

exit;
```

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

b) Go back to the Performance tabbed page.

c) On the Performance page, you should now see the benefit of the SQL Profile. There is no longer any Other waits in the Average Active Sessions graph.

7) How would you make sure that the SQL Profile was implemented?

a) On the Average Active Sessions graph, click the Other wait category in the caption.

b) On the Active Sessions Waiting: Other page, you should see that your statement waits for the latch free event. You can see that from the `ACTIVITY(%)` column.

c) Click the 5mxdwvuf9j3vp SQL Id link in the Top SQL table.

d) On the SQL Details: 5mxdwvuf9j3vp page, click the Plan Control tab in the Details section.

e) You should see that this statement is associated to a SQL Profile that was manually implemented. It is part of the DEFAULT category and is ENABLED.

8) Clean up your environment by executing the following commands from your command-line window:

```
./stop_dians.sh
./cleanup_dina.sh
```

```
[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$ ./stop_dinas.sh
Killing stream with pid=31740
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed

Killing stream with pid=31735
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed

Killing stream with pid=31734
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed

Killing stream with pid=31733
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed

Killing stream with pid=31732
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed
```

```
Killing stream with pid=31731
DECLARE
*
ERROR at line 1:
ORA-00028: your session has been killed

[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$



[oracle@edrsr33p1-orcl SQL_Tuning_Advisor]$ ./cleanup_dina.sh

Revoke succeeded.

specify password for SH as parameter 1:

specify default tablespace for SH as parameter 2:

specify temporary tablespace for SH as parameter 3:

specify password for SYS as parameter 4:

specify directory path for the data files as parameter 5:

writeable directory path for the log files as parameter 6:

specify version as parameter 7:


Session altered.


User dropped.

old   1: CREATE USER sh IDENTIFIED BY &pass
new   1: CREATE USER sh IDENTIFIED BY sh

User created.

old   1: ALTER USER sh DEFAULT TABLESPACE &tbs
new   1: ALTER USER sh DEFAULT TABLESPACE example
old   2:  QUOTA UNLIMITED ON &tbs
new   2:  QUOTA UNLIMITED ON example

User altered.

old   1: ALTER USER sh TEMPORARY TABLESPACE &ttbs
new   1: ALTER USER sh TEMPORARY TABLESPACE temp

User altered.


Grant succeeded.
```

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

```
Grant succeeded.


...


Grant succeeded.


Grant succeeded.


PL/SQL procedure successfully completed.

Connected.

Grant succeeded.

old   1: CREATE OR REPLACE DIRECTORY data_file_dir AS '&data_dir'
new   1: CREATE OR REPLACE DIRECTORY data_file_dir AS
'/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/'

Directory created.

old   1: CREATE OR REPLACE DIRECTORY log_file_dir AS '&log_dir'
new   1: CREATE OR REPLACE DIRECTORY log_file_dir AS '/home/oracle/'

Directory created.


Grant succeeded.


Grant succeeded.


Grant succeeded.

Connected.

Session altered.


Session altered.


Table created.


Table created.


...


Table created.
```

```
Creating constraints ...

Table altered.


...


Table altered.


specify password for SH as parameter 1:

specify path for data files as parameter 2:

specify path for log files as parameter 3:

specify version as parameter 4:

Looking for indexes that could slow down load ...

no rows selected


loading TIMES using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/time_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/time_v
3.dat
/home/oracle/time_v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:23
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 1000.

Load completed - logical record count 1826.


loading COUNTRIES using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/coun_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/coun_v
3.dat
/home/oracle/coun_v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:24
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.
```

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

```
Load completed - logical record count 23.


loading CUSTOMERS using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/cust_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/cust1v
3.dat
/home/oracle/cust1v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:24
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 10000.
Save data point reached - logical record count 20000.
Save data point reached - logical record count 30000.
Save data point reached - logical record count 40000.
Save data point reached - logical record count 50000.

Load completed - logical record count 55500.


loading PRODUCTS  using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/prod_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/prod1v
3.dat
/home/oracle/prod1v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:24
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Load completed - logical record count 72.


loading PROMOTIONS  using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/prom_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/prom1v
3.dat
/home/oracle/prom1v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:25
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 10.
Save data point reached - logical record count 20.
...
Save data point reached - logical record count 500.
```

```
Load completed - logical record count 503.


loading CHANNELS using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/chan_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/chan_v
3.dat
/home/oracle/chan_v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:25
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Load completed - logical record count 5.


loading SALES  using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/sale_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/sale1v
3.dat
/home/oracle/sale1v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:25
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 100000.
...
Save data point reached - logical record count 900000.

Load completed - logical record count 916039.


loading COSTS using external table


Table created.


82112 rows created.


loading additonal SALES using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/dmsal_
v3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/dmsal_
v3.dat
/home/oracle/dmsal_v3.log
```

```
SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:38
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 100.
...
Save data point reached - logical record count 2800.

Load completed - logical record count 2804.


loading SUPPLEMENTARY DEMOGRAPHICS using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/dem_v3
.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/dem1v3
.dat
/home/oracle/dem1v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:38
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 10.
...
Save data point reached - logical record count 4500.

Load completed - logical record count 4500.


Commit complete.


Enabling constraints ...

Table altered.


...


Table altered.


Creating additional indexes ...

Index created.


...


Index created.
```

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

```
Create dimensions ...
Dimension created.


Commit complete.


PL/SQL procedure successfully completed.


no rows selected


Dimension created.


PL/SQL procedure successfully completed.


no rows selected


Dimension created.


PL/SQL procedure successfully completed.


no rows selected


Dimension created.


PL/SQL procedure successfully completed.


no rows selected


Dimension created.


PL/SQL procedure successfully completed.


no rows selected
Creating MVs as tables ...


View created.


Table created.
```

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

```
Table created.


Index created.


Index created.


Index created.


Index created.

Creating materialized views ...


Materialized view created.


Materialized view created.


Creating comments ...

Comment created.


...


Comment created.


Creating OLAP metadata ...
<<<<< CREATE CWMLite Metadata for the Sales History Schema >>>>>
-
<<<<< CREATE CATALOG sh_cat for Sales History >>>>>
        Catalog Dropped
        CWM Collect Garbage
-
<<<<< CREATE the Sales CUBE >>>>>
        Sales amount, Sales quantity
        <TIMES CHANNELS PRODUCTS CUSTOMERS PROMOTIONS >
        Drop SALES_CUBE prior to recreation
        Cube Dropped
        Add dimensions -
         to SALES_CUBE and map the foreign keys
        Create measures -
         for SALES_CUBE and map to columns in the fact table
        Set default aggregation method -
         to SUM for all measures over TIME
        Add SALES_CUBE to the catalog
        SALES_CUBE successfully added to sh_cat
```

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

```
-
<<<<< CREATE the Cost CUBE >>>>>
        Unit Cost, Unit Price < TIMES PRODUCTS CHANNELS PROMOTIONS >
        Drop COST_CUBE prior to recreation
        Cube Dropped
        Add dimensions -
         to COST_CUBE and map the foreign keys
        Create measures -
         for COST_CUBE and map to columns in the fact table
        Set default aggregation method -
         to SUM for all measures over TIME
        Add COST_CUBE to the catalog
        COST_CUBE successfully added to sh_cat
-
<<<<< TIME DIMENSION >>>>>
Dimension - display name, description and plural name
Level - display name and description
Hierarchy - display name and description
        - default calculation hierarchy
        - default display hierarchy
Level Attributes - name, display name, description
Drop dimension attributes prior to re-creation
Create dimension attributes and add their level attributes
        - Long Description created
        - Short Description created
        - Period Number of Days created
        - Period End Date created
Classify entity descriptor use
        - Time dimension
        - Long description
        - Day name
        - Calendar month description
        - Calendar quarter description
        - Fiscal month description
        - Fiscal quarter description
        - Short Description
        - Day name
        - Calendar month description
        - Calendar quarter description
        - Fiscal month description
        - Fiscal quarter description
        - Time Span
        - Days in calendar month
        - Days in calendar quarter
        - Days in calendar year
        - Days in fiscal month
        - Days in fiscal quarter
        - Days in fiscal year
        - End Date
        - End of calendar month
        - End of calendar quarter
        - End of calendar year
        - End of fiscal month
        - End of fiscal quarter
        - End of fiscal year
-
```

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

```
<<<<< CUSTOMERS DIMENSION >>>>>
Dimension - display name, description and plural name
Level - display name and description
Hierarchy - display name and description
        - default calculation hierarchy
        - default display hierarchy
Level Attributes - name, display name, description
Drop dimension attributes prior to re-creation
No attribute to drop
No attribute to drop
No attribute to drop
No attribute to drop
No attribute to drop
No attribute to drop
No attribute to drop
        No attribute to drop
No attribute to drop
No attribute to drop
        No attribute to drop
        No attribute to drop
Create dimension attributes and add their level attributes
        - Long Description created
        - Short Description created
        - Other Customer Information created
Classify entity descriptor use
        - Long Description
        - Short Description
-
<<<<< PRODUCTS DIMENSION >>>>>
Dimension - display name, description and plural name
Level - display name and description
Hierarchy - display name and description
        - default calculation hierarchy
        - default display hierarchy
Level Attributes - name, display name, description
Drop dimension attributes prior to re-creation
No attribute to drop
Create dimension attributes and add their level attributes
        - Long Description created
        - Short Description created
Classify entity descriptor use
        - Long Description
        - Short Description
-
<<<<< PROMOTIONS DIMENSION >>>>>
Dimension - display name, description and plural name
Level - display name and description
Hierarchy - display name and description
        - default calculation hierarchy
        - default display hierarchy
Level Attributes - name, display name, description
Drop dimension attributes prior to re-creation
No attribute to drop
Create dimension attributes and add their level attributes
        - Long Description created
        - Short Description created
```

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

```
Classify entity descriptor use
        - Long Description
        - Short Description
-
<<<<< CHANNELS DIMENSION >>>>>
Dimension - display name, description and plural name
Level - display name and description
Hierarchy - display name and description
        - default calculation hierarchy
        - default display hierarchy
Level Attributes - name, display name, description
Drop dimension attributes prior to re-creation
No attribute to drop
Create dimension attributes and add their level attributes
        - Long Description created
        - Short Description created
Classify entity descriptor use
        - Long Description
        - Short Description
-
<<<<< FINAL PROCESSING >>>>>
        - Changes have been committed

PL/SQL procedure successfully completed.


Commit complete.


gathering statistics ...

PL/SQL procedure successfully completed.


PL/SQL procedure successfully completed.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl SQL_Access_Advisor]$


--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/SQL_Tuning_Advisor

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin
```

```
#
# Cleanup ADDM snapshot settings
#
sqlplus -s /NOLOG <<EOF >> /tmp/cleanup_dina.log 2>&1

  connect / as sysdba

  rem -- change INTERVAL setting to 30 minute
  execute dbms_workload_repository.modify_snapshot_settings(interval
=> 60);

  rem -- change ADDM sensitiveness back to normal
  exec
dbms_advisor.set_default_task_parameter('ADDM','DB_ACTIVITY_MIN',300
);

  connect sh/sh

  drop index sales_time_idx;

  create bitmap index sales_time_bix
  on sales(time_id)
  tablespace example
  local nologging compute statistics;


EOF

#
# Cleanup sql profile
#
sqlplus -s /NOLOG <<EOF > /tmp/cleanup_dina.log 2>&1

  connect / as sysdba

  set head off
  set timing off
  set feedback off;
  set pagesize 0

spool /tmp/drop_dyn.sql;

select q'#connect / as sysdba;#' from dual;

select q'#execute dbms_sqltune.drop_sql_profile('#' || name || q'#')
;#'
from dba_sql_profiles ;

select q'#execute dbms_advisor.delete_task('#' || task_name || q'#')
;#'
from user_advisor_tasks
where  CREATED > SYSDATE-(1/24);

select q'#connect system/oracle;#' from dual;
```

Practice 11-1: Proactively Tuning High-Load SQL Statements (continued)

```
select q'#execute dbms_advisor.delete_task('#' || task_name || q'#')
;#'
from user_advisor_tasks
where  CREATED > SYSDATE-(1/24);

spool off

@/tmp/drop_dyn.sql

EOF


cp /home/oracle/solutions/SQL_Access_Advisor/sh/*
$ORACLE_HOME/demo/schema/sales_history

cd /home/oracle/solutions/SQL_Access_Advisor/sh

sqlplus -s /NOLOG <<EOF

set echo on

connect / as sysdba

revoke dba from sh;

@sh_main sh example temp oracle
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/
/home/oracle/ v3

EOF
```

Practice 11-2: Using SQL Access Advisor

The following scenario illustrates the types of recommendations that can be made by SQL Access Advisor. The scenario also uses the SQL Performance Analyzer to prove that recommendations made by SQL Access Advisor are good.

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 lab. In particular, it generates the SQL Tuning Set that is used to represent the workload you want to analyze.

```
./sqlaccessadv_setup.sh

[oracle@edrsr33p1-orcl SQL_Access_Advisor]$ ./sqlaccessadv_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 18 21:14:49
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

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> SQL> SQL> SQL> SQL> SQL> DROP TABLE temp_table purge
           *
ERROR at line 1:
ORA-00942: table or view does not exist


SQL>
Table created.

SQL> SQL> SQL> SQL> SQL> SQL> SQL>
System altered.

SQL> BEGIN dbms_sqltune.drop_sqlset('SQLSET_MY_SQLACCESS_WORKLOAD');
END;

*
ERROR at line 1:
```

Practice 11-2: Using SQL Access Advisor (continued)

```
ORA-13754: "SQL Tuning Set" "SQLSET_MY_SQLACCESS_WORKLOAD" does not
exist for user "SH".
ORA-06512: at "SYS.DBMS_SQLTUNE_INTERNAL", line 8406
ORA-06512: at "SYS.DBMS_SQLTUNE", line 2949
ORA-06512: at line 1


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


SQL> SQL>
Table created.

SQL> SQL> SQL>   2    3    4    5    6    7
PL/SQL procedure successfully completed.

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


SQL>
Table created.

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   25   26
27   28   29   30   31   32   33   34   35   36   37   38   39   40
41   42   43   44   45   46   47   48   49
PL/SQL procedure successfully completed.

SQL> SQL> SQL>
  COUNT(*)
----------
         5

1 row selected.

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

SQL>
PL/SQL procedure successfully completed.

SQL> SQL>
PL/SQL procedure successfully completed.

SQL> SQL>
PL/SQL procedure successfully completed.

SQL> SQL>
LAST_ANAL
---------
18-MAR-08
```

```
18-MAR-08
18-MAR-08
18-MAR-08
22-AUG-07
18-MAR-08
18-MAR-08
18-MAR-08
18-MAR-08
18-MAR-08
18-MAR-08
18-MAR-08
18-MAR-08

14 rows selected.

SQL> SQL> SQL> SQL> Disconnected from Oracle Database 11g Enterprise
Edition Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl SQL_Access_Advisor]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/SQL_Access_Advisor

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

sqlplus / as sysdba <<FIN!

SET ECHO ON
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 8000
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET LONG 1000

CONNECT / AS SYSDBA

grant dba to sh;

alter user sh identified by sh account unlock;

connect sh/sh

set serveroutput on size 32768;
set echo on;
```

Practice 11-2: Using SQL Access Advisor (continued)

```
variable norecs number;


Rem Clean up


declare
    name varchar2(30);
    cursor name_cur1 is
      select task_name from user_advisor_templates
        where task_name like '%SQLACCESS%';
  begin
    -------------------------------------------------------------------
-----------
    --  Get rid of templates, tasks and workloads.
    -------------------------------------------------------------------
-----------

    open name_cur1;

    loop
      fetch name_cur1 into name;
      exit when name_cur1%NOTFOUND;


dbms_advisor.update_task_attributes(name,null,null,'FALSE','FALSE');
      dbms_advisor.delete_task(name);
    end loop;

    close name_cur1;

  end;
/



Rem make a temp table


DROP TABLE temp_table purge;
CREATE TABLE temp_table AS SELECT * FROM SYS.WRI\$_ADV_SQLW_STMTS
WHERE NULL IS NOT NULL;



Rem create a large number of pseudo-random (repeatable) queries in
the temporary table


alter system flush shared_pool;
execute dbms_sqltune.drop_sqlset('SQLSET_MY_SQLACCESS_WORKLOAD');

drop table tempjfv purge;

create table tempjfv (c number, d varchar2(1000));
```

```
begin
for i in 1..20000 loop
  insert into tempjfv values(-
i,'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa');
end loop;
commit;
end;
/

drop table customersjfv purge;
create table customersjfv as select * from customers;


DECLARE
   sql_stmt   varchar2(2000);
   sqlsetname  VARCHAR2(30);
   sqlsetcur   dbms_sqltune.sqlset_cursor;
   refid    NUMBER;
 k NUMBER := 0;
 num_queries NUMBER := 500;
BEGIN

sql_stmt := 'SELECT /* QueryJFV 2 */ ch.channel_class, c.cust_city,
t.calendar_quarter_desc, SUM(s.amount_sold) sales_amount FROM
sh.sales s, sh.times t, sh.customers c, sh.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'') GROUP BY ch.channel_class, c.cust_city,
t.calendar_quarter_desc';

insert into temp_table values(1,1,NULL,0,'SH','Access
Advisor','Workload',0,0,0,0,1,100,2,to_date('02-FEB-
2007'),3,0,sql_stmt,1);

sql_stmt := 'SELECT /* QueryJFV 3 */ ch.channel_class, c.cust_city,
t.calendar_quarter_desc, SUM(s.amount_sold) sales_amount FROM
sh.sales s, sh.times t, sh.customers c, sh.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
(''1999-03'',''1999-04'') GROUP BY ch.channel_class, c.cust_city,
t.calendar_quarter_desc';

insert into temp_table values(1,1,NULL,0,'SH','Access
Advisor','Workload',0,0,0,0,1,100,2,to_date('02-FEB-
2007'),3,0,sql_stmt,1);
```

Practice 11-2: Using SQL Access Advisor (continued)

```
sql_stmt := 'SELECT /* QueryJFV 4 */ c.country_id, c.cust_city,
c.cust_last_name FROM sh.customers c WHERE c.country_id in (52790,
52798) ORDER BY c.country_id, c.cust_city, c.cust_last_name';

insert into temp_table values(1,1,NULL,0,'SH','Access
Advisor','Workload',0,0,0,0,1,100,2,to_date('02-FEB-
2007'),3,0,sql_stmt,1);


sql_stmt := 'select /* func_indx */ count(*) from tempjfv where
abs(c)=5';

insert into temp_table values(1,1,NULL,0,'SH','Access
Advisor','Workload',0,0,0,0,1,100,2,to_date('02-FEB-
2007'),3,0,sql_stmt,1);

sql_stmt := 'SELECT /* QueryJFV 5 */ * FROM sh.customersjfv WHERE
cust_state_province = ''CA''';

insert into temp_table values(1,1,NULL,0,'SH','Access
Advisor','Workload',0,0,0,0,1,100,2,to_date('02-FEB-
2007'),3,0,sql_stmt,1);


sqlsetname := 'SQLSET_MY_SQLACCESS_WORKLOAD';

  dbms_sqltune.create_sqlset(sqlsetname, 'Generated STS');

   OPEN sqlsetcur FOR
     SELECT
          SQLSET_ROW(null,null, sql_text, null, null, username,
module,
                     action, elapsed_time, cpu_time, buffer_gets,
disk_reads,
                     0,rows_processed, 0, executions, 0,
optimizer_cost, null,
                     priority, command_type,
                     to_char(last_execution_date,'yyyy-mm-
dd/hh24:mi:ss'),
                     0,0,NULL,0,NULL,NULL
                     )
    FROM temp_table;

   dbms_sqltune.load_sqlset(sqlsetname, sqlsetcur);
END;
/


SELECT COUNT(*) FROM
TABLE(DBMS_SQLTUNE.SELECT_SQLSET('SQLSET_MY_SQLACCESS_WORKLOAD'));




Rem Cleanup anything left behind
```

Practice 11-2: Using SQL Access Advisor (continued)

```
execute dbms_advisor.delete_task('%');
execute dbms_advisor.delete_sqlwkld('%');

EXECUTE DBMS_STATS.UNLOCK_SCHEMA_STATS('SH');

execute dbms_stats.gather_schema_stats(ownname => 'SH',
estimate_percent=> DBMS_STATS.AUTO_SAMPLE_SIZE, method_opt => 'FOR
ALL COLUMNS SIZE  AUTO', degree => 4 );

select distinct last_analyzed from dba_tab_statistics where
owner='SH';

REM EXECUTE DBMS_STATS.LOCK_SCHEMA_STATS('SH');

exit;

FIN!
```

2) Using Enterprise Manager, create a SQL Access Advisor tuning task based on the captured workload held in the SH.SQLSET_MY_ACCESS_WORKLOAD SQL tuning set using the SQLACCESS_WAREHOUSE template.
   a) Connect to Enterprise Manager Database Control as the `sh` user (password: `sh`). On the Home page, click the Advisor Central link in the Related Links section.
   b) On the Advisor Central page, click the SQL Advisors link. Then on the SQL Advisors page, click the SQL Access Advisor link.
   c) On the Initial Options page, select Inherit Options from a previously saved Task or Template, and then select the SQLACCESS_WAREHOUSE template. After this is done, click Continue.
   d) On the Workload Source page, select Use an existing SQL Tuning Set and enter SH.SQLSET_MY_SQLACCESS_WORKLOAD in the SQL Tuning Set field. (This SQL Tuning Set was generated earlier. It represents a warehouse workload that you want to analyze.) Click Next.
   e) On the Recommendation Options page, ensure that all possible access structures are selected, and that Comprehensive is selected. After this is done, click Next.
   f) On the Schedule page, enter `MY_SQLACCESS_TASK` in the Task Name field. Select the first Time Zone from the provided list (Click the torch icon.). After this is done, click Next.
   g) On the Review page, click Submit.
   h) Back to the Advisor Central page, click Refresh until the Status of your task is COMPLETED.
   i) After this is done, click the MY_SQLACCESS_TASK link in the Results table.
3) After this is done, investigate the proposed recommendations:
   a) Back to the Advisor Central page, click the MY_SQLACCESS_TASK link in the Results table. The task should have COMPLETED as the status.

b) This takes you to the Results page. From this page, you can see the potential benefit of implementing the SQL Access Advisor recommendations on the workload. There should be a huge difference between the original and the new costs. Click the Recommendation subtab.

c) On the Recommendations subtab, you can see the high-level overview of the recommendations. Basically, all possible types of recommendations were generated for this workload (Indexes, Materialized Views, Materialized View Logs, Partitions, and Others).

d) Ensure that all recommendations are selected, and click the Recommendation Details button. This takes you to the Details page, where you can see more details about each of the recommendations, as well as the corresponding SQL statements from the workload that are affected by these recommendations. You should see the following recommendations:
- Partition CUSTOMERS table.
- Create four materialized view log.
- Create a materialized view.
- Create one bitmap index.
- Create a function-based index.
- Create a B*-tree index on multiple columns.

e) Click OK.

4) Try to implement the generated recommendations. What happens, and why?

a) Back to the Recommendations subtab, click the Schedule Implementation button.

b) On the Schedule Implementation page, a warning is displayed indicating that the wizard will not try to implement its recommendations because some of them are very important changes that should be looked at closely by the administrator.

c) Click the Show SQL button to look at the script you could use to implement all recommendations. In fact, you already created this script and you will use it later in this lab. After you review the script, click Done.

d) Back to the Schedule Implementation page, click Cancel.

e) Click the Advisor Central locator link at the top of the “Results for Task” page.

f) On the Advisor Central page, select the SQL Access Advisor MY_SQLACCESS_TASK task and click Delete.

g) On the Information page, click Yes.

5) Use Enterprise Manager to verify the performance improvement if you implement the recommendations mentioned.

a) Click the Database tab at the top-right corner and then the “Software and Support” tab. On the “Software and Support” tabbed page, click the SQL Performance Analyzer link. You want to prove that implementing the recommendations is beneficial.

b) On the SQL Performance Analyzer page, click the Guided Workflow link.

Practice 11-2: Using SQL Access Advisor (continued)

c) On the Guided Workflow page, click the Execute icon on the line corresponding to step 1.

d) On the Create SQL Performance Analyzer Task page, enter `MY_SPA_TASK` in the SQL Performance Analyzer Task Name field. Then, enter `SH.SQLSET_MY_SQLACCESS_WORKLOAD` in the SQL Tuning Set Name field. After this is done, click Create.

e) Back to the Guided Workflow page, click the Execute icon for step 2.

f) On the Create Replay Trial page, enter `MY_SQL_REPLAY_BEFORE` in the Replay Trial Name field, and ensure that you select the Trial environment established check box. Then, click Submit.

g) Wait on the Guided Workflow page until step 2 is completed.

h) From your terminal session, connect as the `sh` user (password: `sh`) in the SQL*Plus session, and execute the `implement.sql` script. This script is a precreated script corresponding to the recommendations previously generated by your SQL Access Advisor session.

```
cd /home/oracle/solutions/SQL_Access_Advisor
[oracle@edrsr33p1-orcl SQL_Access_Advisor]$ ls
implement.sql  revert.sh  sqlaccessadv_setup.sh
[oracle@edrsr33p1-orcl SQL_Access_Advisor]$ sqlplus sh/sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 18 21:57:25
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> @implement
SQL>
SQL> Rem
SQL> Rem Creating new partitioned table
SQL> Rem
SQL> CREATE TABLE "SH"."CUSTOMERS1"
  2  (     "CUST_ID" NUMBER,
  3       "CUST_FIRST_NAME" VARCHAR2(20),
  4       "CUST_LAST_NAME" VARCHAR2(40),
  5       "CUST_GENDER" CHAR(1),
  6       "CUST_YEAR_OF_BIRTH" NUMBER(4,0),
  7       "CUST_MARITAL_STATUS" VARCHAR2(20),
  8       "CUST_STREET_ADDRESS" VARCHAR2(40),
  9       "CUST_POSTAL_CODE" VARCHAR2(10),
 10       "CUST_CITY" VARCHAR2(30),
 11       "CUST_CITY_ID" NUMBER,
 12       "CUST_STATE_PROVINCE" VARCHAR2(40),
 13       "CUST_STATE_PROVINCE_ID" NUMBER,
```

```
 14      "COUNTRY_ID" NUMBER,
 15      "CUST_MAIN_PHONE_NUMBER" VARCHAR2(25),
 16      "CUST_INCOME_LEVEL" VARCHAR2(30),
 17      "CUST_CREDIT_LIMIT" NUMBER,
 18      "CUST_EMAIL" VARCHAR2(30),
 19      "CUST_TOTAL" VARCHAR2(14),
 20      "CUST_TOTAL_ID" NUMBER,
 21      "CUST_SRC_ID" NUMBER,
 22      "CUST_EFF_FROM" DATE,
 23      "CUST_EFF_TO" DATE,
 24      "CUST_VALID" VARCHAR2(1)
 25  ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS
NOLOGGING
 26  TABLESPACE "EXAMPLE"
 27  PARTITION BY RANGE ("CUST_ID") INTERVAL( 3000) ( PARTITION
VALUES LESS THAN (3000)
 28  );

Table created.

SQL>
SQL> Rem
SQL> Rem Copying comments to new partitioned table
SQL> Rem
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_ID" IS 'primary key';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_FIRST_NAME" IS 'first
name of the customer';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_LAST_NAME" IS 'last
name of the customer';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_GENDER" IS 'gender;
low cardinality attribute';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_YEAR_OF_BIRTH" IS
'customer year of birth';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_MARITAL_STATUS" IS
'customer marital status; low cardinality attribute';

Comment created.
```

```
SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_STREET_ADDRESS" IS
'customer street address';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_POSTAL_CODE" IS
'postal code of the customer';

Comment created.

SQL>
SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_CITY" IS 'city where
the customer lives';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_STATE_PROVINCE" IS
'customer geography: state or province';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."COUNTRY_ID" IS 'foreign
key to the countries table (snowflake)';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_MAIN_PHONE_NUMBER" IS
'customer main phone number';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_INCOME_LEVEL" IS
'customer income level';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_CREDIT_LIMIT" IS
'customer credit limit';

Comment created.

SQL>
SQL> COMMENT ON COLUMN "SH"."CUSTOMERS1"."CUST_EMAIL" IS 'customer
email id';

Comment created.

SQL>
```

```
SQL> COMMENT ON TABLE "SH"."CUSTOMERS1" IS 'dimension table';

Comment created.

SQL>
SQL> Rem
SQL> Rem Copying constraints to new partitioned table
SQL> Rem
SQL> ALTER TABLE "SH"."CUSTOMERS1" ADD CONSTRAINT "CUSTOMERS_PK1"
PRIMARY KEY ("CUST_ID")
  2  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 NOLOGGING
COMPUTE STATISTICS
  3  TABLESPACE "EXAMPLE" ENABLE NOVALIDATE;

Table altered.

SQL>
SQL> Rem
SQL> Rem Copying referential constraints to new partitioned table
SQL> Rem
SQL> ALTER TABLE "SH"."CUSTOMERS1" ADD CONSTRAINT
"CUSTOMERS_COUNTRY_FK1" FOREIGN KEY ("COUNTRY_ID")
  2       REFERENCES "SH"."COUNTRIES" ("COUNTRY_ID") ENABLE
NOVALIDATE;

Table altered.

SQL>
SQL> Rem
SQL> Rem Copying indexes to new partitioned table
SQL> Rem
SQL> CREATE BITMAP INDEX "SH"."CUSTOMERS_GENDER_BIX1" ON
"SH"."CUSTOMERS1" ("CUST_GENDER")
  2  PCTFREE 10 INITRANS 2 MAXTRANS 255 NOLOGGING COMPUTE STATISTICS
  3  TABLESPACE "EXAMPLE" LOCAL;

Index created.

SQL>
SQL> Rem
SQL> Rem Copying object grants to new partitioned table
SQL> Rem
SQL> GRANT SELECT ON "SH"."CUSTOMERS1" TO "BI";

Grant succeeded.

SQL>
SQL> Rem
SQL> Rem Populating new partitioned table with data from original
table
SQL> Rem
SQL> INSERT /*+ APPEND */ INTO "SH"."CUSTOMERS1"
  2  SELECT * FROM "SH"."CUSTOMERS";

55500 rows created.

SQL> COMMIT;
```

```
Commit complete.

SQL>
SQL> begin
  2  dbms_stats.gather_table_stats('"SH"', '"CUSTOMERS1"', NULL,
dbms_stats.auto_sample_size);
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL>
SQL> Rem
SQL> Rem Renaming tables to give new partitioned table the original
table name
SQL> Rem
SQL> ALTER TABLE "SH"."CUSTOMERS" RENAME TO "CUSTOMERS11";

Table altered.

SQL> ALTER TABLE "SH"."CUSTOMERS1" RENAME TO "CUSTOMERS";

Table altered.

SQL>
SQL> Rem
SQL> Rem Revalidating dimensions for use with new partitioned table
SQL> Rem
SQL> ALTER DIMENSION "SH"."CUSTOMERS_DIM" COMPILE;

Dimension altered.

SQL>
SQL>
SQL> CREATE MATERIALIZED VIEW LOG ON
  2  "SH"."CUSTOMERS"
  3  WITH ROWID,
SEQUENCE("CUST_ID","CUST_CITY","CUST_STATE_PROVINCE")
  4  INCLUDING NEW VALUES;

Materialized view log created.

SQL>
SQL> CREATE MATERIALIZED VIEW LOG ON
  2  "SH"."CHANNELS"
  3  WITH ROWID,
SEQUENCE("CHANNEL_ID","CHANNEL_DESC","CHANNEL_CLASS")
  4  INCLUDING NEW VALUES;

Materialized view log created.

SQL>
SQL> CREATE MATERIALIZED VIEW LOG ON
  2  "SH"."TIMES"
  3  WITH ROWID, SEQUENCE("TIME_ID","CALENDAR_QUARTER_DESC")
  4  INCLUDING NEW VALUES;
```

Practice 11-2: Using SQL Access Advisor (continued)

```
Materialized view log created.

SQL>
SQL> CREATE MATERIALIZED VIEW LOG ON
  2  "SH"."SALES"
  3  WITH ROWID,
SEQUENCE("CUST_ID","TIME_ID","CHANNEL_ID","AMOUNT_SOLD")
  4  INCLUDING NEW VALUES;

Materialized view log created.

SQL>
SQL> CREATE MATERIALIZED VIEW "SH"."MV_01DF0000"
  2  REFRESH FAST WITH ROWID
  3  ENABLE QUERY REWRITE
  4  AS SELECT SH.CUSTOMERS.CUST_STATE_PROVINCE C1,
SH.CUSTOMERS.CUST_CITY C2, SH.CHANNELS.CHANNEL_CLASS
  5  C3, SH.CHANNELS.CHANNEL_DESC C4, SH.TIMES.CALENDAR_QUARTER_DESC
C5, SUM("SH"."SALES"."AMOUNT_SOLD")
  6  M1, COUNT("SH"."SALES"."AMOUNT_SOLD") M2, COUNT(*) M3 FROM
SH.CUSTOMERS,
  7  SH.CHANNELS, SH.TIMES, SH.SALES WHERE SH.SALES.CHANNEL_ID =
SH.CHANNELS.CHANNEL_ID
  8  AND SH.SALES.TIME_ID = SH.TIMES.TIME_ID AND SH.SALES.CUST_ID =
SH.CUSTOMERS.CUST_ID
  9  AND (SH.TIMES.CALENDAR_QUARTER_DESC IN ('1999-04', '1999-03',
'1999-02'
 10  , '1999-01')) AND (SH.CHANNELS.CHANNEL_DESC IN ('Internet',
'Catalog'
 11  )) AND (SH.CUSTOMERS.CUST_STATE_PROVINCE = 'CA') GROUP BY
SH.CUSTOMERS.CUST_STATE_PROVINCE,
 12  SH.CUSTOMERS.CUST_CITY, SH.CHANNELS.CHANNEL_CLASS,
SH.CHANNELS.CHANNEL_DESC,
 13  SH.TIMES.CALENDAR_QUARTER_DESC;

Materialized view created.

SQL>
SQL> begin
  2
dbms_stats.gather_table_stats('"SH"','"MV_01DF0000"',NULL,dbms_stats
.auto_sample_size);
  3  end;
  4  /

PL/SQL procedure successfully completed.

SQL>
SQL> CREATE BITMAP INDEX "SH"."CUSTOMERSJFV_IDX_01DF0000"
  2  ON "SH"."CUSTOMERSJFV"
  3  ("CUST_STATE_PROVINCE")
  4  COMPUTE STATISTICS;

Index created.

SQL>
```

Practice 11-2: Using SQL Access Advisor (continued)

```
SQL> CREATE INDEX "SH"."TEMPJFV_IDX_01DF0001"
  2  ON "SH"."TEMPJFV"
  3  (ABS("C"))
  4  COMPUTE STATISTICS;

Index created.

SQL>
SQL> CREATE INDEX "SH"."CUSTOMERS_IDX_01DF0002"
  2  ON "SH"."CUSTOMERS"
  3  ("COUNTRY_ID","CUST_CITY","CUST_LAST_NAME")
  4  COMPUTE STATISTICS;

Index created.

SQL>
SQL> exit
Disconnected from Oracle Database 11g Enterprise Edition Release
11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl SQL_Access_Advisor]$
```

i) Back to your Guided Workflow page, click the Execute icon corresponding to step 3.

j) On the Create Replay Trial page, enter `MY_SQL_REPLAY_AFTER` in the Replay Trial Name field. Ensure that you select the Trial environment established check box, and click Submit.

k) Wait until step 3 is completed.

l) Back to your Guided Workflow Enterprise Manager page, click the Execute icon corresponding to step 4.

m) On the Run Replay Trial Comparison page, ensure that you create a comparison between MY_SQL_REPLAY_BEFORE and MY_SQL_REPLAY_AFTER. Click Submit.

n) Wait until step 4 is completed.

o) Back to your Guided Workflow Enterprise Manager page, click the Execute icon corresponding to step 5.

p) On the SQL Performance Analyzer Task Result page, you can clearly see that the second trial is much faster than the original one. You should see that all five SQL statements are improved in the second trial due a changed execution plan.

q) To get more details, analyze the differences in execution plan for all five statements. You can do so directly from the SQL Performance Analyzer Task Result page by clicking each SQL ID in the Top 10 table. Each time you click a SQL ID, you can see in the SQL Details section all statistics differences between the two trials as well as the differences in execution plans.

Practice 11-2: Using SQL Access Advisor (continued)

r) After this is done, go back to the SQL Performance Analyzer page (Home > Software and Support > SQL Performance Analyzer), and delete MY_SPA_TASK by selecting it and clicking Delete. On the Confirmation page, click Delete.

s) Log out from Enterprise Manager.

6) From a terminal session, execute the `revert.sh` script to return to the situation you were in before you started the lab.

```
[oracle@edrsr33p1-orcl SQL_Access_Advisor]$ ./revert.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:02
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

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> SQL> SQL> SQL> SQL>
Table dropped.

SQL> SQL>
System altered.

SQL> SQL>
Table dropped.

SQL> SQL>
Table dropped.

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

SQL>
PL/SQL procedure successfully completed.

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

Practice 11-2: Using SQL Access Advisor (continued)

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

SQL> SQL> SQL> SQL>
Materialized view log dropped.

SQL> SQL>
Materialized view log dropped.

SQL> SQL>
Materialized view log dropped.

SQL> SQL>
Materialized view log dropped.

SQL> SQL>
Materialized view dropped.

SQL> SQL>
Index dropped.

SQL> SQL>
Table dropped.

SQL> SQL>
Table dropped.

SQL> SQL> Connected.
SQL> SQL>
Revoke succeeded.

SQL> SQL> SQL> Rem
SQL> Rem $Header: sh_main.sql 06-mar-2008.15:00:45 cbauwens Exp $
SQL> Rem
SQL> Rem sh_main.sql
SQL> Rem
SQL> Rem Copyright (c) 2001, 2008, Oracle. All rights reserved.
SQL> Rem
SQL> Rem    NAME
SQL> Rem      sh_main.sql - Main schema creation and load script
...
SQL> Rem
SQL>
SQL> SET ECHO OFF

specify password for SH as parameter 1:

specify default tablespace for SH as parameter 2:

specify temporary tablespace for SH as parameter 3:

specify password for SYS as parameter 4:

specify directory path for the data files as parameter 5:

writeable directory path for the log files as parameter 6:
```

```
specify version as parameter 7:


Session altered.


User dropped.

old   1: CREATE USER sh IDENTIFIED BY &pass
new   1: CREATE USER sh IDENTIFIED BY sh

User created.

old   1: ALTER USER sh DEFAULT TABLESPACE &tbs
new   1: ALTER USER sh DEFAULT TABLESPACE example
old   2:  QUOTA UNLIMITED ON &tbs
new   2:  QUOTA UNLIMITED ON example

User altered.

old   1: ALTER USER sh TEMPORARY TABLESPACE &ttbs
new   1: ALTER USER sh TEMPORARY TABLESPACE temp

User altered.


Grant succeeded.


Grant succeeded.


...


Grant succeeded.


Grant succeeded.


PL/SQL procedure successfully completed.

Connected.

Grant succeeded.

old   1: CREATE OR REPLACE DIRECTORY data_file_dir AS '&data_dir'
new   1: CREATE OR REPLACE DIRECTORY data_file_dir AS
'/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/'

Directory created.

old   1: CREATE OR REPLACE DIRECTORY log_file_dir AS '&log_dir'
new   1: CREATE OR REPLACE DIRECTORY log_file_dir AS '/home/oracle/'
```

```
Directory created.

Grant succeeded.


Grant succeeded.


Grant succeeded.

Connected.

Session altered.


Session altered.


Table created.


Table created.


...


Table created.


Creating constraints ...

Table altered.


...


Table altered.


specify password for SH as parameter 1:

specify path for data files as parameter 2:

specify path for log files as parameter 3:

specify version as parameter 4:

Looking for indexes that could slow down load ...

no rows selected


loading TIMES using:
```

Practice 11-2: Using SQL Access Advisor (continued)

```
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/time_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/time_v
3.dat
/home/oracle/time_v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:23
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 1000.

Load completed - logical record count 1826.


loading COUNTRIES using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/coun_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/coun_v
3.dat
/home/oracle/coun_v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:24
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Load completed - logical record count 23.


loading CUSTOMERS using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/cust_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/cust1v
3.dat
/home/oracle/cust1v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:24
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 10000.
Save data point reached - logical record count 20000.
Save data point reached - logical record count 30000.
Save data point reached - logical record count 40000.
Save data point reached - logical record count 50000.

Load completed - logical record count 55500.


loading PRODUCTS  using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/prod_v
3.ctl
```

```
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/prod1v
3.dat
/home/oracle/prod1v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:24
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Load completed - logical record count 72.


loading PROMOTIONS  using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/prom_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/prom1v
3.dat
/home/oracle/prom1v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:25
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 10.
Save data point reached - logical record count 20.
...
Save data point reached - logical record count 500.

Load completed - logical record count 503.


loading CHANNELS using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/chan_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/chan_v
3.dat
/home/oracle/chan_v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:25
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Load completed - logical record count 5.


loading SALES  using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/sale_v
3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/sale1v
3.dat
/home/oracle/sale1v3.log
```

Practice 11-2: Using SQL Access Advisor (continued)

```
SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:25
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 100000.
...
Save data point reached - logical record count 900000.

Load completed - logical record count 916039.


loading COSTS using external table


Table created.


82112 rows created.


loading additonal SALES using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/dmsal_
v3.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/dmsal_
v3.dat
/home/oracle/dmsal_v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:38
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 100.
...
Save data point reached - logical record count 2800.

Load completed - logical record count 2804.


loading SUPPLEMENTARY DEMOGRAPHICS using:
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/dem_v3
.ctl
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/dem1v3
.dat
/home/oracle/dem1v3.log

SQL*Loader: Release 11.1.0.6.0 - Production on Thu Mar 20 15:22:38
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.

Save data point reached - logical record count 10.
...
Save data point reached - logical record count 4500.

Load completed - logical record count 4500.
```

Practice 11-2: Using SQL Access Advisor (continued)

```
Commit complete.

Enabling constraints ...
Table altered.

...

Table altered.

Creating additional indexes ...
Index created.

...

Index created.

Create dimensions ...
Dimension created.

Commit complete.

PL/SQL procedure successfully completed.

no rows selected

Dimension created.

PL/SQL procedure successfully completed.

no rows selected

Dimension created.

PL/SQL procedure successfully completed.

no rows selected
```

Practice 11-2: Using SQL Access Advisor (continued)

```
Dimension created.


PL/SQL procedure successfully completed.


no rows selected


Dimension created.


PL/SQL procedure successfully completed.


no rows selected

Creating MVs as tables ...


View created.


Table created.


Table created.


Index created.


Index created.


Index created.


Index created.

Creating materialized views ...


Materialized view created.


Materialized view created.


Creating comments ...

Comment created.


...
```

```
Comment created.


Creating OLAP metadata ...
<<<<< CREATE CWMLite Metadata for the Sales History Schema >>>>>
-
<<<<< CREATE CATALOG sh_cat for Sales History >>>>>
        Catalog Dropped
        CWM Collect Garbage
-
<<<<< CREATE the Sales CUBE >>>>>
        Sales amount, Sales quantity
        <TIMES CHANNELS PRODUCTS CUSTOMERS PROMOTIONS >
        Drop SALES_CUBE prior to recreation
        Cube Dropped
        Add dimensions -
         to SALES_CUBE and map the foreign keys
        Create measures -
         for SALES_CUBE and map to columns in the fact table
        Set default aggregation method -
         to SUM for all measures over TIME
        Add SALES_CUBE to the catalog
        SALES_CUBE successfully added to sh_cat
-
<<<<< CREATE the Cost CUBE >>>>>
        Unit Cost, Unit Price < TIMES PRODUCTS CHANNELS PROMOTIONS >
        Drop COST_CUBE prior to recreation
        Cube Dropped
        Add dimensions -
         to COST_CUBE and map the foreign keys
        Create measures -
         for COST_CUBE and map to columns in the fact table
        Set default aggregation method -
         to SUM for all measures over TIME
        Add COST_CUBE to the catalog
        COST_CUBE successfully added to sh_cat
-
<<<<< TIME DIMENSION >>>>>
Dimension - display name, description and plural name
Level - display name and description
Hierarchy - display name and description
        - default calculation hierarchy
        - default display hierarchy
Level Attributes - name, display name, description
Drop dimension attributes prior to re-creation
Create dimension attributes and add their level attributes
        - Long Description created
        - Short Description created
        - Period Number of Days created
        - Period End Date created
Classify entity descriptor use
        - Time dimension
        - Long description
        - Day name
        - Calendar month description
```

```
        - Calendar quarter description
        - Fiscal month description
        - Fiscal quarter description
        - Short Description
        - Day name
        - Calendar month description
        - Calendar quarter description
        - Fiscal month description
        - Fiscal quarter description
        - Time Span
        - Days in calendar month
        - Days in calendar quarter
        - Days in calendar year
        - Days in fiscal month
        - Days in fiscal quarter
        - Days in fiscal year
        - End Date
        - End of calendar month
        - End of calendar quarter
        - End of calendar year
        - End of fiscal month
        - End of fiscal quarter
        - End of fiscal year
-
<<<<< CUSTOMERS DIMENSION >>>>>
Dimension - display name, description and plural name
Level - display name and description
Hierarchy - display name and description
        - default calculation hierarchy
        - default display hierarchy
Level Attributes - name, display name, description
Drop dimension attributes prior to re-creation
No attribute to drop
No attribute to drop
No attribute to drop
No attribute to drop
No attribute to drop
No attribute to drop
No attribute to drop
        No attribute to drop
No attribute to drop
No attribute to drop
        No attribute to drop
        No attribute to drop
Create dimension attributes and add their level attributes
        - Long Description created
        - Short Description created
        - Other Customer Information created
Classify entity descriptor use
        - Long Description
        - Short Description
-
<<<<< PRODUCTS DIMENSION >>>>>
Dimension - display name, description and plural name
Level - display name and description
Hierarchy - display name and description
        - default calculation hierarchy
```

Practice 11-2: Using SQL Access Advisor (continued)

```
        - default display hierarchy
Level Attributes - name, display name, description
Drop dimension attributes prior to re-creation
No attribute to drop
Create dimension attributes and add their level attributes
        - Long Description created
        - Short Description created
Classify entity descriptor use
        - Long Description
        - Short Description
-
<<<<< PROMOTIONS DIMENSION >>>>>
Dimension - display name, description and plural name
Level - display name and description
Hierarchy - display name and description
        - default calculation hierarchy
        - default display hierarchy
Level Attributes - name, display name, description
Drop dimension attributes prior to re-creation
No attribute to drop
Create dimension attributes and add their level attributes
        - Long Description created
        - Short Description created
Classify entity descriptor use
        - Long Description
        - Short Description
-
<<<<< CHANNELS DIMENSION >>>>>
Dimension - display name, description and plural name
Level - display name and description
Hierarchy - display name and description
        - default calculation hierarchy
        - default display hierarchy
Level Attributes - name, display name, description
Drop dimension attributes prior to re-creation
No attribute to drop
Create dimension attributes and add their level attributes
        - Long Description created
        - Short Description created
Classify entity descriptor use
        - Long Description
        - Short Description
-
<<<<< FINAL PROCESSING >>>>>
        - Changes have been committed

PL/SQL procedure successfully completed.


Commit complete.


gathering statistics ...

PL/SQL procedure successfully completed.
```

Practice 11-2: Using SQL Access Advisor (continued)

```
PL/SQL procedure successfully completed.

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl SQL_Access_Advisor]$


--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/SQL_Access_Advisor/sh

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin:/bin

cp * $ORACLE_HOME/demo/schema/sales_history

sqlplus / as sysdba <<FIN!

SET ECHO ON
SET FEEDBACK 1
SET NUMWIDTH 10
SET LINESIZE 8000
SET TRIMSPOOL ON
SET TAB OFF
SET PAGESIZE 100
SET LONG 1000

CONNECT / AS SYSDBA

grant dba to sh;

alter user sh identified by sh account unlock;

connect sh/sh

set serveroutput on size 32768;
set echo on;
variable norecs number;


Rem Clean up


declare
    name varchar2(30);
    cursor name_cur1 is
      select task_name from user_advisor_templates
        where task_name like '%SQLACCESS%';
```

```
  begin
    --------------------------------------------------------------------
-----------
    --  Get rid of templates, tasks and workloads.
    --------------------------------------------------------------------
-----------

    open name_cur1;

    loop
      fetch name_cur1 into name;
      exit when name_cur1%NOTFOUND;


dbms_advisor.update_task_attributes(name,null,null,'FALSE','FALSE');
      dbms_advisor.delete_task(name);
    end loop;

    close name_cur1;

  end;
/


Rem make a temp table


DROP TABLE temp_table purge;

alter system flush shared_pool;

drop table tempjfv purge;

drop table customersjfv purge;


execute dbms_advisor.delete_task('%');
execute dbms_advisor.delete_sqlwkld('%');

execute dbms_sqltune.drop_sqlset('SQLSET_MY_SQLACCESS_WORKLOAD');


EXECUTE DBMS_STATS.UNLOCK_SCHEMA_STATS('SH');



DROP MATERIALIZED VIEW LOG ON "SH"."CUSTOMERS";

DROP MATERIALIZED VIEW LOG ON "SH"."CHANNELS";

DROP MATERIALIZED VIEW LOG ON "SH"."TIMES";

DROP MATERIALIZED VIEW LOG ON "SH"."SALES";

DROP MATERIALIZED VIEW "SH"."MV_01DF0000";

DROP INDEX "SH"."CUSTOMERS_IDX_01DF0002";
```

Practice 11-2: Using SQL Access Advisor (continued)

```
DROP TABLE "SH"."CUSTOMERS" PURGE;

DROP TABLE "SH"."CUSTOMERS11" CASCADE CONSTRAINTS PURGE;

connect / as sysdba

revoke dba from sh;

@sh_main sh example temp oracle
/u01/app/oracle/product/11.1.0/db_1/demo/schema/sales_history/
/home/oracle/ v3

exit;

FIN!
```

Practice 11-3: Using Automatic SQL Tuning

In this practice, you manually launch Automatic SQL Tuning to automatically tune a small application workload. You then investigate the outcomes and configuration possibilities.

1) On the Server page, click Automated Maintenance Tasks, check that Status is set to Enabled, and click Configure. Click the Configure button next to Automatic SQL Tuning. Select Yes for "Automatic Implementation of SQL Profiles." Then, click Apply. Execute the `ast_setup.sh` script from a terminal window connected as the `oracle` user. This script creates the `AST` user used throughout this practice, turns off automatic maintenance tasks, and drops any existing profiles on queries executed by the `AST` user.

```
$ cd /home/oracle/solutions/Automatic_SQL_Tuning

$ ./ast_setup.sh

[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$ ./ast_setup.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 18 15:31:49
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL> SQL> SQL> drop user ast cascade
          *
ERROR at line 1:
ORA-01918: user 'AST' does not exist


SQL> SQL>
User created.

SQL> SQL>
Grant succeeded.

SQL> SQL>
System altered.

SQL> SQL> SQL> SQL> SQL> SQL>
System altered.

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

SQL> SQL> SQL> SQL> SQL> SQL>   2    3    4    5    6    7    8    9
PL/SQL procedure successfully completed.
```

```
SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Automatic_SQL_Tuning

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin

sqlplus / as sysdba <<FIN!

set echo on

drop user ast cascade;

create user ast identified by ast;

grant dba to ast;

alter system flush shared_pool;

--
-- Turn off AUTOTASK
--

alter system set "_enable_automatic_maintenance"=0;

--
-- Clear out old executions of auto-sqltune
--

exec dbms_sqltune.reset_tuning_task('SYS_AUTO_SQL_TUNING_TASK');

--
-- Drop any profiles on AST queries
--

declare
  cursor prof_names is
    select name from dba_sql_profiles where sql_text like '%AST%';
begin
  for prof_rec in prof_names loop
    dbms_sqltune.drop_sql_profile(prof_rec.name);
  end loop;
end;
```

```
/

FIN!
```

2) In preparation for the practice, you should execute a workload. Execute the `run_workload_stream.sh` script. This script executes, multiple times, a query that is not correctly optimized. The query in question uses hints that force the optimizer to pick a suboptimal execution plan. The script execute for approximately 30 seconds.

```
./run_workload_stream.sh

[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$
./run_workload_stream.sh
Tue Mar 18 15:38:05 GMT-7 2008

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 18 15:38:05
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL> SQL> SQL>
no rows selected

SQL>
no rows selected

SQL>
no rows selected

...

SQL>
no rows selected

SQL>
no rows selected

SQL>
no rows selected

SQL>
no rows selected

SQL>
no rows selected

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
```

```
and Real Application Testing options
Tue Mar 18 15:38:23 GMT-7 2008
[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Automatic_SQL_Tuning

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin

date

sqlplus ast/ast <<FIN!

set echo on

select /*+ USE_NL(s c) FULL(s) FULL(c) AST */ c.cust_id,
sum(s.quantity_sold) from sh.sales s, sh.customers c where s.cust_id
= c.cust_id and c.cust_id < 2 group by c.cust_id;

…

select /*+ USE_NL(s c) FULL(s) FULL(c) AST */ c.cust_id,
sum(s.quantity_sold) from sh.sales s, sh.customers c where s.cust_id
= c.cust_id and c.cust_id < 2 group by c.cust_id;

FIN!

date
```

3) Automatic SQL Tuning is implemented using an automated task that runs during maintenance windows. However, you do not wait for the next maintenance window to open. Instead, you force the opening of your next maintenance window now. This automatically triggers the Automatic SQL Tuning task. Execute the `run_ast.sh` script to open your next maintenance window now. The script's execution takes a couple of minutes.

```
./run_ast.sh

[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$ ./run_ast.sh
Tue Mar 18 15:43:48 GMT-7 2008

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 18 15:43:48
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.
```

```
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

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

SQL> SQL> SQL> SQL>   2    3    4
PL/SQL procedure successfully completed.

SQL> SQL>
WINDOW
--------------------------------------------------------------------
------------
TUESDAY_WINDOW

SQL> SQL> SQL> SQL> SQL> SQL>   2
System altered.

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

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

SQL> SQL>
PL/SQL procedure successfully completed.

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

SQL> SQL>   2    3    4    5    6    7    8    9   10   11   12   13
14   15   16   17   18   19   20   21   22
PL/SQL procedure successfully completed.

SQL> SQL>   2
System altered.

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

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

SQL> SQL> SQL> Disconnected from Oracle Database 11g Enterprise
Edition Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
Tue Mar 18 15:44:50 GMT-7 2008
[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$

--------------------------------------------------------------

#!/bin/bash
```

Practice 11-3: Using Automatic SQL Tuning (continued)

```
cd /home/oracle/solutions/Automatic_SQL_Tuning

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin

date

sqlplus / as sysdba <<FIN!

set echo on

exec dbms_workload_repository.create_snapshot;

variable window varchar2(20);

begin
 select upper(to_char(sysdate,'fmday'))||'_WINDOW' into :window from
dual;
end;
/

print window;

--
-- Open the corresponding maintenance window, but with other clients
disabled
--

alter system set "_enable_automatic_maintenance"=1
/

exec dbms_auto_task_admin.disable( -
  'auto optimizer stats collection', null, :window);

exec dbms_auto_task_admin.disable( -
  'auto space advisor', null, :window);

exec dbms_scheduler.open_window(:window, null, true);

--
-- Close the maintenance window when sqltune is done
--

exec dbms_lock.sleep(60);

declare
  running number;
begin

  loop
    select count(*)
```

```
    into   running
    from   dba_advisor_executions
    where  task_name = 'SYS_AUTO_SQL_TUNING_TASK' and
           status = 'EXECUTING';

    if (running = 0) then
      exit;
    end if;

    dbms_lock.sleep(60);
  end loop;

  dbms_scheduler.close_window(:window);

end;
/

alter system set "_enable_automatic_maintenance"=0
/

--
-- Re-enable the other guys so they look like they are enabled in
EM.
-- Still they will be disabled because we have set the underscore.
--

exec dbms_auto_task_admin.enable( -
  'auto optimizer stats collection', null, :window);

exec dbms_auto_task_admin.enable( -
  'auto space advisor', null, :window);


FIN!

date
```

4) Execute the `run_workload_stream.sh` script again. What do you observe?

   a) You should see that the execution time for `run_workload_stream.sh` is much faster than the original execution. This is probably due to the fact that Automatic SQL Tuning implemented a profile for your statement automatically.

```
./run_workload_stream.sh

[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$
./run_workload_stream.sh
Tue Mar 18 15:46:42 GMT-7 2008

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 18 15:46:42
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
```

```
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL> SQL> SQL>
no rows selected

SQL>
no rows selected

SQL>
no rows selected

...

SQL>
no rows selected

SQL>
no rows selected

SQL>
no rows selected

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
Tue Mar 18 15:46:42 GMT-7 2008
[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$

------------------------------------------------------------
#!/bin/bash

cd /home/oracle/solutions/AST

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin

date

sqlplus ast/ast <<FIN!

set echo on

select /*+ USE_NL(s c) FULL(s) FULL(c) AST */ c.cust_id,
sum(s.quantity_sold) from sh.sales s, sh.customers c where s.cust_id
= c.cust_id and c.cust_id < 2 group by c.cust_id;

…
```

```
select /*+ USE_NL(s c) FULL(s) FULL(c) AST */ c.cust_id,
sum(s.quantity_sold) from sh.sales s, sh.customers c where s.cust_id
= c.cust_id and c.cust_id < 2 group by c.cust_id;

FIN!

date
```

5) Force the creation of an Automatic Workload Repository (AWR) snapshot.

```
./create_snapshot.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 18 15:51:19
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

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

SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Automatic_SQL_Tuning

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin

sqlplus / as sysdba <<FIN!

set echo on

exec dbms_workload_repository.create_snapshot;

FIN!
```

Practice 11-3: Using Automatic SQL Tuning (continued)

6) How would you confirm that a SQL Profile was automatically implemented?

   a) In Oracle Enterprise Manager, locate the Automatic SQL Tuning summary page under **Server > Automated Maintenance Tasks > Automatic SQL Tuning**. The task has already run in one maintenance window and has results ready to be viewed.

   b) View the tuning results.

   c) Look at the graphs on the Task Activity Summary page.

   d) Focus on understanding the pie chart and the bar graph next to it. You should be able to get a feeling for the general finding breakdown, as well as the number of SQL profiles implemented by the task.

   e) Click View Report to see a detailed SQL-level report. Find the SQL that ran in the `AST` schema. Note the green check mark meaning that the profile was implemented.

   f) Click the corresponding option button and then View Recommendations.

   g) Click the Compare Explain Plans eyeglass icon for the SQL Profile entry.

   h) View the old and new explain plans for the query.

   i) Then click the "Recommendations for SQL_ID" locator link to return to the previous screen.

   j) Investigate a SQL profile. While still on the "Recommendations for SQL_ID" page, click the SQL text to go to the SQL Details page for this SQL.

   k) This takes you to the Tuning History tab. Note the link to `SYS_AUTO_SQL_TUNING_TASK` that is there to show that the SQL was tuned by this tuning task.

   l) Look at the Plan Control subpage and note that a profile was created automatically for this SQL. The `AUTO` type means it was automatically created.

   m) Click the Statistics tab to take a look at the execution history for this SQL.

   n) Depending on the speed of your machine, you may not see two hash values. If that is the case, ignore this step and the following one. Select Real Time: Manual Refresh from the View Data and then each of possible two Plan Hash Values from the corresponding drop-down list. Choose one after the other and wait for the page to refresh each time.

   o) Depending on the speed of your environment, you should see one statement with a relatively high elapsed time per execution, and one with very low elapsed time per execution. This shows the improved plan. If you select All from the Plan Hash Values drop-down list, you might not be able to see the execution corresponding to the statement after tuning in the Summary graph. This might be because the workload was too short to execute.

7) Generate a text report for more indepth information. From the command line, execute the `get_task_report.sh` script. What do you observe?

Practice 11-3: Using Automatic SQL Tuning (continued)

a) Note the first queries that fetch execution name and object number from the advisor schema, followed by the final query that gets the text report. In the text report, look for the section about the SQL profile finding and peruse the Validation Results section. This shows you the execution statistics observed during test-execute and allows you to get a better idea about the profile's quality. You can also use the `report_auto_tuning_task` API to get reports that span multiple executions of the task.

```
./get_task_report.sh

[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$ ./get_task_report.sh

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 18 16:02:18
2008

Copyright (c) 1982, 2007, Oracle.  All rights reserved.


Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 -
Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options

SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL>
Session altered.

SQL> SQL>   2    3    4
EXECUTION_NAME                  STATUS      EXECUTION_START
------------------------------ ----------- -------------------
EXEC_1                          COMPLETED   03/18/2008 15:43:54

SQL> SQL> SQL> SQL>   2    3    4    5    6    7
PL/SQL procedure successfully completed.

SQL> SQL>
LAST_EXEC
--------------------------------------------------------------------
------------
EXEC_1

SQL> SQL> SQL> SQL> SQL> SQL> SQL> SQL>   2    3    4    5    6    7
8    9   10
PL/SQL procedure successfully completed.

SQL> SQL>
    OBJ_ID
----------
         3

SQL> SQL> SQL> SQL> SQL> SQL>   2    3  GENERAL INFORMATION SECTION
--------------------------------------------------------------------
-----------
Tuning Task Name                        : SYS_AUTO_SQL_TUNING_TASK
Tuning Task Owner                       : SYS
```

Practice 11-3: Using Automatic SQL Tuning (continued)

```
Workload Type                           : Automatic High-Load SQL
Workload
Scope                                   : COMPREHENSIVE
Global Time Limit(seconds)              : 3600
Per-SQL Time Limit(seconds)             : 1200
Completion Status                       : COMPLETED
Started at                              : 03/18/2008 15:43:54
Completed at                            : 03/18/2008 15:44:14
Number of Candidate SQLs                : 4
Cumulative Elapsed Time of SQL (s)      : 27

--------------------------------------------------------------------
-----------
Object ID  : 3
Schema Name: AST
SQL ID     : by9m5m597zh19
SQL Text   : select /*+ USE_NL(s c) FULL(s) FULL(c) AST */
c.cust_id,
             sum(s.quantity_sold) from sh.sales s, sh.customers c
where
             s.cust_id = c.cust_id and c.cust_id < 2 group by
c.cust_id

--------------------------------------------------------------------
-----------
FINDINGS SECTION (2 findings)
--------------------------------------------------------------------
-----------

1- SQL Profile Finding (see explain plans section below)
--------------------------------------------------------
  A potentially better execution plan was found for this statement.
  SQL profile "SYS_SQLPROF_01463043cc730000" was created
automatically for
  this statement.

  Recommendation (estimated benefit: 98.62%)
  ------------------------------------------
  - An automatically-created SQL profile is present on the system.
    Name:   SYS_SQLPROF_01463043cc730000
    Status: ENABLED

  Validation results
  ------------------
  The SQL profile was tested by executing both its plan and the
original plan
  and measuring their respective execution statistics. A plan may
have been
  only partially executed if the other could be run to completion in
less time.

                           Original Plan  With SQL Profile  %
Improved
                           -------------  ----------------  --------
--
  Completion Status:            COMPLETE          COMPLETE
```

```
  Elapsed Time(ms):                   182                 0
100%
  CPU Time(ms):                       182                 0
100%
  User I/O Time(ms):                    0                 0
  Buffer Gets:                       3177                44
98.61%
  Disk Reads:                           0                 0
  Direct Writes:                        0                 0
  Rows Processed:                       0                 0
  Fetches:                              0                 0
  Executions:                           1                 1

2- Index Finding (see explain plans section below)
--------------------------------------------------
  The execution plan of this statement can be improved by creating
one or more
  indices.

  Recommendation (estimated benefit: 90.97%)
  ------------------------------------------
  - Consider running the Access Advisor to improve the physical
schema design
    or creating the recommended index.
    create index SH.IDX$$_00010001 on SH.SALES("CUST_ID");

  Rationale
  ---------
    Creating the recommended indices significantly improves the
execution plan
    of this statement. However, it might be preferable to run
"Access Advisor"
    using a representative SQL workload as opposed to a single
statement. This
    will allow to get comprehensive index recommendations which
takes into
    account index maintenance overhead and additional space
consumption.

--------------------------------------------------------------------
-----------
EXPLAIN PLANS SECTION
--------------------------------------------------------------------
-----------

1- Original With Adjusted Cost
------------------------------
Plan hash value: 4005616876

--------------------------------------------------------------------
------------
-------------------
| Id  | Operation              | Name      | Rows   | Bytes | Cost
(%CPU)| Time
  | Pstart| Pstop |
--------------------------------------------------------------------
------------
```

Practice 11-3: Using Automatic SQL Tuning (continued)

```
-------------------
|   0 | SELECT STATEMENT      |           |     1 |    13 |   902
(2)| 00:00:1
1 |       |       |
|   1 |  HASH GROUP BY        |           |     1 |    13 |   902
(2)| 00:00:1
1 |       |       |
|   2 |   NESTED LOOPS        |           |     1 |    13 |   901
(2)| 00:00:1
1 |       |       |
|*  3 |    TABLE ACCESS FULL  | CUSTOMERS |     1 |     5 |   405
(1)| 00:00:0
5 |       |       |
|   4 |    PARTITION RANGE ALL|           |     1 |     8 |   495
(3)| 00:00:0
6 |     1 |    28 |
|*  5 |     TABLE ACCESS FULL | SALES     |     1 |     8 |   495
(3)| 00:00:0
6 |     1 |    28 |
-----------------------------------------------------------------------
------------
-------------------

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

   3 - filter("C"."CUST_ID"<2)
   5 - filter("S"."CUST_ID"<2 AND "S"."CUST_ID"="C"."CUST_ID")

2- Using SQL Profile
--------------------
Plan hash value: 3070788227

-----------------------------------------------------------------------
------------
---------------------------------------
| Id  | Operation                            | Name            | Rows
| Bytes |
Cost (%CPU)| Time      | Pstart| Pstop |
-----------------------------------------------------------------------
------------
---------------------------------------
|   0 | SELECT STATEMENT                     |                 |
1 |    13 |
   55   (2)| 00:00:01 |       |       |
|   1 |  HASH GROUP BY                       |                 |
1 |    13 |
   55   (2)| 00:00:01 |       |       |
|   2 |   NESTED LOOPS                       |                 |
1 |    13 |
   54   (0)| 00:00:01 |       |       |
|   3 |    PARTITION RANGE ALL               |                 |
1 |     8 |
   54   (0)| 00:00:01 |     1 |    28 |
|   4 |     TABLE ACCESS BY LOCAL INDEX ROWID| SALES           |
1 |     8 |
   54   (0)| 00:00:01 |     1 |    28 |
```

```
|   5 |      BITMAP CONVERSION TO ROWIDS     |                 |
|       |
           |           |       |       |
|*  6 |       BITMAP INDEX RANGE SCAN        | SALES_CUST_BIX |
|       |
           |           |     1 |    28 |
|*  7 |    INDEX UNIQUE SCAN                 | CUSTOMERS_PK   |
1 |     5 |
    0   (0)| 00:00:01 |       |       |
--------------------------------------------------------------------
------------
---------------------------------------

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

   6 - access("S"."CUST_ID"<2)
       filter("S"."CUST_ID"<2)
   7 - access("S"."CUST_ID"="C"."CUST_ID")
       filter("C"."CUST_ID"<2)

3- Using New Indices
--------------------
Plan hash value: 1871796534

--------------------------------------------------------------------
------------
---------------------------------------
| Id  | Operation                            | Name            | Rows
| Bytes |
Cost (%CPU)| Time      | Pstart| Pstop |
--------------------------------------------------------------------
------------
---------------------------------------
|   0 | SELECT STATEMENT                     |                 |
1 |    13 |
    5   (0)| 00:00:01 |       |       |
|   1 |  SORT GROUP BY NOSORT                |                 |
1 |    13 |
    5   (0)| 00:00:01 |       |       |
|   2 |   NESTED LOOPS                       |                 |
|       |
           |           |       |       |
|   3 |    NESTED LOOPS                      |                 |
1 |    13 |
    5   (0)| 00:00:01 |       |       |
|*  4 |     INDEX RANGE SCAN                 | CUSTOMERS_PK   |
1 |     5 |
    2   (0)| 00:00:01 |       |       |
|*  5 |     INDEX RANGE SCAN                 | IDX$$_00010001 |
1 |       |
    2   (0)| 00:00:01 |       |       |
|   6 |    TABLE ACCESS BY GLOBAL INDEX ROWID| SALES           |
1 |     8 |
    3   (0)| 00:00:01 | ROWID | ROWID |
--------------------------------------------------------------------
------------
```

```
---------------------------------------

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

   4 - access("C"."CUST_ID"<2)
   5 - access("S"."CUST_ID"="C"."CUST_ID")
       filter("S"."CUST_ID"<2)

--------------------------------------------------------------------
-----------


SQL> SQL> Disconnected from Oracle Database 11g Enterprise Edition
Release 11.1.0.6.0 - Production
With the Partitioning, Oracle Label Security, OLAP, Data Mining
and Real Application Testing options
[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Automatic_SQL_Tuning

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin

sqlplus / as sysdba <<FIN!

set echo on
set long 1000000000
set longchunksize 1000

--
-- Check the execution names
--
alter session set nls_date_format = 'MM/DD/YYYY HH24:MI:SS';

select execution_name, status, execution_start
from   dba_advisor_executions
where  task_name = 'SYS_AUTO_SQL_TUNING_TASK'
order by execution_start;

variable last_exec varchar2(30);

begin
  select max(execution_name) keep (dense_rank last order by
execution_start)
  into   :last_exec
  from   dba_advisor_executions
  where  task_name = 'SYS_AUTO_SQL_TUNING_TASK';
```

```
end;
/

print :last_exec

--
-- Find the object ID for query AST with sql_id by9m5m597zh19
--

variable obj_id number;

begin
  select object_id
  into   :obj_id
  from   dba_advisor_objects
  where  task_name = 'SYS_AUTO_SQL_TUNING_TASK' and
         execution_name = :last_exec and
         type = 'SQL' and
         attr1 = 'by9m5m597zh19';
end;
/

print :obj_id

--
-- Get a text report to drill down on this one query
--
set pagesize 0
select dbms_sqltune.report_auto_tuning_task(
  :last_exec, :last_exec, 'TEXT', 'TYPICAL', 'ALL', :obj_id)
from dual;

FIN!
```

8) Investigate how to configure Automatic SQL Tuning using Enterprise Manager.
   a) Back in EM, go to the Automated Maintenance Tasks page.
   b) The chart here shows times in the past when each client was executed, and times in the future when they are scheduled to run again.
   c) Modify the graph's begin and end points with the widgets at the upper right.
   d) Click the Configure button.
   e) This brings you to the Automated Maintenance Tasks Configuration page.
   f) From this page, you can disable individual clients and change which windows they run in.
   g) Disable the Automatic SQL Tuning client entirely, click Apply, and then click the locator link to return to the last page.
   h) Note that no light blue bars appear for Automatic SQL Tuning in the future.
   i) Return to the configuration page, enable the task again, and click Apply to undo your changes.

Practice 11-3: Using Automatic SQL Tuning (continued)

j) Click the Automatic SQL Tuning link on the Automated Maintenance Tasks Configuration page.

k) This takes you to the page where you can configure the task itself, and set beyond when it will run.

l) Note that there are more fine-grained controls here, such as one that allows the task to run but not implement profiles, and one that allows you to control the maximum number of profiles created per run.

9) Investigate how to configure Automatic SQL Tuning using PL/SQL. From your terminal session, execute the `manual_config.sh` script. What does it do?

a) Note the first action. You changed the total time limit for the task. Instead of running for an unlimited amount of time (still bound by the maintenance window boundaries), it now runs for a maximum of one hour. The `execute_tuning_task` API call runs the task immediately, in the foreground. Use this to run the task yourself whenever you want.

```
./manual_config.sh

--------------------------------------------------------------

#!/bin/bash

cd /home/oracle/solutions/Automatic_SQL_Tuning

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin

sqlplus / as sysdba <<FIN!

connect / as sysdba

set echo on

--
-- Configure the task to run for at most 30 minutes.  The value of
the
-- TIME_LIMIT parameter determines the total time allowed for a task
execution.
--

select parameter_value
from   dba_advisor_parameters
where  task_name = 'SYS_AUTO_SQL_TUNING_TASK' and
       parameter_name = 'TIME_LIMIT';

exec dbms_sqltune.set_tuning_task_parameter( -
  'SYS_AUTO_SQL_TUNING_TASK', 'TIME_LIMIT', 1800);
```

```
select parameter_value
from   dba_advisor_parameters
where  task_name = 'SYS_AUTO_SQL_TUNING_TASK' and
       parameter_name = 'TIME_LIMIT';

--
-- Run the task immediately
--

exec dbms_sqltune.execute_tuning_task('SYS_AUTO_SQL_TUNING_TASK');

FIN!
```

10) **Note:** In your case, the task executes quickly because the workload to take into account is really small. However, you could use the `interrupt_task.sh` script from another session to stop the task, should it last too long.

```
[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$ cat interrupt_task.sh
#!/bin/bash

cd /home/oracle/solutions/Automatic_SQL_Tuning

export ORACLE_SID=orcl

export ORACLE_HOME=/u01/app/oracle/product/11.1.0/db_1

export
PATH=/u01/app/oracle/product/11.1.0/db_1/bin:/bin:/usr/bin:/usr/loca
l/bin:/usr/X11R6/bin:/usr/java/jdk1.5.0_11/bin

sqlplus / as sysdba <<FIN!

connect / as sysdba

set echo on

--
-- Interrupt the task
--

exec dbms_sqltune.interrupt_tuning_task('SYS_AUTO_SQL_TUNING_TASK');

FIN!

[oracle@edrsr33p1-orcl Automatic_SQL_Tuning]$
```

11) Ensure that you disable automatic implementation of SQL profiles to clean up your environment.

a) On the EM Server page, click Automated Maintenance Tasks.

b) Check that Status is set to Enabled, and click Configure.

c) Click the Configure button next to Automatic SQL Tuning.

d) Select No for “Automatic Implementation of SQL Profiles.”

e) Then, click Apply.