SUPINFO International University

INSTITUTE OF INFORMATION TECHNOLOGY

Oracle Technologies - DBA 10g

Practice

Oracle DBA 10g

Objectives

Managing the Database
Administrating Database Users and Managing Schema Objects
Manipulating Database Data
Securing the Oracle Database
Auditing the Oracle Database
Monitoring and Management
Managing Undo
Backup and Recovery
Recovery Manager
Managing Data Recovery

Version 1.3
Last update: 27/04/2011
Use: SUPINFO
Author: Pierre Vetter

Table des matières

1. ORACLE DATABASE ADMINISTRATION 3
1.1. STARTING WITH THE VIRTUAL MACHINE 3
1.1.1. The tool 3
1.1.2. Configuration (mandatory): 3
2. DATABASE MANAGEMENT 6
3. MANAGING TABLESPACES 8
4. ADMINISTRATING DATABASE USERS 10
5. MANAGING SCHEMA OBJECTS 13
6. MANIPULATING DATABASE DATA 16
6.1. DATA PUMP 16
6.2. SQL*LOADER 17
7. CREATING AND USING PASSWORD PROFILES 18
8. ENABLING AUDITING 19
9. MONITORING AND MANAGEMENT 21
9.1. MANAGING DATABASE PERFORMANCE 21
9.1.1. Repairing Invalid Objects 21
9.1.2. Repairing Unusable Indexes 23
9.1.3. Automating Statistics Collection 24
9.2. MONITORING ORACLE 25
9.2.1. Generating an ADDM Report 25
9.2.2. Configuring Alerts 26
10. MANAGING UNDO 27
10.1. CREATING AN UNDO TABLESPACE WITH DATABASE CONTROL 27
10.2. MONITORING UNDO WITH SQL*PLUS 28
11. BACKUP AND RECOVERY 29
11.1. CONFIGURING THE DATABASE FOR BACKUP AND RECOVERY 29
11.2. BACKING UP AN ORACLE DATABASE 30
11.2.1. Part One 30
11.2.2. Part Two 30
11.3. RECOVERING ORACLE DATABASES 32
11.3.1. Part One 32
11.3.2. Part Two 32
11.3.3. Part Three 34
12. RECOVERY MANAGER 36
12.1. RECOVERY MANAGER CONFIGURATION 36
12.2. USING RECOVERY MANAGER 38
13. MANAGING DATA RECOVERY 41
13.1. RECOVERING FROM NONCRITICAL LOSSES 41
13.2. DATABASE RECOVERY 43
13.2.1. Part One 43
13.2.2. Part Two 45

1. Oracle Database Administration

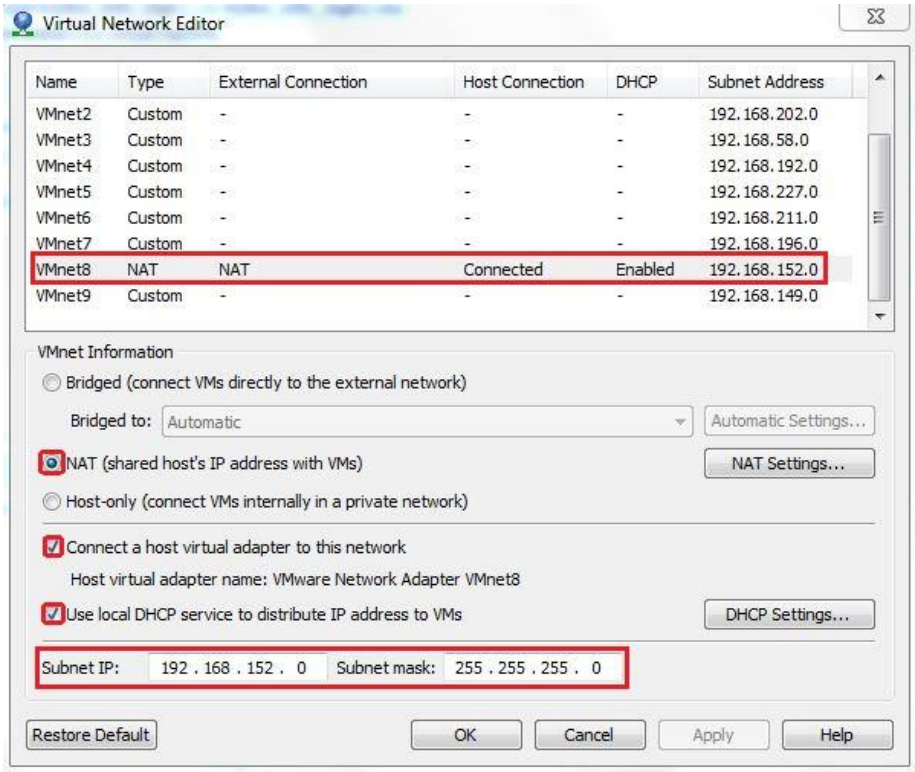
1.1. Starting with the Virtual Machine

1.1.1. The tool

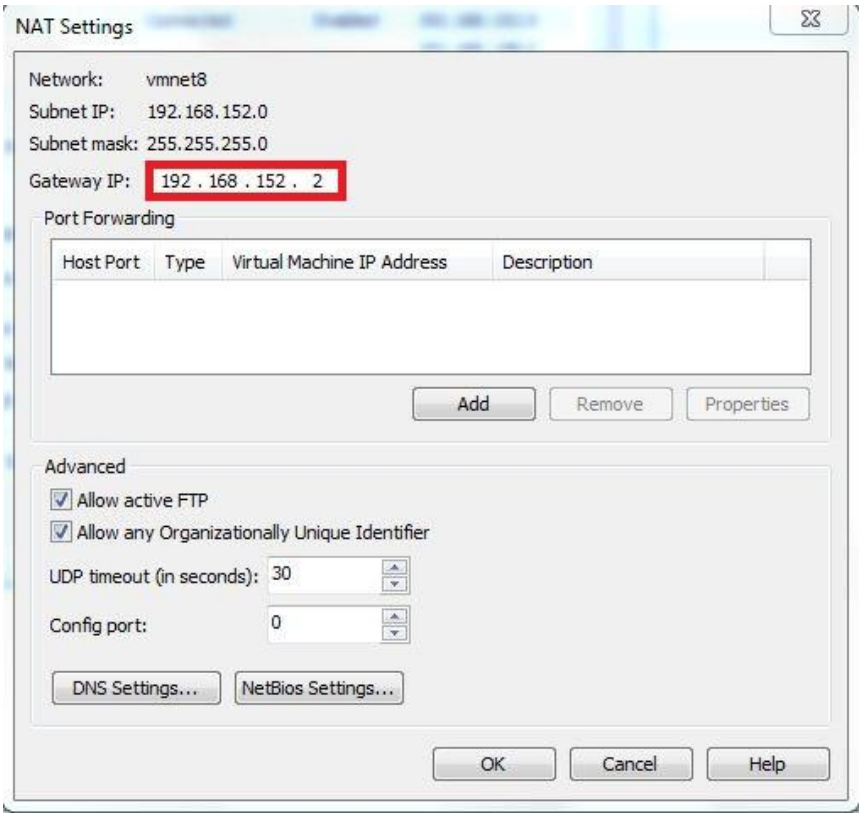
- Name : ORA_LIN_10gR1_v2.3
- Where : ftp-ssc.supinfo.com/Labs/CONTENTS/ORACLE or Local Files Sharing
- OS version : Redhat Enterprise Edition Linux 3
- Oracle Version : Oracle Database 10gR1
- OS credentials : root / rootoracle
  oracle / oracle
- Oracle Credentials : sys/oracle as sysdba

1.1.2. Configuration (mandatory):

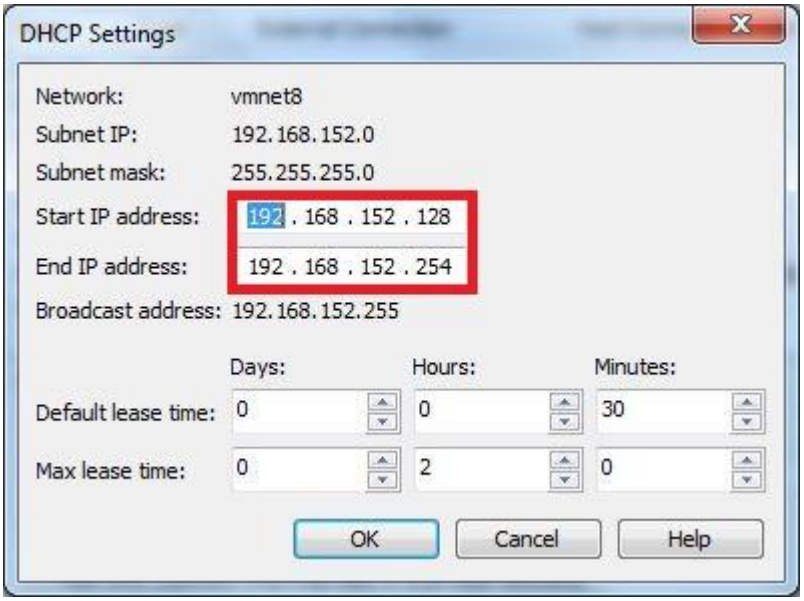
Download and extract ORA_LIN_10gR1_v2.3.rar.
BEFORE starting the VM, you have to configure Virtual Network.
Launch VMware Workstation and browse to the Virtual Network Editor in Edit menu.
Choose VMnet8 and change the Subnet IP to 192.168.152.0 and mask to 255.255.255.0.
Be sure that other options are configured like below and click Apply.



Verify in NAT Settings… that changes have been applied:



Then check changes in DHCP settings…:



Now browse to your local hosts file (WIN+r then %WINDIR%\system32\drivers\etc) and open it with administrative privilege to add the following line:

192.168.152.3 vmware.labo-oracle.com

Then add the .vmx file to VMware. Be sure to have at least 512Mo memory allowed (768 or 1024 are recommended).

You can now launch the VM, but when asked, choose **KEEP** or **I MOVED IT**.

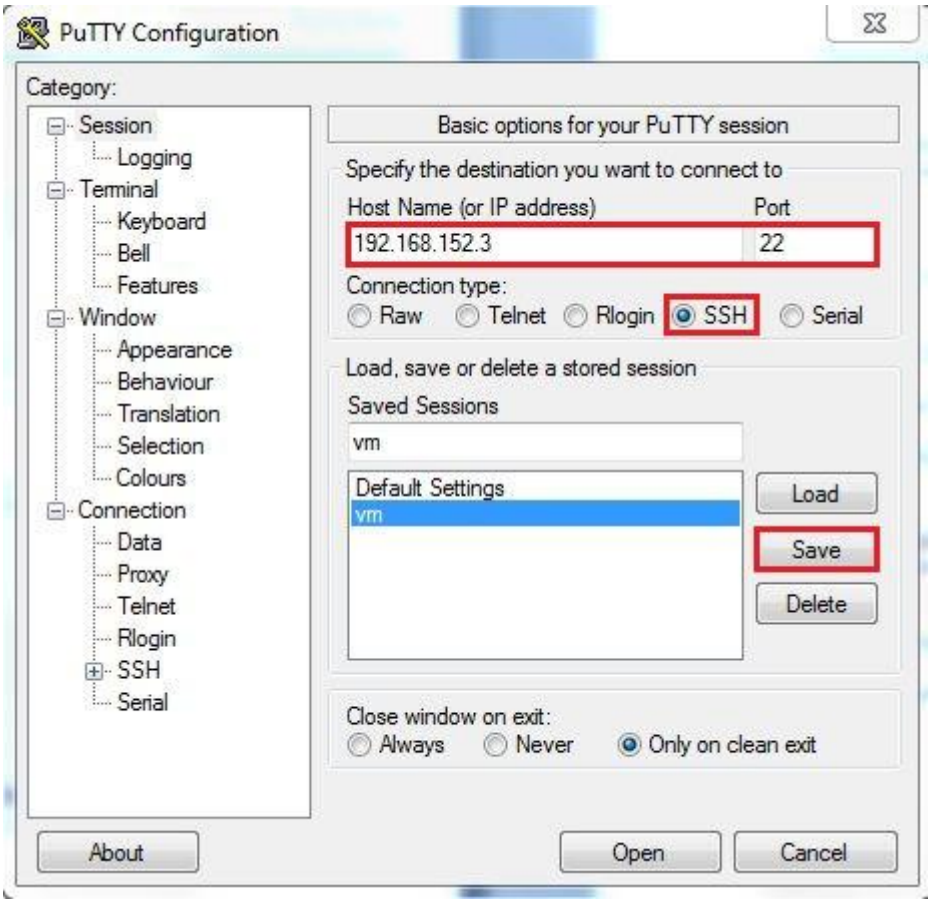
Be sure to respect above indications, else your VM won't work.

Then log in with oracle OS account (password: oracle) and execute the following command in this order:

1. lsnrctl start
2. sqlplus / as sysdba then startup and exit in sql prompt
3. emctl start dbconsole
4. isqplusctl start

Your tool is now fully functional.

For more easiness (copy/paste from host OS,…) you can connect to the VM with putty configured as below:



2. Database Management

Background: Your system administrator asks that you stop all Oracle services in preparation for system maintenance. After maintenance is completed, restart all Oracle services.

1. Start the Oracle Listener using command-line tools.
2. Start Database Control.
3. Start *i*SQL*Plus process.
4. Connect to Database Control, with the following URL : http://vmware.labo-oracle.com:5500/em. When prompted, log in with the following information:
   username: sys AS SYSDBA
   password: oracle

5. Start the Oracle Database 10*g* instance.

Enter host credentials (username *oracle*, password *oracle*).
Enter database credentials (username *sys*, password *oracle*, connect as SYSDBA). Click OK.
When prompted to confirm startup/shutdown, click Yes.

6. View information in the instance's alert log. Click Alert Log Content from the Related Links region of Database Control. From this page, you can see the most recent entries in the instance's alert log. Included in the alert contents are any initialization parameters that are set to a nondefault value when the instance is started. Locate the nondefault initialisation parameters for your instance.

```
Starting up ORACLE RDBMS Version: 10.1.0.2.0.
processes = 250
shared_pool_size = 100663296
large_pool_size = 8388608
java_pool_size = 50331648
...
```

7. View initialization parameters. Navigate to the Administration properties page by clicking the Administration link near the top of the page.

Click All Initialization Parameters in the Instance region of the Administration properties page. Do not change any of the parameters (you will customize the instance by changing parameters in later lessons).

8. Stop the Oracle Listener using Database Control.
9. Shut down the database instance using Database Control.
10. Stop *i*SQL*Plus.
11. Stop Database Control.

3. Managing Tablespaces

Background: You will be supporting a new inventory application with your database. The installation instructions for the application instruct you to create a tablespace to hold data for the new application.

1. Use Database Control to view all tablespaces in your database. For each tablespace, record the tablespace name, type, size and percent used.

Tablespace Name	Type	Size (Mo)	% Used
EXAMPLE	PERMANENT	150	53,21
SYSAUX	PERMANENT	210	99,73
SYSTEM	PERMANENT	440	00,43
TEMP	TEMPORARY	20	85
UNDOTBS	UNDO	30	98,54
USERS	PERMANENT	5	66,25

2. View all datafiles in your database. For each datafile record the file name, tablespace name, current size, autoextend status, and maximum file size (if autoextend is enabled).

File Name	Tablespace Name	Current Size (Mo)	Autoextend	Maximum Size (Mo)
example01.dbf	EXAMPLE	150	ON	32767
sysaux01.dbf	SYSAUX	210	ON	32767
system01.dbf	SYSTEM	440	ON	32767
temp01.dbf	TEMP	20	ON	32767
undotbs01.dbf	UNDOTBS1	30	ON	32767
users01.dbf	USERS	5	ON	32767

- The SYSTEM tablespace is over 90 % full. Based on the information you've just collected, should you be concerned?
  - No, there is no need for concern at this time. The data file associated with the SYSTEM tablespace has autoextend enabled and can grow considerably before transactions in that tablespace will fail due to lack of space.
- Why is autoextend an attribute of the data file rather than the tablespace?
  - Tablespaces may have up to 1,024 data files, with each data files existing on a separate disk or mount point. The ability to autoextend and the maximum size of a data file depend on the file system that the data file resides on, and might be different for different data files belonging to the same tablespace.

3. Create a new tablespace to hold information for the inventory application. Characteristics for the new tablespace are:

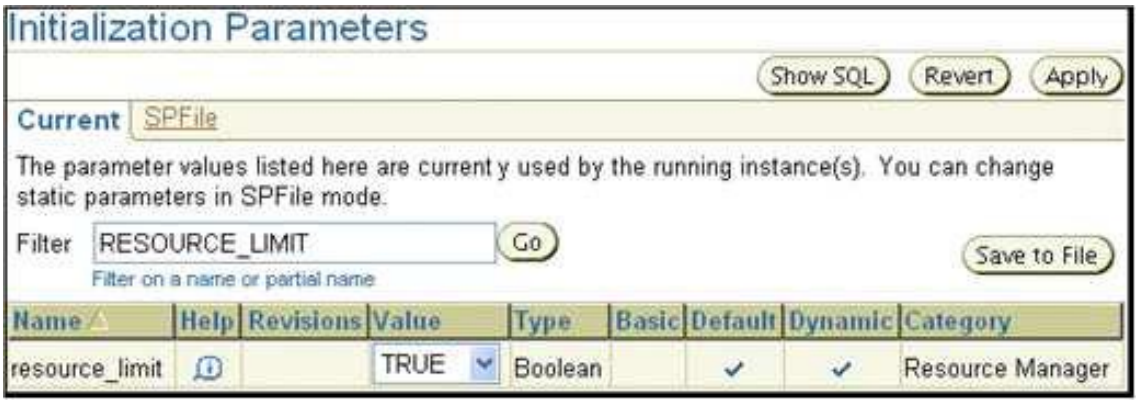
Parameter	Value
Tablespace name	INVENTORY
File size	50 MB
Extent management	Local
Autoextend	Disabled
Type	Permanent Extent
Allocation	Automatic
Status	Read Write
Segment space management	Auto
File name	inventory01.dbf
Enable Logging	Yes
File directory	Default
Thresholds	Use the default values

4. Administrating Database Users

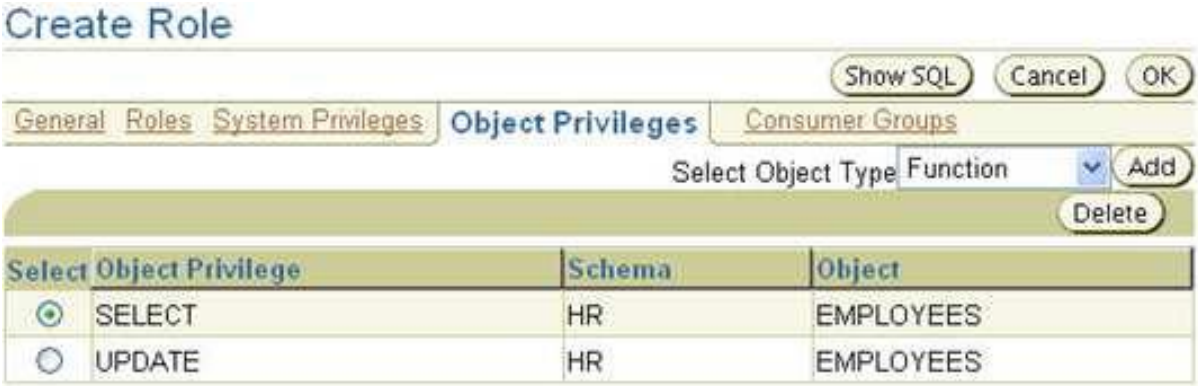
- Create and manage database user accounts
- Create and manage roles
- Grant and revoke privileges
- Control resource usage by users

1. Create a profile named **HRPROFILE** that limits idle time to 15 minutes. Leave all other fields set to **DEFAULT**.

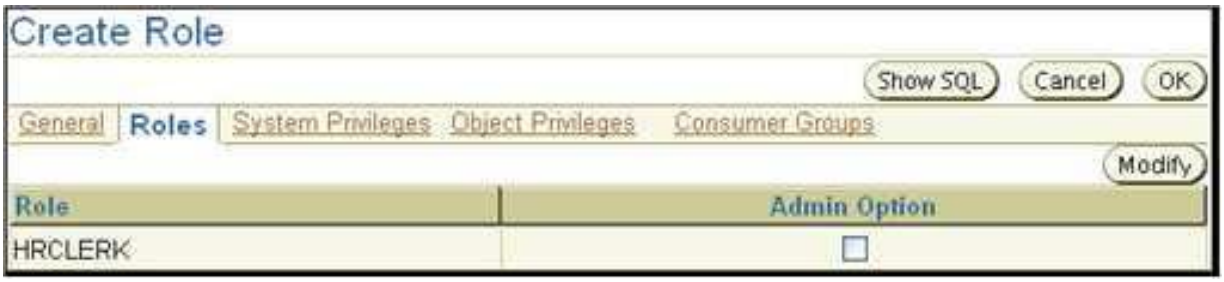
2. Set the initialization parameter **RESOURCE_LIMIT** to **TRUE** so that your profile limits will be enforced.



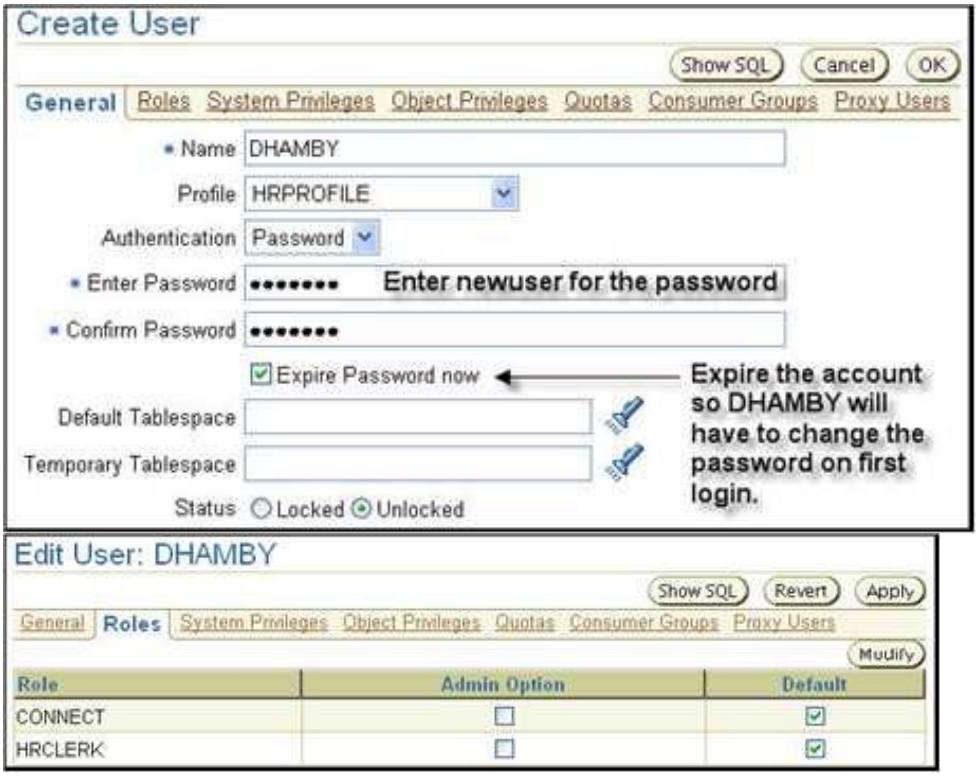
3. Create a role named **HRCLERK** without authentication and with **SELECT** and **UPDATE** permissions on the hr.employees table. This role will be used for clerks of the HR department.



4. Create a role named **HRMANAGER** with **INSERT** and **DELETE** permissions on the hr.employees table. Grant the **HRCLERK** role to the **HRMANAGER** role. This role will be used by managers of the HR department.



5. Create an account for David Hamby (DHAMBY), a new HR clerk. His profile is **HRPROFILE**, his password is newuser and this one expires immediately.



6. Create an account for Jenny Goodman (JGOODMAN), the HR new manager. His profile is HRPROFILE, his password is newmanager and this one expire immediately.

7. Connect to the database as user DHAMBY using SQL*Plus. Attemps to select from the hr.employees table.

```
CONNECT dhamby/newuser
ERROR:
ORA-28001: the password has expired
Changing password for dhamby
New password: oracle
Retype password: oracle
Password changed
Connected.
SQL> SELECT salary FROM hr.employees;
```

8. New attempt to delete a record from the hr.employees table. You may get an error.

9. Connect to the database as JGOODMAN and attempt to select and then delete (employee_id = 143 for example) from the hr.employees table.

10. Roll back the delete operation because this was only a test.

- When you created the new users you did not select a default temporary tablespace. What determines which tablespaces the new users will use?
  - The system defined default permanent and temporary tablespace.
- You did not grant the **CREATE SESSION** system privilege to either of the new users, but they can both connect to the database. Why?
  - Because Enterprise Manager automatically assigned the **CONNECT** role to the new users, and **CREATE SESSION** is contained within that role.

11. Create a new user account to own database objects for a new inventory application. The username should be inventory with a password of verysecure. Make the user's default tablespace the **INVENTORY** tablespace. Grant the user the **CONNECT** and **RESOURCE** role.

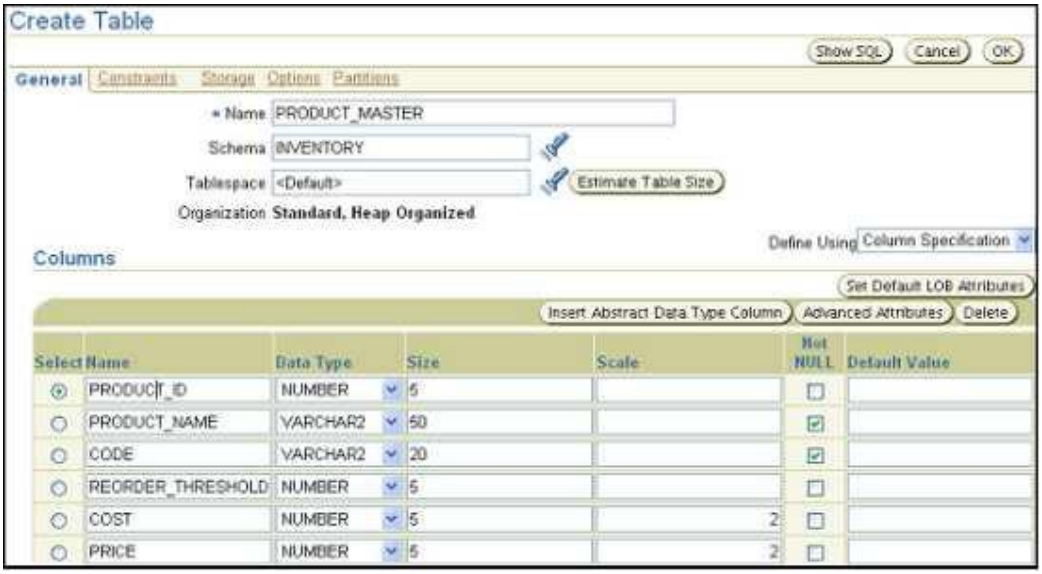
12. Leave one of the new users (RPANDYA) connected to the database during the next lesson. Verify that the user is automatically logged out after fifteen minutes.

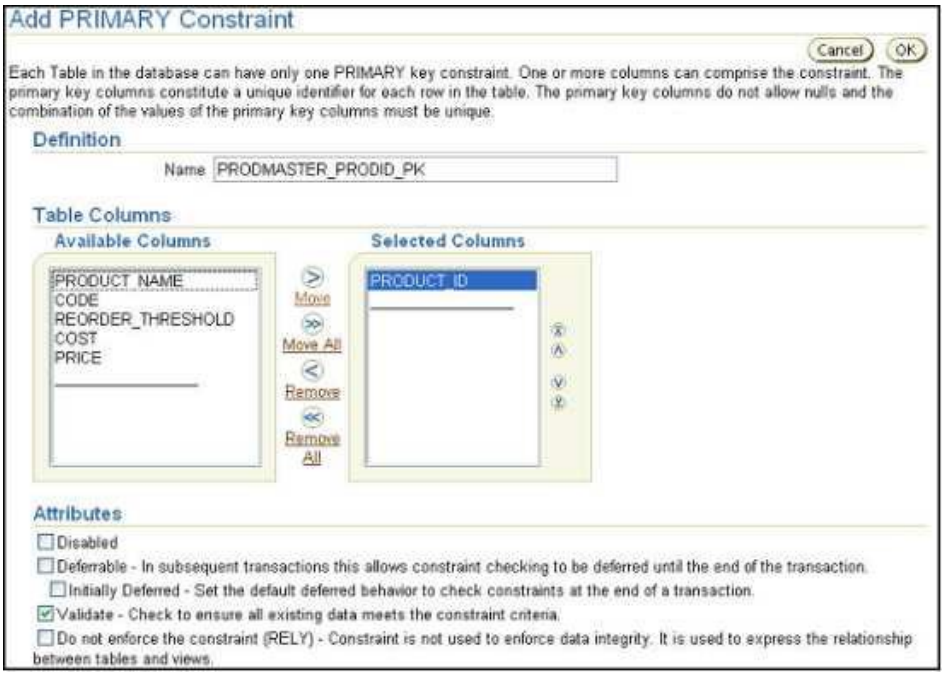
5. Managing Schema Objects

- Create and modify tables
- Define constraints
- Create indexes and views

1. In the **INVENTORY** tablespace, create the **PRODUCT_MASTER** table, in the **INVENTORY** schema, using Enterprise Manager. The specifications of the table are:

Column name	Type	Information
PRODUCT_ID	number(5)	This is the primary key field
PRODUCT_NAME	varchar2(50)	With a Not Null constraint
CODE	varchar2(20)	With a Not Null constraint
REORDER_THRESHOLD	number(5)	With a check constraint ensuring that the number is always greater than zero
COST	number(5,2)	
PRICE	number(5,2)	





2. In the **INVENTORY** tablespace, create the **PRODUCT_ON_HAND** table, in the **INVENTORY** schema. The specifications of the table are:

Column name	Type	Information
ON_HAND_ID	number(5)	This is the primary key field
PRODUCT_ID	number(5)	This field should have a foreign key constraint linking it to the product_id field in the product_master table.
QUANTITY	number(5)	
WAREHOUSE_CITY	varchar2(30)	

3. Then create the OBSOLETE_PRODUCTS table. The specifications of the table are:

Column name	Type	Information
PRODUCT_ID	number(5)	This is the primary key field
PRODUCT_NAME	varchar2(50)	with a Not Null constraint
CODE	varchar2(20)	with a Not Null constraint
COST	number(5,2)	
PRICE	number(5,2)	

- When you click OK to create the index, you switch to a list of indexes for the **INVENTORY** schema. Why are there four indexes when you've created only one?
  - Because each of the **PRIMARY KEY** constraints that you created, automatically created an index with the name of the primary key.

4. In the **INVENTORY** tablespace, create an index on the PRODUCT_NAME and CODE columns of the PRODUCT_MASTER table in the **INVENTORY** schema.

5. Create an index on the PRODUCT_ID and QUANTITY columns of the PRODUCT_ON_HAND table.

6. You receive an update of the inventory application that requires you add two columns to the PRODUCT_MASTER table:
   - PRIMARY_SOURCE of datatype VARCHAR2(50)
   - SECONDARY_SOURCE, VARCHAR2(50).

7. The inventory application also requires you add the column LAST_UPDATE of datatype DATE in PRODUCT_ON_HAND table.

8. Add a column named OBSOLETED of datatype DATE to the OBSOLETE_PRODUCTS table.

9. You receive another update for the inventory application. This update instructs you to drop the OBSOLETE_PRODUCTS table and add a column OBSOLETED to the PRODUCT_MASTER table.

10. Then, you have to create a view named WAREHOUSE_VW that shows (in order):
    - The name of the product (product_name)
    - The amount of the product on hand (quantity)
    - The warehouse city name (warehouse_city)

6. Manipulating Database Data

- Manipulate data through SQL
- Use Data Pump to export data
- Use Data Pump to import data
- Load data with SQL*Loader

6.1. Data Pump

1. Create a directory to be used by Data Pump. For example, from a command prompt:

```
mkdir /home/test_dtp
```

2. Using SQL*Plus, log on to your instance as user SYSTEM and create a table to be used for testing Data Pump.

```
CREATE TABLE dtp_test AS
 SELECT * FROM all_users;

SELECT COUNT(*) FROM dtp_test;
```

3. Still within SQL*Plus, create the Oracle directory to be used by Data Pump and grant all users permission to read and write to the directory.

```
CREATE DIRECTORY dtp_dir AS '/home/test_dtp';
GRANT read, write ON DIRECTORY dtp_dir TO PUBLIC;
```

4. Log on to Database Control as user SYSTEM.

5. From the database control home page, click the Maintenance tab.

6. In the Utilities section, click the Export To File link.

7. Select Tables and enter an operating system username and password with read/write permissions on the directory specified in step 3.

8. On the next screen, click Add, and specify SYSTEM as the schema and dtp_test as the table. Click Select and Next.

9. Then, choose the directory DTP_DIR as the location for the logfile, and click Next.

10. Choose dtp_dir as the location and click Finish.

11. In the following screen, take a look at the script that has been generated to see how you would run the job using the Data Pump API, and click Submit Job.

12. On the Status window, click OK.

13. From an operating system prompt, navigate to the directory specified in step 3. There will be a file EXPDAT01.DMP, which is the export dump file, and a lagfile, EXPDAT.LOG. Examine the logfile to check that the job did complete successfully.

6.2. SQL*Loader

Use SQL*Loader and Database Control to load data from text files into the PRODUCT_MASTER and PRODUCT_ON_HAND tables. Files are in the /home/oracle/labs directory.

1. Use the control file lab_05_01_04_a.ctl to load data from the text file lab_05_01_04_a.dat into the PRODUCT_MASTER table.

2. Use the control file lab_05_01_04_f.ctl to load data from the text file lab_05_01_04_f.dat into the PRODUCT_ON_HAND table.

1. Using a text editor, create a SQL*Loader controlfile, called STREAMIN.CTL, as follow:

```
load data
infile 'STREAMIN.DAT' "str '\n'"
append
into table dtp_test
fields terminated by ','
(username, user_id, created)
```

Note that the “append” keyword allows SQL*Loader to insert into a table that already contains rows.

2. Using a text editor as before, create the input datafile, to be called STREAMIN.DAT, as follows:

```
John,100,11-MAY-11
Damir,200,11-MAY-11
McGraw,9999,11-MAY-11
```

3. From an operating system prompt, issue this command to load data without Database Control:

```
sqlldr userid=system/oracle control=STREAMIN.CTL
```

4. Log in to your instance with SQL*Plus, and confirm that the three new rows have been inserted into the table:

```
SELECT * FROM dtp_test;
```

7. Creating and Using Password Profiles

- During this labs, you will have to:
  - Create a profile with strict limits
  - Assign a user to this profile
  - Demonstrate the effect

1. Connect to your database as user **SYSTEM** with Database Control.

2. From the database home page, take the Administration tab and then the Profiles link in the Security section.

3. Click Create to reach the Create Profile window, and enter STRICT as the profile name. Take the Password link to reach the password controls window.

4. Set limits for your STRICT profile. Users assigned to this profile will have to change their passwords after two weeks, and they will have three days to do so. A password can only ever be used once, and after two failed login attempts the account will be locked, but only for one minute.

5. Click Show SQL, examine the **CREATE PROFILE** command being generated, and click Return.

6. Click OK to create the profile.

7. Return to the Administration page of Database Control, and take the Users link in the Security section.

8. On the Users windows, find the **SYSTEM** user, select his radio button, and click Edit.

9. In the Edit User: SYSTEM window, select the STRICT profile and expire the password.

10. Click Show SQL, examine the **ALTER USER** command being generated, and click Return.

11. Click Apply to make the change.

12. Connect to your database as user **SYSTEM** user SQL*Plus. Note that you immediately receive a warning that the password has expired, and that you are already in the "grace" period of three days.

13. Attempt to change the password to the value it is already (in the example here, it is ORACLE) with

```
ALTER USER system IDENTIFIED BY oracle;
```

you will receive an error,

```
ORA-28007: the password cannot be reused
```

14. Enter a different password. (Remember it!) For the following examples, it was set to MANAGER.

15. Attempt to connect three times with the wrong password. At the third attempt, you will be told that the account is locked. Wait at least one minute, and then connect with the correct password.

16. Tidy up by assigning **SYSTEM** back to the default profile and dropping the STRICT profile.

```
ALTER USER system PROFILE default;

DROP PROFILE strict;
```

8. Enabling Auditing

- During this labs, you will have to:
  - Enable both database auditing and fine-grained auditing
  - Use data dictionary views to see the results

1. Connect to your database as **SYSDBA** using SQL*Plus.

2. Set the **AUDIT_TRAIL** instance parameter to enable auditing to the data dictionary. As this is a static parameter, you must use the **SCOPE** clause and restart the instance.

```
CONN / as sysdba
Connected.

ALTER SYSTEM SET audit_trail=db SCOPE=spfile;
System altered.

STARTUP FORCE;
```

3. Connect to your database as user **SYSTEM** using SQL*Plus.

4. Create a table and insert some rows as follows:

```
CREATE TABLE audit_test (name VARCHAR2(10), salary NUMBER);

INSERT INTO audit_test VALUES ('McGraw',100);

INSERT INTO audit_test VALUES ('Hill',200);
```

5. Enable database auditing of access to the table.

```
AUDIT select, update ON system.audit_test;
```

6. Execute some statements against the table.

```
SELECT * FROM audit_test;

UPDATE audit_test SET salary = 50
WHERE name='McGraw';
```

7. Query the DBA_AUDIT_TRAIL view to see the results of the auditing.

```
SELECT username, userhost, os_username,ses_actions, obj_name
FROM dba_audit_trail;

USERNAME USERHOST OS_USERNAME SES_ACTIONS   OBJ_NAME
---------------- -------------- -------------------- --------------------  ----------
SYSTEM      ESI\VM      VM\Oracle        ---------SS-----      AUDIT_TEST
```

This shows that Oracle user **SYSTEM**, while logged onto a machine called VM in the domain called ESI as the local user Oracle, executed one or more **SELECT** and **UPDATE** statements successfully against the table called **AUDIT_TEST**.

8. Create an FGA policy to capture all SELECTs against the AUDIT_TEST table that read the SALARY column, if the salary retrieved is greater than 100, with this procedure call:

```
EXEC dbms_fga.add_policy (
object_schema    => 'system',
obect_name       => 'audit_test',
policy_name      => 'high_sal',
audit_condition => 'salary > 100',
audit_column     => 'salary',
statement_types => 'select');
```

9. Run some queries against the table.

```
SELECT * FROM audit_test;

SELECT salary FROM audit_test WHERE name='Hill';

SELECT salary FROM audit_test WHERE name='McGraw';

SELECT name FROM audit_test;
```

10. Query the fine-grained audit trail.

```
SELECT os_user, db_user, sql_text
FROM dba_fga_audit_trail;

OS_USER    DB_USER  SQL_TEXT
-------------- -------------- ------------------------------------
VM\Oracle SYSTEM    SELECT * FROM audit_test
VM\Oracle SYSTEM    SELECT salary FROM audit_test WHERE
                    name='Hill';
```

Note that only the first and second question from step 9 generated audit records, and that the actual statement used can be retrieved.

11. Tidy up by canceling the database auditing, dropping the FGA policy, and dropping the table.

```
NOAUDIT select, update ON system.audit_test;

 EXEC dbms_fga.drop_policy (
object_name => 'audit_test',
policy_name => 'high_sal');

DROP TABLE audit_test;
```

9. Monitoring and Management

- Troubleshoot invalid and unusable objects
- Gather optimizer statistics

9.1. Managing Database Performance

9.1.1. Repairing Invalid Objects

1. Using SQL*Plus, connect to your database as user SYSTEM.

2. Create a user TESTUSER to be used for this exercise; grant him the DBA privilege.

```
GRANT dba TO testuser IDENTIFIED BY testuser;
```

3. Connect as TESTUSER, and create some objects.

```
CONN testuser/testuser
Connected.

CREATE TABLE testtab(n1 NUMBER, d1 DATE);
Table created.

INSERT INTO testtab VALUES (1, SYSDATE);
1 row created.

CREATE OR REPLACE VIEW v1 AS
SELECT d1 FROM testtab;
View created.

CREATE OR REPLACE PROCEDURE p1 AS
cnt NUMBER;
BEGIN
SELECT COUNT(*) INTO cnt FROM testtab;
END;
/
Procedure created.
```

4. Confirm the status of the objects.

```
SELECT object_name, object_type, status
FROM user_objects;
```

They will have the STATUS of VALID.

5. Perform a DDL command on the table.

```
ALTER TABLE testtab DROP COLUMN d1;
```

6. Re-run the query from step 4. Note that both the procedure and the view are now INVALID.

7. Recompile the procedure.

```
ALTER PROCEDURE p1 COMPILE;
Procedure altered.
```

This succeeds, because dropping a column does not mean that the procedure (which does not actually reference any columns by name) cannot run.

8. Re-compile the view.

```
ALTER VIEW v1 COMPILE;
Warning: View altered with compilation errors.
```

9. To diagnose the problem, query the DBA_DEPENDENCIES view.

```
SELECT referenced_name, referenced_owner, referenced_type
FROM dba_dependencies
WHERE name = 'V1';
```

This show that the view refers to a table, TESTTAB, and a nonexistent object called D1.

10. To pinpoint the exact problem, retrieve the code on which the view is based.

```
SELECT text
FROM user_views
WHERE view_name = 'V1';
```

The problem is now apparent: the view references a valid table, but the column it needs no longer exists.

11. To fix the problem, add the column back to the table and recompile.

```
ALTER TABLE testtab ADD (d1 DATE);
Table altered.

ALTER VIEW v1 COMPILE;
View altered.
```

12. Confirm that all the objects are now valid by re-running the query from step 4.

13. Tidy up by dropping view and procedure (the table will be used in the next exercise).

```
DROP VIEW v1;

DROP PROCEDURE p1;
```

9.1.2. Repairing Unusable Indexes

1. In your SQL*Plus session, connect as TESTUSER and create two indexes.

```
CREATE INDEX d1_idx ON testtab(d1);

CREATE INDEX n1_idx ON testtab(n1);
```

2. Confirm the index creation and status. Both will be VALID.

```
SELECT index_name, status
FROM user_indexes;
```

3. Move the table.

```
ALTER TABLE testtab MOVE;
```

4. Run the query from step 2 again. The move of the table, which changed any rowids, will have rendred the indexes unusable.

5. Rebuild one index, using the NOLOGGING and ONLINE options.

```
ALTER INDEX n1_idx
REBUILD ONLINE NOLOGGING;
```

6. Connect to your database as user SYSTEM using Database Control.

7. From the database home page, take the Administration tab and then the Indexes link in the Schema section.

8. In the Search section of the Indexes window, enter TESTUSER as the Schema, and click Go. This will show the two indexes on the TESTTAB table, one of which, D1_IDX, is still unusable.

9. Select the radio button for the unusable index, select Reorganize in the Actions drop-down box, and click Go to launch the Reorganize Objects Wizard.

10. Click Next, leave all the options on default, and click Next again to generate the reorganization script and reach the Impact Report window. This should confirm that there is sufficient free space for the operation to proceed. Click Next to proceed.

11. On the Reorganize Objects: Schedule window, leave everything on default to run the job immediately, and click Next to reach the Review window.

12. In the Review window, click Submit Job to rebuild the index.

13. In your SQL*Plus session, confirm that the index is now valid by running the query from step 2.

9.1.3. Automating Statistics Collection

1. Connect to your database as user TESTUSER using Database Control.

2. Take the Administration tab, then the Jobs link in the Scheduler section to reach the Scheduler Jobs window.

3. Click Create to reach the Create Job window. In the Credential section, enter the Name as Analyze testtab, and leave everything else on default.

4. In the Command section, replace the sample code with this:.

```
BEGIN
dbms_stats.gather_table_stats(
 ownname => 'TESTUSER',
 tabname => 'TESTTAB',
 estimate_percent => 100,
 cascade => true,
 method_opt => 'for all indexed columns size auto');
END;
```

5. Take the Schedule link to reach the Schedule window. Leave everything on default, to run the job once only right away, and return to the Scheduler Jobs window.

6. Take the Run History link, and you will see that the job has succeeded.

7. In your SQL*Plus session, set your NLS_DATE_FORMAT session parameter to show the full time and confirm that statistics were indeed collected.

8. Tidy up by connecting as user SYSTEM and dropping the TESTUSER schema.

```
DROP USER 'TESTUSER' CASCADE;
```

9.2. Monitoring Oracle

- View performance metrics
- Set warning and critical alert thresholds
- Use baseline metrics, tuning, diagnostic advisors, the Automatic Database Diagnostic Monitor, the Automatic Workload Repository

9.2.1. Generating an ADDM Report

1. Connect to your database as user SYSTEM with SQL*Plus.

2. Force the creation of an AWR snapshot.

```
EXEC dbms_workload_repository.create_snapshot;
```

3. Simulate a workload by creating a table and running this anonymous PL/SQL block to generate some activity:

```
CREATE TABLE tmptab AS
SELECT * FROM all_objects;

BEGIN
FOR i IN 1..10 LOOP
INSERT INTO tmptab
SELECT * FROM all_objects;
DELETE FROM tmptab;
END LOOP;
COMMIT;
END;
/
```

4. Repeat the command from step 2 to generate another snapshot.

5. Connect to your database as user SYSTEM using Database Control.

6. Take the Advisor Central link in the Related Links section on the database home page. The first report listed will be the ADDM report generated as a result of the snapshot.

7. Select the radio button for the latest ADDM report, and click View Result.

8. Study the report. In the example, the worst problem (as shown in the graph) is disk I/O; the findings at the bottom of the screen show the causes of this I/O problem. Double-click the findings links for further details.

9. Tidy up by dropping the TMPTAB table.

```
DROP TABLE tmptab;
```

NOTE: Your results depend on your database configuration; if the report shows no problem, edit the code in step 3 to force more activity, and repeat the exercise.

9.2.2. Configuring Alerts

1. Connect to your database with Database Control as user SYSTEM.
2. From the database home page, take the Manage Metrics link in the Related Links section.
3. Click Edit Thresholds to reach the Edit Thresholds window.
4. Scroll down to the "User Commits (per second)" alter, and set the warning and critical value to 1 and 4. These are artificially low thresholds that it will be simple to cross. Click OK to save this change.
5. Connect to your database as user SYSTEM with SQL*Plus, and issue the COMMIT command a few times quickly.

```
COMMIT;
Commit complete.

/
Commit complete.

/
Commit complete.

/
Commit complete.
```

6. In your Database Control session, within a few seconds you will see that the alert has been raised.
7. Tidy up by returning to the Edit Thresholds window and clearing the threshold values.

10. Managing Undo

- Monitor and administer undo
- Configure undo retention
- Guarantee undo retention
- Use undo advisor
- Describe the relationship between undo and transactions
- Size the undo tablespace

10.1. Creating an Undo Tablespace with Database Control

1. Connect to your instance as user SYSTEM with Database Control.

2. From the Administration tab in the Storage section, take the Tablespaces link.

3. Click Create.

4. Enter UNDO2 as the tablespace name, and set the radio buttons to Extent Management "Locally Managed", Type "Undo", and Status "Read Write".

5. At the bottom of the screen, click Add to specify a datafile.

6. Enter UNDO2-01.DBF as the File Name, leave everything else on default, and click Continue.

7. On the Create Tablespace screen, click Show SQL, and study the statement used to create your undo tablespace. Click Return to return to the Create Tablespace screen, and click OK to create the tablespace.

8. Connect to your instance as user SYSTEM through SQL*Plus.

9. Run this query, which will return one row for each tablespace in your database, and note that your new tablespace has contents UNDO, meaning that it can only be used for undo segments, and that retention is NOGUARANTEE, a topic covered shortly.

```
SELECT tablespace_name, contents, retention
FROM dba_tablespaces;
```

10. Run this query, which will return one row for each rollback or undo segment in your database, and note that a number of undo segments have been created automatically in your new undo tablespace, but that they are all offline. Also note that the names of the automatic undo segments are in the form of "_SYSSMU*n*$", where *n* is the undo segment number (usn).

```
SELECT tablespace_name, segment_name, status
FROM dba_rollback_segs;
```

10.2. Monitoring Undo with SQL*Plus

1. Connect to your instance with SQL*Plus as user SYSTEM.

2. Set up your session for displaying dates conveniently.

```
ALTER SESSION SET nls_date_format = 'dd/mm/yy hh24:mi:ss';
```

3. Query V$UNDOSTAT as follows:

```
SELECT begin_time, end_time, undoblks, maxquerylen, ssolderrcnt, nospaceerrcnt
FROM v$undostat;
```

4. Interpret the results of the query. Note that the view has one row per ten-minute interval, showing you how much undo was generated, in blocks; how long the longest query was, in seconds; and whether there were any "snapshot too old" errors, or errors from transactions running out of undo space.

5. Calculate the minimum necessary size in bytes for your undo tablespace that will prevent errors, given your current activity data, with this query:

```
SELECT
 (SELECT MAX(undoblks)/600 * MAX(maxquerylen)
FROM v$undostat)
*
(SELECT value
FROM v$parameter
WHERE name='db_block_size')
FROM dual;
```

11. Backup and Recovery

- Describe the basics of database backups, restore and recovery
- List the types of failure that may occur
- Describe ways to tune instance recovery
- Identify the importance of checkpoints, redo log files, and archived redo log files
- Configure ARCHIVELOG mode

11.1. Configuring the Database for Backup and Recovery

1. Using SQL*Plus, connect as user SYSTEM.

2. Disable checkpoint tuning by setting the FAST_START_MTTR_TARGET parameter to zero.

```
ALTER SYSTEM SET fast_start_mttr_target = 0;
```

3. Simulate a workload by creating a table and starting a transaction.

```
CREATE TABLE t1 AS
SELECT * FROM all_objects WHERE 1=2;

INSERT INTO t1 SELECT * FROM all_objects;
```

4. Run a query to see how much work would be required to recover the instance if it crashed right now.

```
SELECT recovery_estimated_ios, actual_redo_blks, estimated_mttr
FROM v$instance_recovery;
```

5. The query shows how many read/write operations would be required on the datafiles and how many blocks on redo would have to be processed during an instance recovery. The ESTIMATED_MTTR column shows, in seconds, how long the recovery would take.

6. Commit the transaction, and re-run the query from Step 3. Note that nothing much has changed: COMMIT has no effect on DBWn and will not advance the checkpoint position.

7. Issue a manual checkpoint.

```
ALTER SYSTEM checkpoint;
```

   This may take a few seconds to complete, as DBWn flushes all changed blocks to disk.

8. Re-run the query from Step 4. Note that the RECOVERY_ESTIMATED_IOS and ACTUAL _REDO_BLKS columns have dropped substantially, perhaps to zero. The ESTIMATED_MTTR column may not have reduced, because this column is not updated in real time.

9. Tidy up by dropping the table.

```
DROP TABLE t1;
```

NOTE: Your results depend on your database configuration; if the report shows no problem, edit the code in step 3 to force more activity, and repeat the exercise.

11.2. Backing Up an Oracle Database

11.2.1. Part One

1. Connect using Database Control as user SYSTEM. From the database home page, take the Schedule Backup link in the Maintenance section to reach theSchedule Backup: Strategy window.

2. In the Backup Strategy drop-down box, select Customized and click the Whole Database radio button. In the Host Credentials section, enter an operating system username and password. Click Next to reach the Schedule Backup: Options window.

3. Leave everything on defaults: a full, online backup with all archive logs. Click Next to reach the Schedule Backup: Settings window.

4. Leave everything on default to schedule a disk backup to your flash recovery area directory. Click Next to reach the Schedule Backup: Schedule window.

5. Leave everything on default to run the backup immediately as a one-off job. Click Next to reach the Schedule Backup: Review window.

6. Click the Submit Job button to launch the backup.

7. Click the View Job button to check how the job is running, and then refresh the browser window to monitor progress.

11.2.2. Part Two

1. Connect with SQL*Plus as user SYSTEM.

2. Issue this command:

```
ALTER DATABASE BACKUP controlfile TO trace;
```

3. Locate your user dump destination.

```
SHOW parameters user_dump_dest;
```

4. From an operating system prompt, change to the user dump destination directory.

5. Identify the newest file in the directory. For example, on Windows us dir /od or on Unix, ls -ltr. The newly generated trace file will be the last file listed. Open the trace file with any editor you please and study the contents. The critical section is the CREATE CONTROLFILE command, which will resemble this:

```
CREATE CONTROLFILE REUSE DATABASE
"ORCL" NORESETLOGS ARCHIVELOG
MAXLOGFILES 16
MAXLOGMEMBERS 3
MAXDATAFILES 100
MAXINSTANCES 8
MAXLOGHISTORY 454
LOGFILE
GROUP 1 (
        '/u01/app/oracle/product/10.1.0/oradata/
        orcl/REDO01.LOG',
         '/u01/app/oracle/product/10.1.0/oradata/
        orcl/REDO01B.LOG'
) SIZE 10M,
GROUP 2 (
        '/u01/app/oracle/product/10.1.0/oradata/
        orcl/REDO02.LOG',
        '/u01/app/oracle/product/10.1.0/oradata/
        orcl/REDO02B.LOG'
) SIZE 10M,
GROUP 3 (
        '/u01/app/oracle/product/10.1.0/oradata/
        orcl/REDO03.LOG',
        '/u01/app/oracle/product/10.1.0/oradata/
        orcl/REDO03B.LOG'
) SIZE 10M
-- STANDBY LOGFILE
DATAFILE
'/u01/app/oracle/product/10.1.0/oradata/
orcl/SYSTEM01.DBF',
'/u01/app/oracle/product/10.1.0/oradata/
orcl/UNDOTBS01.DBF',
'/u01/app/oracle/product/10.1.0/oradata/
orcl/SYSAUX01.DBF',
'/u01/app/oracle/product/10.1.0/oradata/
orcl/USERS01.DBF',
'/u01/app/oracle/product/10.1.0/oradata/
orcl/EXAMPLE01.DBF',
'/u01/app/oracle/product/10.1.0/oradata/
orcl/UNDO2-01.DBF'
CHARACTER SET WE8MSWIN1252
;
```

11.3. Recovering Oracle Databases

11.3.1. Part One

1. Connect to your database with SQL*Plus, and ensure that your controlfile is multiplexed with this query:

```
SELECT * FROM v$controlfile;
```

   This query must return at least two rows. If it does not, multiplex your controlfile.

2. Simulate damage to a controlfile by crashing the database and renaming one of your controlfiles.

   Note that on Windows you may have to stop the Windows service before Windows will let you rename the file, and start it again afterward.

3. Issue a startup command. The startup will stop in nomount mode, with an "ORA-00205: error in identifying controlfile, check alert log for more info" error message.

4. Copy your surviving controlfile to the name and location of the file you renamed.

5. Issue another startup command, which will be successful.

11.3.2. Part Two

1. Using SQL*Plus, connect to your database as user SYS with SYSDBA privilege.

```
CONNECT / AS sysdba;
```

2. Observe the state of your online logs with the following query:

```
SELECT group#,status,member FROM v$logfile
ORDER BY group#;

        GROUP# STATUS MEMBER
        ----------------------------------------------
        /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO01.LOG
        /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO01B.LOG
        /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO02.LOG
        …
```

3. Confirm that you do have at least two members of each group and that all the members have the STATUS column on NULL, as in the example here. If any groups do not have two members, multiplex them immediately by following the instructions given in first exercise. If any members do not have a STATUS of NULL, execute the command a few times to cycle through the groups, and then re-run the query.

```
ALTER SYSTEM switch logfile;
```

4. Shut down the database:

```
shutdown immediate;
```

5. Using an operating system command, simulate media failure by deleting one of the members.

```
! rm /u01/app/oracle/product/10.1.0/oradata/orcl/ REDO01.LOG
```

6. Start up the database and simulate user activity by performing a few log switches.

```
startup;

ALTER SYSTEM switch logfile;

ALTER SYSTEM switch logfile;

ALTER SYSTEM switch logfile;
```

7. Check the state of your logfile members.

```
SELECT group#,status,member
FROM v$logfile
ORDER BY group#;

        GROUP# STATUS MEMBER
        ----------------------------------------------
        INVALID /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO01.LOG
        /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO01B.LOG
        /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO02.LOG
        /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO02B.LOG
        /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO03.LOG
        /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO03B.LOG
```

Note that the missing file is now marked as being INVALID.

8. Connect to your database as user SYSTEM, using Database Control.

9. From the database home page, take the Administration tab, and then the Red Logs link in the Storage section.

10. If the group with the problem (group number 1 in the example shown) is not INACTIVE, use the Switch Logfile choice in the Actions drop-down list and click Go to force log switches until it is inactive.

11. Clear the logfile group by selecting its radio button using the Clear Logfile choice in the Actions drop-down list, and clicking Go.

12. In your SQL*Plus session, confirm that the problem has been fixed.

```
SELECT group#,status,member
FROM v$logfile
ORDER BY group#;

        GROUP# STATUS MEMBER
        ---------- ------- ----------------------------
        /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO01.LOG
        /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO01B.LOG
        /u01/app/oracle/product/10.1.0/oradata/orcl/
        REDO02.LOG
```

11.3.3. Part Three

1. Connect to your database as user SYSTEM using SQL*Plus, and create a tablespace.

```
create tablespace noncrit datafile
'/u01/app/oracle/product/10.1.0/oradata/orcl/noncrit.dbf' size 2m;
```

2. Create a table within the new tablespace and insert a row into it.

```
create table ex1133 (c1 date) tablespace noncrit;

insert into ex1133 values(sysdate);

commit;
```

3. Using Database Control, connect to your database as user SYSTEM.

4. From the database home page, take the Maintenance tab, then the Schedule Backup link in the Backup/Recovery section.

5. In the Schedule Backup: Strategy window, select Customized in the Backup Strategy drop-down box.

6. Select the Tablespaces radio button, and click Next.

7. In the Schedule Backup: Tablespaces window, click Add.

8. In the Tablespaces: Available Tablespaces window, select the radio button for your new NONCRIT tablespace, and click Select.

9. In the Schedule Backup: Tablespaces window, click Next.

10. In the Schedule Backup: Options window, leave everything on defaults and click Next.

11. In the Schedule Backup: Settings window, leave everything on defaults and click Next.

12. In the Schedule Backup: Schedule window, leave everything on defaults and click Next to schedule an immediate backup.

13. In the Schedule Backup: Review click Submit to run the backup.
    Simulate a disk failure by corrupting the new datafile. You can use any editor you please, such as vi. Make sure that the characters deleted are at the start of the file, to ensure that the file header is damaged.

14. Confirm that the file is damaged by attempting to query the table:

```
select * from ex203;

select * from ex203
 *
ERROR at line 1:
ORA-01578: ORACLE data block corrupted
(file # 7, block # 9)
ORA-01110: data file 7:
 '/home/oracle/product/10.1.0/oradata/orcl/
NONCRIT.DBF'
```

If the damage is not yet apparent, repeat Step 13 until it is.

15. In your Database Control session, take the Maintenance tab from the database home page, and then the Perform Recovery link in the Backup/Recovery section.

16. In the Perform Recovery: Type window, select Datafiles in the Object Type drop-down box, and click Next.

17. In the Perform Recovery: Datafiles window, the new datafile will be listed. Select it, and click Next.

18. In the Perform Recovery: Review, leave everything on defaults and click Submit.

19. When the operation has completed, return to your SQL*Plus prompt and bring the file online, specifying it by name or by number.

```
alter database datafile 7 online;
```

20. Confirm that the tablespace and the tables within it are now usable, with no loss of data.

```
select * from ex203;

 C1
 ---------
21-OCT-04
```

21. Tidy up the database.

```
drop tablespace noncrit including contents and datafiles;
```

12. Recovery Manager

12.1. Recovery Manager Configuration

In this exercise you will become familiar with configuring **RMAN** and viewing the current configuration.

1. Connect to your database as the target database in the default NOCATALOG mode as the SYSTEM user.

```
rman TARGET system/oracle NOCATALOG

Recovery Manager: Release 10.1.0.2.0 - Production
Copyright (c) 1995, 2004, Oracle. All rights reserved.
connected to target database: ORCL (DBID=1045444042)
using target database controlfile instead of recovery catalog
```

2. Use the **RMAN SHOW ALL** command to generate a listing of the **RMAN** configuration settings.

```
SHOW ALL;

RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; #default
CONFIGURE CONTROLFILE AUTOBACKUP OFF; #default
…
```

3. Configure **RMAN** to automatically back up the control file and SPFILE whenever a backup of the database or data files is taken.

```
CONFIGURE CONTROLFILE AUTOBACKUP ON;

new RMAN configuration parameters:
CONFIGURE CONTROLFILE AUTOBACKUP ON;
new RMAN configuration parameters are successfully stored
```

4. Use the Enterprise Manager Database Control Console to set the backup retention policy to a recovery window of 2 days. Log in to the Database Control Console as the SYSTEM user. If this is your first time logging in to EM as the SYSTEM user, you will need to click "I agree" on the License Agreement screen.

5. Go to the Maintenance page.

6. Click Configure Backup Settings link under Backup/Recovery region. If the Select User browser window appears, click Cancel.

7. Click the Policy tab. If the Select User browser window appears, choose oracle from the list, then click OK. If the oracle user does not appear on the list, click Cancel.

8. Scroll down to the Retention Policy region.

9. Choose "Retain backups that are necessary for a recovery to any time within the specified number of days and specify a value of 2. To save the modified details, entrer the Host Credentials of oracle/oracle and click OK.

10. Verify the backup retention policy setting using the **RMAN** utility and the **SHOW** command.

```
SHOW RETENTION POLICY;

RMAN configuration parameters are:
CONFIGURE RETENTION POLICY TO RECOVERY
WINDOW OF 2 DAYS;
```

12.2. Using Recovery Manager

In this exercise you will become familiar with using the Recovery Manager utility to perform and manage backups.

1. Using *SQL*Plus* or the **EM** Database Control Console, verify the database is in **ARCHIVELOG** mode. If not, alter the database to enable archiving of the online redo logs.

2. Go to the Maintenance page.

3. To check using **EM**, click Configure Recovery Settings link under Backup/Recovery region. See if **ARCHIVELOG** Mode is checked in the Media Recovery region.

4. To check using SQL*Plus, use the ARCHIVE LOG LIST command.

```
ARCHIVE LOG LIST

Database log mode              No Archive Mode
Automatic archival             Disabled
Archive destination            USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence     67
Current log sequence           69
```

5. The database is not currently archiving. Correct this problem with the following commands, or use Enterprise Manager.

```
SHUTDOWN IMMEDIATE

Database closed.
Database dismounted.
ORACLE instance shut down.

STARTUP MOUNT

ORACLE instance started.
Total System Global Area 188743680 bytes
...
Redo Buffers 262144 bytes
Database mounted.

ALTER DATABASE ARCHIVELOG;
Database altered.

ALTER DATABASE OPEN;
Database altered.

ARCHIVE LOG LIST;
Database log mode Archive Mode     Automatic archival Enabled
Archive destination                USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence         67
Next log sequence to archive       69
Current log sequence               69
```

6. Connect to your database using RMAN in the default NOCATALOG mode as the SYSTEM user.

```
rman TARGET system/oracle NOCATALOG
```

7. Use the RMAN REPORT command to generate a listing of your database structure.

```
REPORT SCHEMA;
```

8. Obtain a listing of all database backup sets that currently exist.

```
LIST BACKUP OF DATABASE;
LIST COPY OF DATABASE;
```

9. Use RMAN to back up the data files belonging to the EXAMPLE and USERS tablespaces. Be sure you also make a copy of the current control file and server parameter file.
   Your backups should be placed in the $HOME/DONTTOUCH/ directory and should use the format df_%d_%s_%p.bak for the file names.

```
BACKUP AS BACKUPSET
FORMAT '$HOME/DONTTOUCH/df_%d_%s_%p.bak'
TABLESPACE USERS, EXAMPLE;

Starting backup at 19-FEB-04
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=235 devtype=DISK
channel ORA_DISK_1: starting full datafile backupset
channel ORA_DISK_1: specifying datafile(s) in backupset
input datafile fno=00005 name=/u01/app/oracle/oradata/orcl/example01.dbf
input datafile fno=00004 name=/u01/app/oracle/oradata/orcl/users01.dbf
channel ORA_DISK_1: starting piece 1 at 19-FEB-04
channel ORA_DISK_1: finished piece 1 at 19-FEB-04
piece handle=/home/oracle/DONTTOUCH/df_ORCL_10_1.bak comment=NONE
channel ORA_DISK_1: backup set complete, elapsed time: 00:00:15
Finished backup at 19-FEB-04

Starting Control File and SPFILE Autobackup at 19-FEB-04
Piece handle=/u01/app/oracle/flash_recovery_area/ORCL_EDRSR12P1/
tobackup/2004_02_19/o1_mf_s_525279729_09j4q1m0_.bkp comment=NONE
Finished Control File and SPFILE Autobackup at 19-FEB-04
```

10. Create an image copy of two data files. Use the following information:
    - Copy the SYSTEM tablespace and name the copy sys01.cpy with a tag of SYSTEM01
    - Copy the SYSAUX tablespace and name the copy sysaux01.cpy with a tag of SYSAUX01
    - The files should be written to the Flash Recovery Area.

```
BACKUP AS COPY
FORMAT 'sys01.cpy'
TABLESPACE SYSTEM
TAG=SYSTEM01;

Starting backup at 19-FEB-04
using channel ORA_DISK_1
...

BACKUP AS COPY
FORMAT 'sysaux01.cpy'
TABLESPACE SYSAUX
TAG=SYSAUX01;

Starting backup at 19-FEB-04
using channel ORA_DISK_1
...
```

11. Obtain a listing of all database files that have not been backed up.

```
REPORT NEED BACKUP;

RMAN retention policy will be applied to the command
RMAN retention policy is set to recovery window of 2 days
Report of files whose recovery needs more than 2 days of archived logs
File Days Name
---- ----- ----------------------------------------------
2    89    /u01/app/oracle/oradata/orcl/undotbs01.dbf
```

12. Take a full backup of the database, including archived logs. Use as little space as possible to store the backup.

```
BACKUP AS COMPRESSED BACKUPSET DATABASE PLUS ARCHIVELOG;
```

13. Managing Data Recovery

13.1. Recovering from Noncritical Losses

1. Get the name of the default temporary tablespace from the DATABASE_PROPERTIES view and the data files associated with this tablespace from DBA_TEMP_FILES.

```
SELECT * FROM database_properties
WHERE property_name like '%TEMP%';
…
SELECT file_name FROM dba_temp_files
WHERE tablespace_name = 'TEMP';

FILE_NAME
------------------------------------------------------------
/u01/app/oracle/oradata/orcl/temp01.dbf
```

2. Delete the temporary tablespace data files at the operating system level.

```
rm /u01/app/oracle/oradata/orcl/temp01.dbf
```

3. Connect to the database as a SYSDBA user, shutdown the instance, and restart it.

```
sqlplus "/as sysdba"
…
shutdown immediate;
…
startup;
…
```

4. Perform a query against a table in the database that involves sorting of data. What happens?

```
SELECT text FROM dba_source
WHERE owner='SYSMAN'
ORDER BY name, type, line;

SELECT text FROM dba_source
                        *
ERROR at line 1:
ORA-01157: cannot identify/lock data file 201 - see DBWR trace file
ORA-01110: data file 201: '/u01/app/oracle/oradata/orcl/temp01.dbf'
```

5. Attempt to take the temporary tablespace offline before recovering it. What happens?

```
ALTER TABLESPACE temp OFFLINE;

ALTER TABLESPACE temp OFFLINE
*
ERROR at line 1:
ORA-03217: invalid option for alter of TEMPORARY TABLESPACE
```

6. Drop the temporary tablespace. What happens?

```
DROP TABLESPACE temp;

DROP TABLESPACE temp
*
ERROR at line 1:
ORA-12906: cannot drop default temporary tablespace
```

7. Create a new temporary tablespace named TEMP1 containing a single data file named temp1.dbf which is 100 MB in size.

```
CREATE TEMPORARY TABLESPACE temp1
TEMPFILE '/u01/app/oracle/oradata/orcl/temp1.dbf' SIZE 100M;

Tablespace created.
```

8. Change the database default temporary tablespace to TEMP1.

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp1;

Database altered.
```

9. Retry your query that involved a sort operation. What happens now?

```
SELECT text FROM dba_source
WHERE owner='SYSMAN'
ORDER BY name, type, line;

123045 rows selected.
```

10. Drop the temporary tablespace with the missing data files. You must remove the tablespace and the file associated using a single SQL command.

```
DROP TABLESPACE temp INCLUDING CONTENTS AND DATAFILES;

Tablespace dropped.

!ls $ORACLE_BASE/oradata/orcl
control01.ctl example01.dbf redo03.log temp1.dbf
control02.ctl redo01.log sysaux01.dbf undotbs01.dbf
control03.ctl redo02.log system01.dbf users01.dbf
```

11. Perform a backup of the database.

```
rman target /
…
BACKUP DATABASE;

using target database controlfile instead of recovery catalog
allocated channel: ORA_DISK_1
channel ORA_DISK_1: sid=254 devtype=DISK
…
```

13.2. DATABASE RECOVERY

13.2.1. Part One

This exercise simulates the need to recover a database to a point in the past because of the introduction of questionable data.

1. As user system/oracle create the table HR.DEPARTMENTS2 by selecting all rows from the HR.DEPARTMENTS table. Confirm that the new table exists, and record the total number of rows in the table.
   View the active log by querying V$LOG. Perform a log switch when finished.

```
sqlplus /nolog
SQL*Plus: Release 10.1.0.2.0 - Production on Thu Feb 19 03:29:16 2004
Copyright (c) 1982, 2004, Oracle. All rights reserved.

connect system/oracle as sysdba
Connected.

create table HR.DEPARTMENTS2 as select * from HR.DEPARTMENTS;
Table created.

select count(*) from hr.departments2;
COUNT(*)
----------
27

select sequence#, status from v$log;
SEQUENCE#       STATUS
----------      ----------------
65              CURRENT
63              INACTIVE
64              INACTIVE

alter system switch logfile;
System altered.
```

2. Check and record the system time and date.

```
!date
Thu Feb 19 03:31:49 PST 2004
```

3. Query V$LOG again to confirm the switch and then insert three lines into the HR.DEPARTMENTS2 table and commit. Confirm the number of row in the table. These INSERTs represent the introduction of questionable data into the table.

```
select sequence#, status from v$log;
SEQUENCE#       STATUS
----------      ----------------
65              ACTIVE
66              CURRENT
64              INACTIVE

insert into hr.departments2 values (280, 'DUMMY1','','');
insert into hr.departments2 values (290, 'DUMMY2','','');
insert into hr.departments2 values (300, 'DUMMY3','','');
commit;
select count(*) from hr.departments2;
COUNT(*)
----------
30
```

4. Shutdown the database, and restart it in mount mode.

```
shutdown immediate;

startup mount;
ORACLE instance started.
```

5. Using RMAN, recover the database to a point in time before the new data was introduced using the information you recorded before the inserts were performed.

```
rman target /
Recovery Manager: Release 10.1.0.2.0 - Production
Copyright (c) 1995, 2004, Oracle. All rights reserved.
connected to target database: ORCL (DBID=1045444042)

run {
set until time "TO_DATE('04-FEB-19:03:31:49','YY-MON-DD:HH24:MI:SS')";
restore database;
recover database;
}

executing command: SET until clause
using target database controlfile instead of recovery catalog
…
```

6. Open the database with the RESETLOGS option and confirm the recovery.

```
sqlplus system/oracle as sysdba
SQL*Plus: Release 10.1.0.2.0 - Production on Fri Feb 19 03:36:49 2004
Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options

alter database open resetlogs;
Database altered.

select count(*) from hr.departments2;
COUNT(*)
----------
27
```

13.2.2. Part Two

This exercise simulates the recovery to a point in the past using the EM interface to RMAN.

1. Determine the current log sequence and write it down.

```
select sequence#, status from v$log;
SEQUENCE# STATUS
---------- ----------------
0 UNUSED
0 UNUSED
1 CURRENT
```

2. Verify the row count for the HR.DEPARTMENTS2 table.

```
select count(*) from hr.departments2;
COUNT(*)
----------
27
```

3. Force a log switch and verify the switch has taken place. Perform several inserts into the HR.DEPARTMENTS2 table and commit the changes. Verify the new row count. Then exit your SQL*Plus session.

```
alter system switch logfile;
System altered.

select sequence#, status from v$log;
SEQUENCE# STATUS
---------- ----------------
2 CURRENT
0 UNUSED
1 ACTIVE

insert into hr.departments2 values (280, 'DUMMY1','','');
insert into hr.departments2 values (290, 'DUMMY2','','');
insert into hr.departments2 values (300, 'DUMMY3','','');
commit;

select count(*) from hr.departments2;
COUNT(*)
----------
30
```

4. Using Enterprise Manager while logged in as a SYSDBA user, recover the database to a point in time before the new data was introduced using the information you recorded before the inserts were performed.
   - From the Database Control home page, click on the *Maintenance* folder tab, and then click on *Perform Recovery*.
   - Select *Whole Database* from the Object Type pull down list
   - Select *Recover to the current time*
   - Make sure the proper Host Credentials are supplied (**`oracle`**/**`oracle`**).
   - Click Next to continue.
   - On the Recovery Wizard page, wait one to two minutes, and then click Refresh. Or, view the alert log until the message: "Completed: ALTER DATABASE MOUNT" appears, and then click Refresh.
   - To continue recovery, click Perform Recovery on the Database status page, which indicates the database is currently unavailable.

- On the Perform Recovery: Credentials page, enter both the Host Credentials and the Database Credentials for a SYSDBA user. Then click Continue.
- On the Perform Recovery: Type page, the same information should be entered, as was entered for step before. Click Next to continue.
- On the Perform Recovery: Point-in-time page, change the recovery to 'Recover to a prior point-in-time'. Then click on the Sequence button and type in the sequence number of the current log before the inserts were committed.
- Click Next to continue.
- On the Perform Recovery: Rename page, click Next to restore the datafile to their original location.
- On the Perform Recovery: Review page, review the recovery steps and the RMAN script that will be run.
- Then click Submit.

5. Wait until the Operation Succeeded message is displayed, then use SQL*Plus to verify that the recovery was successful by checking the row count in the HR.DEPARTMENTS2 table.

```
connect system/oracle as sysdba
Connected.

select count(*) from hr.departments2;
COUNT(*)
----------
27
```