## PURPOSE:

This lab provides you with the opportunity to gain hands-on experience with important DBMS features – **Point in Time Recovery and Flashback Recovery** – as implemented in Oracle.

## BACKGROUND:

Flashback Recovery is a very useful feature that allows an authorized user to recover an object at a particular point in time. Conceptually, Flashback Recovery is similar to a substantially more powerful version of the *Recycle Bin* in MS Windows*.* We will look at one aspect of Flashback Recovery - recovering a dropped table and its Primary Key constraint implemented as an index. The feature also has much greater capabilities (for example, recovering data at a point in time, or recovering the entire database).

See Chapter 18 of the Oracle19c Backup and Recovery Guide at:

<https://docs.oracle.com/en/database/oracle/oracle-database/19/bradv/rman-performing-flashback-dbpitr.html#GUID-94B90ECF-69C9-40D4-A318-DC4AE9776557>

Certain situations are suited for using point-in-time recovery or flashback features to return the database or database object to its state at a previous point in time.

Some typical situations include the following:

* A user error or corruption removes needed data or introduces corrupted data. For example, a user or DBA might erroneously delete or update the contents of one or more tables, drop database objects that are still needed during an update to an application, or run a large batch update that fails midway.
* A database upgrade fails or an upgrade script goes awry.
* A complete database recovery after a media failure cannot succeed because you do not have all of the needed redo logs or incremental backups.

…

**About Point-in-Time Recovery and Flashback Features**

Database point-in-time recovery (DBPITR) and Flashback features enable you to recover your database to a prior point in time.

DBPITR is the most basic solution to unwanted database changes. It is sometimes called incomplete recovery because it does not use all of the available redo or completely recover all changes to your database. In this case, you restore a whole database backup and then apply redo logs or incremental backups to re-create all changes up to a point in time before the unwanted change.

If unwanted database changes are extensive but confined to specific tablespaces, then you can use tablespace point-in-time recovery (TSPITR) to return these tablespaces to an earlier system change number while the unaffected tablespaces remain available.

If unwanted database changes are limited to specific tables or table partitions, then you can use a previously created RMAN backup to return only these objects to a point in time before the unwanted changes occurred.

Oracle Database also provides a set of features collectively known as Flashback Technology that supports viewing past states of data, and winding and rewinding data back and forth in time, without requiring the restore of the database from backup. Depending on the changes to your database, Flashback Technology can often reverse the unwanted changes more quickly and with less impact on database availability.

## REQUIREMENTS:

Complete the following tasks, submit your solution via BrightSpace and demo the lab to your lab professor. You can earn a maximum of 2 marks towards your lab mark for a complete, correct and on-time submission.

## TASKS:

**Part 1: Setting Parameters:**

1. **Logon to SYS AS SYSDBA:** 
   1. To use the FLASHBACK RECOVERY feature you need to confirm a few parameters have been set to appropriate values. You will (hopefully) recall from an earlier lab that the SPFILE and PFILE files contain initialization parameters. These parameters are read at startup and remain in effect until altered or the instance is restarted – depending on the type of parameter being considered. Running the following query ***SELECT COUNT(\*) FROM V$PARAMETER;*** indicates there are 366 parameters – though there are more than another 900 parameters that aren't documented (i.e., only to be changed on instruction from Oracle support).
   2. Enter: ***SHOW PARAMETERS UNDO*** to determine the values for parameters UNDO\_MANAGEMENT, UNDO\_TABLESPACE, and UNDO\_RETENTION:
      1. List your values for:
         1. UNDO\_MANAGEMENT: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
         2. UNDO\_TABLESPACE: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
         3. UNDO\_RETENTION: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
      2. The SHOW PARAMETER command in SQLPlus in essence issues a ***SELECT NAME FROM V$PARAMETER*** query. The V$PARAMETER view however, contains much more valuable information. Let's take a look at some of this information.

<https://docs.oracle.com/cd/B28359_01/server.111/b28320/dynviews_2085.htm#REFRN30176>

* + - 1. First, use SQLPlus column formatting to change the display format of V$PARAMETER's NAME, VALUE and DESCRIPTION columns to A20 (e.g., COLUMN NAME FORMAT a20).
      2. Enter:

***SELECT name, value, isses\_modifiable, issys\_modifiable, description***

***FROM V$PARAMETER***

***WHERE lower(name) like 'undo%';***

**Paste the result below:**

* + - 1. Repeat the above command changing the ***like 'undo%'*** condition to ***like 'nls\_date%'***

**Paste the result below:**

* + - 1. Enter: ***SELECT SYSDATE FROM DUAL;***

**Paste the result below:**

* + 1. Complete the following table based on your recent results:

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Value** | **ISSES\_**  **Modifiable** | **ISSYS\_**  **Modifiable** |
| UNDO\_MANAGEMENT |  |  |  |
| UNDO\_TABLESPACE |  |  |  |
| UNDO\_RETENTION |  |  |  |
| NLS\_DATE\_FORMAT |  |  |  |

* + 1. Even if your UNDO\_MANAGEMENT parameter value is currently 'AUTO', enter the following commands (Note: some will fail):

***ALTER SYSTEM SET UNDO\_MANAGEMENT=AUTO;***

***ALTER SYSTEM SET UNDO\_MANAGEMENT=AUTO SCOPE=SPFILE;***

***ALTER SYSTEM SET UNDO\_MANAGEMENT=AUTO SCOPE=MEMORY;***

***ALTER SYSTEM SET UNDO\_MANAGEMENT=AUTO SCOPE=BOTH;***

***ALTER SYSTEM SET UNDO\_RETENTION = 1200 SCOPE=BOTH;***

***ALTER SESSION SET NLS\_DATE\_FORMAT ='DAY MONTH DD,YYYY';***

(Note: If your ***SHOW PARAMETERS UNDO\_MANAGEMENT*** command returned the value ***MANUAL*** instead of ***AUTO***, then you will also need to "bounce" (i.e., shutdown then start) the database.

* + 1. Confirm your changes were effective by running the following query and copying the results below:

***SELECT name, value***

***FROM V$PARAMETER***

***WHERE lower(name) like 'undo%' OR lower(name)*** ***like 'nls\_date%'*** ;

**Part 2: Creating Some Database Objects for later update & recovery tests:**

1. **Logon as a general user.** 
   1. Create a basic sequence named Student\_seq (i.e., that starts at 1 and increments by 1). Refer to this online Oracle document page for syntax help: <http://docs.oracle.com/database/121/SQLRF/statements_6017.htm#SQLRF01314>

**Note**: if the account is lacking privileges, use sqlplus (sqlplus / as sysdba) to grant the account the necessary privileges.

**Show your work below**.

Enter: ***Set Timing On***

Create a table Students with the following columns:

* + 1. Sid, with a datatype of number, -- this is the primary key, give the constraint an obvious name (e.g., St\_sid\_pk)
    2. Fname, with a datatype of varchar2(12), and
    3. Lname, with a datatype of varchar2(24)

**Show your work below**.

Run the following PL/SQL block to insert 1000 records into the ***Students*** table:

**DECLARE**

**i NUMBER := 1;**

**BEGIN**

**LOOP**

**INSERT INTO Students**

**VALUES ( Student\_seq.nextval,**

**concat('FName\_', to\_char(Student\_seq.currval)),**

**concat('LName\_', to\_char(Student\_seq.currval))**

**);**

**Commit;**

**i := i+1;**

**EXIT WHEN i>1000;**

**END LOOP;**

**END;**

**/**

**Show your work below**.

How long did it take to insert the 1000 records in the Students table (in seconds)?

* 1. Create a sequence named ***Emp\_seq*** that **starts at 1 and increments by 2**.

Create an ***Employees*** table with the following columns:

* + 1. Ee#, with a datatype of number, -- this is the primary key. Give the constraint an obvious name (e.g., Emp\_ee\_pk).
    2. First, with a datatype of varchar2(12), and
    3. Last, with a datatype of varchar2(24)

Run the following PL/SQL block to insert 1000 records into the ***Employees*** table.

**DECLARE**

**e NUMBER := 1;**

**BEGIN**

**LOOP**

**INSERT INTO Employees**

**VALUES ( Emp\_seq.nextval,**

**concat('First\_', to\_char(Emp\_seq.currval)),**

**concat('Last\_', to\_char(Emp\_seq.currval))**

**);**

**e := e+1;**

**EXIT WHEN e>1000;**

**END LOOP;**

**Commit;**

**END;**

**/**

How long did it take to insert the 1000 records in the Employees table?

How you would explain the difference in the time required to insert 1000 records in the Students and Employees tables?

* 1. Create an index on ***Students.Lname*** and an index on ***Employees.Last. Show your work below:***
  2. In an SQLPLUS session enter:  
     ALTER SESSION SET NLS\_DATE\_FORMAT = 'DD-MON-YYYY HH24:MI:SS';

Followed by

select CURRENT\_DATE from dual;

Show you work and the results below:

* 1. ***(Make sure you still have your database in ARCHIVELOG mode – as per backup lab).*** Open RMAN and create a level 0 incremental backup of the USERS tablespace (the default tablespace for your general user) ***Show your work below: Take note of the time.***
  2. Use sqlplus to perform an update of all records in the Students table to have the same Fname with the value “OOPS”. (Use: “UPDATE Students SET Fname=”OOPS”
  3. Open RMAN (if not still open) and create a level 1 incremental backup of the USERS tablespace (the default tablespace for your general user). ***Show your work below: Take note of the time.***

**Part 3: Recycle Bin Tests:**

1. **While connected the general user above:** 
   1. Enter: ***purge recyclebin***

Enter: ***show recyclebin***

* 1. Delete all of the **data** in the Employees table. Then,

Enter: ***SELECT count(\*) FROM EMPLOYEES*** to confirm the rows were deleted.

Next issue a ***ROLLBACK***

Re-enter ***SELECT count(\*) FROM EMPLOYEES*** to confirm the transaction was cancelled and the rows were recovered.

Copy your statements and results here:

* 1. Drop the Employees table,

Enter ***SHOW RECYCLEBIN***

Enter ***SELECT \* FROM RECYCLEBIN;*** These two commands should show the same results.

Copy your statements and results here:

* 1. Enter ***FLASHBACK TABLE Employees TO BEFORE DROP;***

Enter ***SELECT count(\*) FROM EMPLOYEES;***

Enter **INSERT INTO Employees VALUES ( 1, 'PK', 'Oops');**

Enter **SELECT \* FROM Employees WHERE ee# = 1;**

Copy your statements and results here:

Has the dbms permitted the above insert? What does this mean?

* 1. Drop the Employees table again.

Create a new Employees table from the Students table using the CTAS (Create Table As Select) approach described here: <http://www.techonthenet.com/sql/tables/create_table2.php>

Make sure rename the fields so that the Last -> Lname, First -> Fname, and Sid->EE#

Show your SQL and result below:

Enter ***FLASHBACK TABLE Employees TO BEFORE DROP***;

What resulted and why?

* 1. Enter ***FLASHBACK TABLE Employees TO BEFORE DROP RENAME TO Employees2;***

Confirm the table was recovered by either **selecting \* from the recovered** **table** or listing user's tables created and recovered in this lab.

**Part 4: Recovery Test:**

1. **Edit the following Recover command from the Oracle manuals and execute it in RMAN in order to recover the Students table to the original values before performing the “OOPS” update above. You will need to change the tablename, the time specification (so that it gets the correct value based on YOUR backup timings, set the destination folder to a folder on your system, and specify a filename to use for the dump file. Show your work and results:**

RECOVER TABLE SCOTT.EMP

UNTIL TIME 'SYSDATE-1'

AUXILIARY DESTINATION '/tmp/oracle/recover'

DATAPUMP DESTINATION '/tmp/recover/dumpfiles'

DUMP FILE 'emp\_dept\_exp\_dump.dat'

NOTABLEIMPORT;

**You’re Done!**