Practice

Performing Flashback Database

Practice Target

In this practice, you will perform the routine tasks involved in managing flashback database.

Practice Overview

In this practice, you will perform the following tasks:

- Use Flashback Database to undo unwanted transactions
- Retrieve important information about flashback database

Assumptions

- This practice assumes that you have `srv1` up and running from the **CDB** snapshot.

A. Using Flashback Database to Undo Data Changes

In this section of the practice, you will use flashback database to undo data changes.

1. Invoke SQL*Plus and login to the local database as sysdba

```
sqlplus / as sysdba
```

2. Check if the ARCHIVELOG mode is enabled in the database.

The ARCHIVELOG mode is enabled which means we can safely enable the flashback database without the need to restart the database.

```
ARCHIVE LOG LIST
```

3. Retrieve the value of the parameter DB_FLASHBACK_RETENTION_TARGET

The parameter is set to its default value 1440 (minutes). This is equivalent to 24 hours.

```
show parameter DB_FLASHBACK_RETENTION_TARGET
```

4. Determine if flashback database is enabled

```
SELECT FLASHBACK_ON FROM V$DATABASE;
```

5. Enable the flashback database.

```
ALTER DATABASE FLASHBACK ON;
```

6. Verify that the flashback database is enabled

```
SELECT FLASHBACK_ON FROM V$DATABASE;
```

7. Check if any flashback database log file is produced.

Flashback log files are produced under the FRA directory (under an automatically created subdirectory named "flashback").

```
SELECT NAME , BYTES/1024/1024 MB FROM V$FLASHBACK_DATABASE_LOGFILE;
```

8. Verify that the rvwr process is running.

This process is responsible of saving the flashback data into the flashback log files.

```
HOST ps -ef | grep ora_rvwr
```

9. Start Swingbench sessions on PDB1.

If you wait for a few minutes, you will observe that a new flashback log file will be produced.

Now we have the flashback running, let's create a testing table, drop it, and then use flashback database to restore it.

Note: There is a better way to recover dropped tables but we are just demonstrating using flashback database.

10. Run the following code to create a testing table.

```
ALTER SESSION SET CONTAINER=PDB1;
CREATE TABLE SOE.CUST AS SELECT * FROM SOE.CUSTOMERS;
```

11. Retrieve the current time and SCN.

We can use either of them as a recovery point reference.

```
SELECT TO_CHAR(SYSDATE,'DD-MM-YYYY HH24:MI:SS') CTIME,
DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER SCN
FROM DUAL ;
```

12. Create a restore point for the current SCN.

```
CREATE RESTORE POINT BEFORE_DROP;
```

13. Drop the testing table. This is the destructive action that we want to recover from.

```
DROP TABLE SOE.CUST PURGE;
```

14. Stop Swingbench sessions.

15. Close `PDB1`.

Because the current container is `PDB1`, only `PDB1` will shutdown by the `SHUTDOWN` command.

```
show con_name
shutdown immediate
```

16. Flashback `PDB1` to the retrieved point-in-time, the retrieved SCN, or the restore point.

You can use the `SCN_TO_TIMESTAMP` and `TIMESTAMP_TO_SCN` functions to convert a SCN to timestamp and vice versa.

Obviously, this command is much simpler and quicker than using RMAN point-in-time-recovery.

```
ALTER SESSION SET CONTAINER=CDB$ROOT;
FLASHBACK PLUGGABLE DATABASE PDB1 TO TIME TO_TIMESTAMP('<ctime>','DD-MM-YYYY
HH24:MI:SS');
FLASHBACK PLUGGABLE DATABASE PDB1 TO SCN <SCN>;
FLASHBACK PLUGGABLE DATABASE PDB1 TO RETORE POINT BEFORE_DROP;
```

17. Open `PDB1` using `RESETLOGS` option.

```
ALTER PLUGGABLE DATABASE PDB1 OPEN RESETLOGS;
```

18. Verify the lost table is recovered.

```
ALTER SESSION SET CONTAINER=PDB1;
SELECT COUNT(*) FROM SOE.CUST;
```

B. Retrieve important information about flashback database

In this section of the practice, you will retrieve important information about the flashback database.

19. Start Swingbench sessions against `PDB1`.

20. Submit the following query to retrieve the current flashback logs size and the estimated size of flashback data needed for the current target retention

The output of this query is important for making sure that the FRA has enough space for the flashback logs future growth.

```
ALTER SESSION SET CONTAINER=CDB$ROOT;

SELECT round(ESTIMATED_FLASHBACK_SIZE/1024/1024) ESTIMATED_FL_SIZE_MB
, round(FLASHBACK_SIZE/1024/1024) FLASHBACK_SIZE_MB
FROM   V$FLASHBACK_DATABASE_LOG;
```

21. Submit the following query to retrieve the time of the lowest SCN in the flashback data, for any incarnation.

This query is useful to make sure that the defined flashback database retention is honored by the database.

```
ALTER SESSION SET NLS_DATE_FORMAT='DD-MM-YYYY HH24:MI:SS';
SELECT OLDEST_FLASHBACK_SCN, OLDEST_FLASHBACK_TIME
FROM   V$FLASHBACK_DATABASE_LOG;
```

22. Submit the following query to displays statistics for monitoring the I/O overhead of logging flashback data and the estimated flashback space needed based on previous workloads.

The flashback size retrieved by the query above is for the current database workload. This view provides the `V$FLASHBACK_DATABASE_LOG.ESTIMATED_FLASHBACK_SIZE` over the past periods.

```
SELECT BEGIN_TIME, END_TIME, FLASHBACK_DATA, DB_DATA, REDO_DATA,
ROUND(ESTIMATED_FLASHBACK_SIZE/1024/1024) EST_FL_MB
FROM V$FLASHBACK_DATABASE_STAT
ORDER BY 1 DESC;
```

Cleanup

23. Stop Swingbench sessions.

24. Shutdown `srv1` and restore it from its CDB snapshot.

Summary

- Flashback database is easy to configure in Oracle database and provides a simple and effective recovery tool against logical errors.

- We should monitor the free space in the FRA and flashback statistics to make sure that the database can honor the flashback retention period.