## Practice

# Managing Undo Data

### Practice Target

In this practice you will perform the normal tasks on managing undo data in a database.

Specifically, you will learn:

- Examine and understand the undo statistics retrieved from `V$UNDOSTAT`
- Activate the Undo Advisor and retrieve its recommendations
- Manually calculate the required undo size for a specific undo retention value

### Practice Assumptions

This practice assumes that the virtual machine `srv1` is restored from the **non-CDB** snapshot and up and running.

# Preparing for the Practice

1. Open a Putty session to `srv1` as `oracle`

2. Run the following code to create a SQL script that displays the undo statistics.

```
cat > display_undo_stats.sql <<EOF
/* the query retreive the undo stats from the last two intervals */
SELECT UNDOBLKS, MAXQUERYLEN, UNEXPIREDBLKS, EXPIREDBLKS, TUNED_UNDORETENTION
RETENTION
FROM V\$UNDOSTAT
ORDER BY BEGIN_TIME DESC
FETCH FIRST 2 ROWS ONLY ;
EOF
```

3. Verify the non-CDB database is up and running.

```
sqlplus / as sysdba
SELECT CDB FROM V$DATABASE;
```

4. As `HR`, run the following code to create a large testing table named as `EMP`. This table will be used in this practice.

   **Note**: `SET TIMING ON` is a SQL*Plus command which makes SQL*Plus display the execution time for each statement it executes.

```
conn HR/ABcd##1234

set timing on

CREATE TABLE EMP NOLOGGING AS
SELECT A.* FROM EMPLOYEES A, EMPLOYEES B,EMPLOYEES C
UNION ALL
SELECT A.* FROM EMPLOYEES A, EMPLOYEES B,EMPLOYEES C
UNION ALL
SELECT A.* FROM EMPLOYEES A, EMPLOYEES B,EMPLOYEES C;

-- this statement gathers statistics for the created table
ANALYZE TABLE EMP COMPUTE STATISTICS;

CREATE INDEX EMPNO_NDX ON EMP(EMPLOYEE_ID) ;
ANALYZE INDEX EMPNO_NDX COMPUTE STATISTICS;

set timing off

-- display table size
SELECT BLOCKS*8/1024 MB FROM USER_TABLES WHERE TABLE_NAME='EMP';
```

5. Change the current user to `SYS`. From now on, in this practice, this session is called the **admin session**.

```
conn / as sysdba
```

# Examining the Undo Statistics

In the following steps, you will generate some undo data and examine its impact on the undo statistics. The target is to understand how the undo is generated and understand its statistics.

**6.** In the admin session, as `SYS`, display the undo statistics.

There is no client sessions on the database that change data. Therefore, only a few undo blocks are there.

```
@ display_undo_stats.sql
```

**7.** Run the following statement to manually create an AWR snapshot.

AWR snapshot is a snapshot of the database performance statistics. In real life scenarios, we do not need to manually create an AWR snapshots. AWR snapshots that are automatically created in the database are adequate.

```
exec DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT(FLUSH_LEVEL=>'ALL')
```

**8.** Display information about the created snapshot. Take a note of the `SNAP_ID`.

```
col BEGIN_INTERVAL_TIME for a26
col END_INTERVAL_TIME for a26

SELECT SNAP_ID, BEGIN_INTERVAL_TIME, END_INTERVAL_TIME
FROM DBA_HIST_SNAPSHOT
ORDER BY SNAP_ID DESC FETCH FIRST 1 ROWS ONLY;
```

**9.** Open **another** Putty session and login to it as `HR`. From now on, in this practice, this session is called the **client session**.

I recommend changing the font color of this session to a different color (like light green) so that you can distinguish it from the admin session.

```
sqlplus hr/ABcd##1234
```

**10.** In the **client** session, run the following code to produce some undo data. You do not have to wait for the statement to finish. Go to the next step.

```
UPDATE EMP SET SALARY=SALARY*1 WHERE EMPLOYEE_ID BETWEEN 100 AND 150;
```

**11.** In the **admin** session, display the undo statistics. You can run the script multiple times to see how the undo segments are increasing as the `UPDATE` statement is still being executed.

Observe that the number of undo blocks is significantly increased.

The database did not need to take blocks from the expired blocks.

**Note**: remember that the `V$UNDOSTAT` is refreshed every 10 minutes. It may take some time till you see changes in the view.

```
@ display_undo_stats.sql
```

**12.** In the client session, commit the command.

```
COMMTI;
```

**13.** In the **admin** session, display the undo statistics.

The number of unexpired undo blocks will significantly increase after a few minutes. In my testing case, it took me nearly 5 minutes before seeing the changes.

**Note**: In SQL*Plus, you can re-run the code in the buffer using the forward slash symbol '/'.

```
@ display_undo_stats.sql
```

You examined the undo blocks generated by user transactions. Let's examine how the long running queries are detected.

**14.** Open a new Putty session to `srv1` as `oracle`. Invoke SQL*Plus and login to the database as `HR`.

```
sqlplus hr/ABcd##1234
```

**15.** Submit the following update statement.

```
UPDATE EMP SET SALARY=SALARY*1 WHERE EMPLOYEE_ID=100;
```

**16.** In the **client** session, submit the following query. This query needs to access the active undo extent to retrieve its rows.

```
SELECT * FROM EMP WHERE EMPLOYEE_ID IN (100,101,102);
```

**17.** In the **admin** session, retrieve the undo statistics.

Observe that the `MAXQUERYLEN` is increased as a result.

```
@ display_undo_stats.sql
```

**18.** In the **admin** session, create an AWR snapshot.

```
exec DBMS_WORKLOAD_REPOSITORY.CREATE_SNAPSHOT(FLUSH_LEVEL=>'ALL')
```

**19.** Display information about the created snapshot. Take a note of the `SNAP_ID`.

```
col BEGIN_INTERVAL_TIME for a26
col END_INTERVAL_TIME for a26

SELECT SNAP_ID, BEGIN_INTERVAL_TIME, END_INTERVAL_TIME
FROM DBA_HIST_SNAPSHOT
ORDER BY SNAP_ID DESC FETCH FIRST 1 ROWS ONLY;
```

**20.** Exit from the third session (where we ran the `UPDATE` statement)

# Activating the Undo Advisor

In the following steps, you will activate the undo advisor and check out its recommendations.

**21.** In the **admin** session, as SYS, submit a job task to the Undo Advisor. When prompted, enter the AWR snapshot begin and end IDs noted earlier.

Regarding the code below, observe the following:

- We can send a message from the code to the SQL*Plus command prompt using the procedure DBMS_OUTPUT.PUT_LINE. However, to make this procedure work, we must execute the SQL*Plus command 'set serveroutput on' before running the SQL or PL/SQL code. In SQL Developer, we can achieve the same by running the code as script.

- We can include a substitution variable in the code using the ampersand symbol. It prompts the user to enter a value for it when executing the code. The entered value will substitute the variable in the code.

```
conn / as sysdba

set serveroutput on

DECLARE
 tid NUMBER;
 tname VARCHAR2(30);
 oid NUMBER;
BEGIN
 DBMS_ADVISOR.CREATE_TASK('Undo Advisor', tid, tname, 'Undo Advisor Task');
 DBMS_OUTPUT.PUT_LINE('Task name....:' || tname);
 DBMS_ADVISOR.CREATE_OBJECT(tname, 'UNDO_TBS', null, null, null, 'null', null,
oid);
 DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'TARGET_OBJECTS', oid);
 DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'START_SNAPSHOT', &beginid);
 DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'END_SNAPSHOT', &endid);
 DBMS_ADVISOR.SET_TASK_PARAMETER(tname, 'INSTANCE', 1);
 DBMS_ADVISOR.EXECUTE_TASK(tname);
END;
/
```

**22.** Open the SQL Developer, connect to the non-CDB database as SYSTEM, submit the following queries. Enter the task name returned by the code in the preceding step.

```
SELECT * FROM DBA_ADVISOR_FINDINGS WHERE TASK_NAME='&TNAME';

SELECT * FROM DBA_ADVISOR_RECOMMENDATIONS WHERE TASK_NAME='&TNAME';
```

# Calculating the Required Undo Size for Specific Undo Retention

In the following steps, you will calculate the minimum required undo tablespace size to serve an undo retention period of 1 hour.

**23.** In the admin session, retrieve the undo tablespaces in the database.

In our environment, we have a single undo tablespace named as `UNDOTBS1`.

```
SELECT TABLESPACE_NAME FROM DBA_TABLESPACES WHERE CONTENTS = 'UNDO';
```

**24.** Retrieve the size specifications of the undo datafiles. Take a note of the output.

```
SELECT FILE_ID,BYTES/1024/1024 MB, STATUS, AUTOEXTENSIBLE,
ROUND(MAXBYTES/1024/1024) MAX_MB
FROM DBA_DATA_FILES
WHERE TABLESPACE_NAME = 'UNDOTBS1';
```

**25.** Submit the following query to display the maximum undo block per second as recorded in the `V$UNDOSTAT`.

In real life, this query should be submitted after the system is being under normal workload for a few hours or days. You might consider running the same query in a few days during normal workload cycle.

```
SELECT MAX(UNDOBLKS/((END_TIME-BEGIN_TIME)*3600*24)) undo_bps
FROM V$UNDOSTAT;
```

**26.** Display the current Undo retention period configured in the system.

The parameter is set to 900 seconds, which is equivalent to 15 minutes. We might need to change this parameter to the required value after we make sure that the undo tablespace is large enough.

```
show parameter UNDO_RETENTION
```

**27.** Display the standard data block size in the database.

Its value is 8192 bytes, which is equivalent to 8 K.

```
show parameter DB_BLOCK_SIZE
```

**28.** Calculate the minimum required undo tablespace size for an undo retention period of 1 hour.

The undo size calculated below is based on the highest undo rate recorded in the system.

undo size = Required Retention period * undo block per second * block size

In my testing case, the calculation goes as follows:

undo size = 1 * 60 * 60 * 28 * 8 = 806400 K

(1 * 60 * 60: retention period in seconds; 28: undo block per second; 8: standard block size)

undo size = 806400/1024 = 787.5 MB

**29.** Compare between the required space and the undo tablespace datafile size noted earlier.

**30.** As a cleanup, exit from SQL Developer and Putty sessions.

**Note:**

Setting the udo tablespace to its optimal size based on its current statistics does not mean that the required udo retention is always guaranteed; simply because the workload might change in the future. To make sure that all the needed unexpired undo is reserved for serving the required udo retention, set the property `RETENTION GUARANTEE` on the tablespace. But remember in this case you risk failing DML transactions if they do find expired undo in the undo tablespace.

## Summary

- Statistics used for monitoring udo usage in a database can be obtained by querying the view `V$UNDOSTAT`
- Udo Advisor can be activated to determine the optimal size of the undo tablespace.
- To make sure that the undo tablespace has enough space to keep unexpired undo enough to achieve the `UNDO_RETENTION` value, we can use the following formula to determine the required undo size:

  undo size = Required Retention period * undo block per second * block size