Ejercicio 10 - tema 7 Administración de datos Undo

Vicente Romero Andrade

I. OBJETIVO

E L objetivo es comprender e ilustrar los efectos que produce un tablespace undo sin espacio durante la operación de la base de datos.

II. DESARROLLO

A. Generar una sentencia SQL que muestre el tablespace undo que actualmente está en uso

```
SELECT value FROM v$parameter WHERE name='
    undo_tablespace';
```

Código 1. sentencia consulta tablespace undo

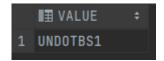


Figure 1. Salida punto A

B. Generar una sentencia SQL que genere un nuevo | tablespace undo cuya característica principal es contar | con un espacio muy limitado y por lo tanto la base | to de datos estará en riesgo de generar errores al re- | tancount as num_querir más espacio undo del disponible. Nombrar al ta- | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data file ubicado en | tancount as num_holespace undotbs2, asignarle un solo data f

Código 2. sentencia generar tablespace undo

C. Generar la sentencia SQL necesaria para que la instancia ahora haga uso del nuevo tablespace en lugar el actual. Aplicar el cambio únicamente mientras la instancia esté iniciada

```
alter system set undo_tablespace='UNDOTBS2' scope=
   memory;
```

Código 3. sentencia generar tablespace undo

D. Crear una nueva sesión con el usuario SYS, ejecutar nuevamente la sentencia del inciso A para verificar que la instancia ahora está haciendo uso del tablespace undotbs2

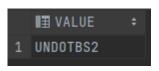


Figure 2. Salida punto D

E. Considerar la vista v\$undostat. Generar la siguiente consulta. Mostrar la fecha inicio y fin del periodo de la muestra hasta nivel de segundos, el id del tablespace que se está empleando, total de bloques undo empleados, total de transacciones, id de la consulta que tardó el mayor tiempo en ejecutarse, tiempo en segundos de la consulta que más tiempo tardó en ejecutarse durante el periodo de la muestra, número de bloques undo activos, no expirados y expirados; el valor del parámetro undo_retention calculado por la instancia. Ordenar los registros con base a la fecha de inicio de forma descendente. Mostrar únicamente los primeros 20 registros

```
SELECT BEGIN_TIME,

END_TIME,
UNDOTSN,
UNDOBLKS as TOTAL_BLOQUES_UNDO_USADOS,
TXNCOUNT as NUM_TRANSACCIONES,
MAXQUERYID,
MAXQUERYLEN,
ACTIVEBLKS,
UNEXPIREDBLKS,
EXPIREDBLKS,
TUNED_UNDORETENTION
FROM V$UNDOSTAT WHERE ROWNUM<=20 ORDER BY BEGIN_TIME
DESC;
```

Código 4. sentencia consulta tablespace stats



Figure 3. Salida punto E

- F. Considerar el periodo de muestreo más reciente de la consulta realizada en el inciso E
- 1) ¿Cuantos bloques podrían ser sobrescritos sin causar mayores inconvenientes?: Los que son EXPIRED pueden ser sobrescritos por lo tanto es 0 asi que ningun bloque se sobreescribira.
- 2) ¿Cuántos bloques NO pueden ser sobrescritos aun?: 160 bloques que corresponden a los activos

G. Observar los valores de la columna undotsn de la consulta 11 del inciso E. Confirmar que a partir del cambio del tablespace 12 undo, el valor de esta columna cambia. Generar una sentencia 14 SQL que permita mostrar los nombres de los tablespaces 15 asociados a los valores de esta columna. Tip: auxiliarse de 16 v\$tablespace. En la consulta incluir únicamente el periodo de 18 la muestra, id del tablespace y su nombre

```
SELECT BEGIN_TIME,

END_TIME,

UNDOTSN,

T.NAME

FROM V$UNDOSTAT U JOIN V$TABLESPACE T ON U.UNDOTSN=T OR USE TO SELECT TO SELE
```

Código 5. sentencia consulta tablespace id-nombre

```
## SEGIN_TIME : ## END_TIME : ## UNDOTSN : ## NAME : 1

2021-07-21 20:38:13 2021-07-21 20:38:10 8 UNDOTSS2 2

2021-07-21 20:28:13 2021-07-21 20:38:13 8 UNDOTSS2 3

2021-07-21 20:18:13 2021-07-21 20:18:13 8 UNDOTSS2 4

4 2021-07-21 20:18:13 2021-07-21 20:18:13 2 UNDOTSS2 5

5 2021-07-21 19:58:13 2021-07-21 20:08:13 2 UNDOTSS1 7

2021-07-21 19:48:13 2021-07-21 19:58:13 2 UNDOTSS1 7

2021-07-21 19:48:13 2021-07-21 19:48:13 2 UNDOTSS1 8

2021-07-21 19:28:13 2021-07-21 19:38:13 2 UNDOTSS1 8

2021-07-21 19:28:13 2021-07-21 19:38:13 2 UNDOTSS1 2

2021-07-21 19:18:13 2021-07-21 19:28:13 2 UNDOTSS1 10

2021-07-21 19:18:13 2021-07-21 19:28:13 2 UNDOTSS1 10

2021-07-21 19:18:13 2021-07-21 19:28:13 2 UNDOTSS1 10

2021-07-21 19:08:13 2021-07-21 19:18:13 2 UNDOTSS1 10
```

Figure 4. Salida punto G

H. Generar una consulta que muestre las siguientes colum-39 nas: Nombre del tablespace undo que fue creado en pasos 40 begin anteriores, total de bloques que contiene, total de bloques li-41 v_stm bres, porcentaje de bloques libres. Confirmar que el porcentaje de bloques libres es bajo, menor al 20%. Esta condición pone 43 en riesgo la operación correcta que requieren hacer uso de 44 los datos Undo

```
1 SELECT T.TABLESPACE_NAME,
2 DF.BLOCKS TOTAL_BLOQUES,
3 (DF.BLOCKS-SUM(E.BLOCKS)) BLOQUES_LIBRES,
4 TRUNC(((DF.BLOCKS-SUM(E.BLOCKS))/DF.BLOCKS *100)
,2) PORCENTAJE_LIBRE FROM DBA_TABLESPACES T

JOIN DBA_DATA_FILES DF

ON T.TABLESPACE_NAME=DF.TABLESPACE_NAME

JOIN DBA_UNDO_EXTENTS E

ON T.TABLESPACE_NAME=E.TABLESPACE_NAME

WHERE T.TABLESPACE_NAME='UNDOTBS2'

GROUP BY T.TABLESPACE_NAME, DF.BYTES, DF.BLOCKS;
```

Código 6. sentencia consulta tablespace bloques libres

```
IN TABLESPACE_MANE : IN TOTAL_BLOQUES : IN BLOQUES_LIBRES : IN PORCENTAJ_LIBRE : 1 UNDOTES2 3840 3080 80.2
```

Figure 5. Salida punto H

I. Empleando el usuario creado en temas anteriores, generar una tabla con la siguiente estructura

```
DECLARE

v_count number;

v_count_s number;

v_username varchar2(30) := 'VRA_TBS_MULTIPLE';

v_table varchar2(30) := 'VRA_CADENA_2';

v_secuencia varchar2(30) := 'SEC_VRA_CADENA_2';

BEGIN

--Verificar si la table existe

select count(*) into v_count

from all_tables
```

```
where table_name = v_table
    and owner = v username;
    --Verificar si la secuencia existe
    select count(*) into v_count_s
      from all_sequences
    where sequence_name = v_secuencia
      and sequence_owner = v_username;
    --Si existe la tabla, entonces se borra
    if v_count > 0 then
20
      execute immediate 'drop table '||v_username||'.'
      ||v_table;
22
    end if;
    if v_count_s > 0 then
      execute immediate 'drop sequence '||v_username||
24
       '.'||v_secuencia;
    end if;
    execute immediate 'create table '||v_username||'.'
26
       ||v_table||' (
      id number constraint vra_cadena_2_pk primary key
      cadena varchar2(1024)
    ) nologging';
29
30
    execute immediate 'create sequence '||v_username||
       '.'||v_secuencia;
31 end:
32 /
33 -- Inserts
34 declare
35
    v_rows number;
    v_stmt varchar2(200);
    v_username varchar2(30) := 'VRA_TBS_MULTIPLE';
    v_table varchar2(30) := 'VRA_CADENA_2';
    v_secuencia varchar2(30) :='SEC_VRA_CADENA_2';
41 v_rows := 50000;
  v_stmt := 'insert into '||v_username||'.'||v_table||
    id, cadena
    ) values (:1, :2)';
45 for v_index in 1 .. v_rows loop
    execute immediate v_stmt using VRA_TBS_MULTIPLE.
      SEC_VRA_CADENA_2.nextval, dbms_random.string('P'
       ,1024);
47 end loop;
48 end:
50 commit:
```

Código 7. sentencia crear tabla

J. Replicación del error al ejecutar sentencias DML



Figure 6. Salida consulta punto E

Registros eliminados	# bloques utilizados	# transacciones ejecutadas	#bloques activos	Retención en segundos calculado
1 - 5000	835	138	160	9835
5001 - 10000	1669	144	160	1896
10001 - 15000	835	12	160	1741
Table I				

TABLA REGISTROS ELIMINADOS

1) Revisar los datos de la tabla y generar una conclusión al respecto: En este caso el unico cambio significativo es el tiempo de retención el cual fue bajando

K. ¿Qué acciones se pueden realizar para corregir el error adicional a aumentar el tamaño del tablespace undo?

A parte de aumentar el tamaño de datafile que contiene el tablespace undo otra acción seria configurar la cláusula retention noguarantee sacrificando datos los datos unexpired.

- L. Replicación del error snapshoot too old
- 1) ¿Cuántas instrucciones delete se tuvieron que ejecutar para provocar el error snapshppt too old?: 4 instrucciones

Figure 7. Salida error snapshoot to old

M. Provocar ahora el comportamiento inverso: Hacer que la base de datos le de preferencia a consultas que requieran hacer uso de datos undo en lugar de dar preferencia a las sentencias SOL

```
1 -- Se configura el retention guarantee
2 alter tablespace undotbs2 retention guarantee;
```

Código 8. Configuración priorizar datos undo

```
declare
   v_rows number;
    v_stmt varchar2(200);
   v_username varchar2(30) := 'VRA_TBS_MULTIPLE';
   v_table varchar2(30) := 'VRA_CADENA_2';
 begin
6
    v_rows := 10;
    v_stmt := 'delete from '||v_username||'.'||v_table
      11'
      where id in (SELECT id from '||v_username||'.'||
      v_table||' WHERE ROWNUM<=5000)';
   execute immediate v_stmt;
11 end;
12 /
13 commit;
```

Código 9. script que se ejecuta constantemente para eliminar los datos y llenar undo

Figure 8. Salida error comportamiento inverso

III. CONCLUSIONES

En este ejercicio se logro comprender mejor la administración de datos undo asi como lo que implican, se puede elegir entre configurar para que estos tengan prioridad o para que las sentencias DML siempre puedan ser ejecutas sin problema.