Workshop Multitenant, Multimodel, In-Memory para la base de Datos Oracle

Parte 1 de 3

Contenidos

Requerimientos iniciales

- Clave SSH privada para acceder a la maquina cloud que contiene las bases de datos. Esta clave se proporciona junto con la documentación necesaria para el workshop.
- Cliente SSH para poder acceder al host de base de datos en el que se ejecutara el workshop
- IP de la maquina

Multitenant

Familiarización con el entorno

El laboratorio se desarrolla en una máquina virtual alojada en el cloud de Oracle, a la que accederemos empleando el usuario "oracle" y protocolo ssh. En la conexión ssh no especificaremos ninguna contraseña, sino que se empleará el fichero de clave privada SSH. Ejemplo:

```
ssh -i id_rsa.pub oracle@130.91.179.135
```

Veremos que en el $HOME del usuario Oracle hay dos Shell script que nos servirán para cargar las variables de entorno de las bases de datos. A modo de ejemplo, podemos cargar las variables de la contenedora CDBA de la siguiente forma:

```
[oracle@single19c ~]$ . CDBA.env
```

Para éste laboratorio emplearemos dos bases de datos contenedoras CDBA y CDBB. El "db_name" de las bases de datos será igual para todas las máquinas de todos los asistentes a éste laboratorio. Sin embargo el "db_unique_name" será exclusivo para cada asistente/máquina, por lo que vamos a apuntar a continuación el "db_unique_name" de ambas, apoyándonos en los siguientes comandos:

```
[oracle@single19c ~]$ srvctl config database | grep -i cdba
CDBA_fra2rx
[oracle@single19c ~]$ srvctl config database | grep -iv cdba
CDBB
```

Es decir, en éste caso concreto el db_unique_name para la CDBA es "CDBA_fra2rx", y el db_unique_name para la CDBB es "CDBB". Tomaremos nota de nuestros propios nombres, y podremos realizar un "buscar y reemplazar" de los siguientes comandos que figuren en éste laboratorio cambiando los valores antiguos por los valores concretos de ésta máquina, que acabamos de obtener.

Revisión de configuración de Transparent Data Encryption - TDE

El contenedor CDBA está configurado con Transparent Data Encryption “TDE”, por lo que tiene creado un wallet donde guarda las claves de cifrado de los contenedores.

El laboratorio de hoy no está focalizado en TDE, y no realizaremos cifrado de columnas o de tablespaces, si bien es interesante entender el estado de esta configuración tanto en la propia contenedora, como en la PDB preexistente:

```
[oracle@single19c ~]$ . CDBA.env
[oracle@single19c ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 5 09:22:18 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0

SQL> show con_name

CON_NAME
------------------------------
CDB$ROOT → Aterrizamos en la contenedora

SQL> show pdbs → la CDB$ROOT aloja la PDB$SEED y las PDB de usuario

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           READ WRITE NO
         4 JSON                           READ WRITE NO
         5 SOE                            READ WRITE NO




SQL> set linesize 999
SQL> column WRL_PARAMETER format a54
SQL> column wallet_type format a9
SQL> column con_id format 9
SQL> column status format a10
```

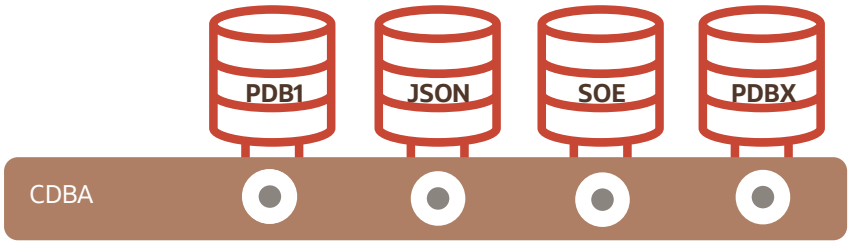
```
SQL> select WRL_PARAMETER, STATUS, WALLET_TYPE, CON_ID from
v$encryption_wallet;


WRL_PARAMETER                                      STATUS   WALLET_TY CON_ID
-------------------------------------------------- ---------- --------- ---
---
/opt/oracle/dcs/commonstore/wallets/tde/CDBA_fra2rx/   OPEN   AUTOLOGIN   1
                                                   OPEN     AUTOLOGIN   2
                                                   OPEN     AUTOLOGIN   3

SQL> exit
```

Es decir, en ésta contenedora tenemos un wallet de tipo autologin, que está abierto tanto en la CDB$ROOT (con_id valor 1) como en el resto de las PDBs. Al ser un wallet de autologin, éste se abrirá automáticamente al arrancar la CDB y las PDBs, por lo que no necesitaremos abrir el wallet expresamente entre reinicios de la contenedora o entre los reinicios de las PDBs.

En este workshop estaremos manejando PDBs sin cifrado, por lo que más allá de la configuración de autologin, no necesitaremos verbos adicionales de cifrado en las operaciones con PDBs.

Creación de PDBs con diferentes configuraciones

La contenedora CDBA contiene ya varias PDBs creadas. En ésta práctica utilizaremos el mismo contenedor para ver ejemplos sencillos de borrado y creación de PDBs. El escenario al finalizar esta práctica será:



En primer lugar vamos a borrar la PDB preexistente PDB1 y a crearla nuevamente. Para borrar una PDB es necesario cerrarla primero (de estado OPEN a estado MOUNT, ya que no existe estado CLOSED en PDBs) :

```
[oracle@single19c ~]$ . CDBA.env
[oracle@single19c ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 5 13:36:14 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0

SQL> alter pluggable database pdb1 close;

Pluggable database altered.

SQL> drop pluggable database PDB1 including datafiles;

Pluggable database dropped.
```

Ahora que la hemos borrado de forma completa, procedemos a recrearla. El comando básico de creación de una PDB nos pedirá un nombre para crear un administración particular para dicha PDB (que podremos usar si se requiriera emplear un administrador diferente por cada PDB) :

```
SQL> create pluggable database PDB1 admin user miadmin identified by
We1c0m3_We1c0m3_;

Pluggable database created.

SQL> alter pluggable database PDB1 open;

Pluggable database altered.

SQL> show pdbs

    CON_ID CON_NAME                  OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         4 JSON                           READ WRITE NO
         5 SOE                            READ WRITE NO
         6 PDB1                           READ WRITE NO
```

Verificamos que los datafiles de la nueva PDB se encuentran en el diskgroup "+DATA", que es el destino por defecto, especificado en el parámetro db_create_file_dest de la CDB (atención, debemos emplear el con_id concreto que tenga PDB1 en nuestro laboratorio, según la salida del comando "show pdbs"):

```
SQL> show parameters db_create_file_dest

NAME                                 TYPE     VALUE
------------------------------------ ----------- ------------------------------
db_create_file_dest                  string   +DATA

SQL> select file_name from cdb_data_files where con_id=6;

FILE_NAME
--------------------------------------------------------------------------------
------------------------------
+DATA/CDBA_FRA2RX/D4D56AD3EA656459E053C900000A7D55/DATAFILE/system.301.10931761
95
+DATA/CDBA_FRA2RX/D4D56AD3EA656459E053C900000A7D55/DATAFILE/sysaux.304.10931761
95
+DATA/CDBA_FRA2RX/D4D56AD3EA656459E053C900000A7D55/DATAFILE/undotbs1.303.109317
6195
```

Existen otras opciones interesantes a la hora de crear una PDB vacía, como puede ser la capacidad de lanzar un mantenimiento de la PDB con paralelismo, o la capacidad de limitar la cantidad de espacio total que la suma de ficheros de una PDB puede llegar a tomar (ambas configuraciones son opcionales).

También podemos elegir un diskgroup diferente al que utiliza la propia contenedora. En este caso crearemos otra PDB en el diskgroup "RECO", ya que es el único DG alternativo en

este entorno, y porque además lo hemos creado con redundancia de tipo “FLEX” (característica relevante para las prácticas de mañana). Por lo tanto, seguiremos usando OMF, pero almacenando los ficheros concretos de ésta PDB en otro diskgroup:

Vamos a crear la PDBX para mostrar éstas capacidades :

```
SQL> create pluggable database PDBX admin user miadmin identified by
We1c0m3_We1c0m3_ parallel 2 storage(maxsize 120G) create_file_dest='+RECO';

Pluggable database created.

SQL> alter pluggable database PDBX open;

Pluggable database altered.

SQL> show pdbs

SQL> show pdbs

    CON_ID CON_NAME                  OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                  READ ONLY  NO
         4 JSON                      READ WRITE NO
         5 SOE                       READ WRITE NO
         6 PDB1                      READ WRITE NO
         7 PDBX                      READ WRITE NO
```

Vamos a verificar que los datafiles se han creado en un diskgroup diferente. Para ello hay que tomar previamente nota del identificador de la PDB (en éste caso es 7) para obtener los paths de la vista cdb_data_files:

```
SQL> set linesize 999
SQL> select file_name from cdb_data_files where con_id=7;

FILE_NAME
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
----------------------------------------
+RECO/CDBA_FRA2RX/D4D4B856213BC036E053C900000A116C/DATAFILE/system.263.10931730
09
+RECO/CDBA_FRA2RX/D4D4B856213BC036E053C900000A116C/DATAFILE/sysaux.264.10931730
11
+RECO/CDBA_FRA2RX/D4D4B856213BC036E053C900000A116C/DATAFILE/undotbs1.262.109317
3009
```

Procedimiento de Unplug/Plug manual de una PDB entre CDBs

UNPlug de PDB1

Para hacer una operación de unplug (una desconexión) de una PDB respecto de su CDB, es necesario ejecutar tres comandos: A) cerrar la PDB, B) Desconectar la PDB y C) borrar la PDB del diccionario de su antigua CDB. Éste último paso "C" suele olvidarse, y se podría dar – erróneamente- por sentado que la PDB se elimina automáticamente del diccionario al realizar el unplug. Esto no es así, ya que es posible dejar una PDB desconectada de su CDB, pero aún formando parte de ella, permitiendo que procesos como el backup de RMAN tenga en cuenta todos los ficheros (incluídas las BDD desconectadas).

Procedemos a realizar el unplug siguiendo los 3 pasos anteriormente descritos:

```
[oracle@single19c ~]$ . CDBA.env
[oracle@single19c ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 5 13:44:26 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0

SQL> alter pluggable database PDB1 close;

Pluggable database altered.

SQL> alter pluggable database pdb1 UNPLUG INTO '/tmp/pdb1.xml';

Pluggable database altered.
```

Antes de finalizar el proceso de unplug revisamos el fichero de manifiesto creado con el unplug:

```
SQL> !cat /tmp/pdb1.xml
```

Y terminamos eliminando la PDB del diccionario de datos de su antigua CDB:

```
 SQL>  drop pluggable database PDB1 KEEP DATAFILES;

Pluggable database dropped.

SQL> exit
```

Es importante señalar que “keep datafiles” es la opción por defecto y no sería necesario declararlo expresamente. Sin embargo lo hemos añadido para dejar patente el objetivo de la operación; eliminarla del diccionario de datos pero sin tocar sus datafiles. Necesitaremos los datafiles y el fichero de manifiesto para hacer plug de nuevo en una CDB.

Plug de PDB1 en el contenedor CDBB

Antes de proceder a conectar “PDB1” a la nueva contenedora, es recomendable verificar que son plenamente compatibles. Para ello ejecutamos el siguiente código en la contenedora destino “CDBB”:

```
[oracle@single19c ~]$ . CDBB.env
[oracle@single19c ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 5 14:09:07 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0


SQL> SET SERVEROUTPUT ON
DECLARE
 compatible CONSTANT VARCHAR2(3) := CASE
DBMS_PDB.CHECK_PLUG_COMPATIBILITY(pdb_descr_file => '/tmp/pdb1.xml',pdb_name =>
'PDB2')
 WHEN TRUE THEN 'YES'
 ELSE 'NO'
END;
BEGIN
 DBMS_OUTPUT.PUT_LINE(compatible);
END;
/

YES
```

```
PL/SQL procedure successfully completed.
```

Dado que la salida de ésta función es muy poco descriptiva, es recomendable revisar los posibles avisos o errores en la vista PDB_PLUG_IN_VIOLATIONS. En éste caso veremos que existen múltiples mensajes de tipo “warning” y ninguno de tipo “error”. Los mensajes de tipo “warning” no impedirán que la operación de plug funcione con éxito, pero nos indican casi siempre sobre divergencias entre la configuración de una y otra contenedora :

```
SQL> set linesize 999
SQL> select type, message from PDB_PLUG_IN_VIOLATIONS where name='PDB2';

TYPE
---------
MESSAGE
-------------------------------------------------------------
WARNING
Tablespace SYSTEM is not encrypted. Oracle Cloud mandates all tablespaces
should be encrypted.
```

Obviaremos los mensajes de warning (en éste laboratorio) y proseguiremos. Teniendo el fichero de manifiesto y los datafiles de “PDB1”, procedemos a conectar ésta PDB al segundo contenedor CDBB. Éste contenedor se encuentra en la misma máquina, por lo que no es necesario que copiemos el fichero xml ni los datafiles a una máquina diferente; decimos que no hay movimiento de datafiles.

```
SQL> show pdbs

    CON_ID CON_NAME          OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                 READ ONLY  NO

SQL> create pluggable database pdb1 using '/tmp/pdb1.xml';

Pluggable database created.

SQL> alter pluggable database pdb1 open;

Pluggable database altered.

SQL> !rm /tmp/pdb1.xml

SQL> show pdbs

    CON_ID CON_NAME                OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                 READ ONLY  NO
         3 PDB1                     READ WRITE NO
```

Como hemos visto, el fichero de manifiesto ".xml" es únicamente necesario para realizar la operación de unplug/plug. Una vez que hemos completado la operación, éste fichero ya no se emplea y podemos decidir eliminarlo o guardarlo para nuestra referencia.

Clonado/duplicado Online en la misma contenedora

Gracias a que cada PDB dispone de su propio tablespace de UNDO (versiones 12.2 en adelante), es posible realizar un duplicado de una PDB sin pérdida de servicio; no es necesario detener la PDB origen ni ponerla en read-only. Por lo tanto el duplicado de una PDB es un proceso puramente online, y de extrema sencillez (1 comando). Procedemos a duplicar la PDB1, que ahora se encuentra en la CDBB, creando una nueva PDB "PDB2" :

```
[oracle@single19c ~]$ . CDBB.env
[oracle@single19c ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 5 14:29:56 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0

SQL> create pluggable database pdb2 from pdb1 parallel 4;

Pluggable database created.

SQL> alter pluggable database pdb2 open;

Pluggable database altered.

SQL> show pdbs

    CON_ID CON_NAME                  OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           READ WRITE NO
         4 PDB2                           READ WRITE NO
```

En éste caso hemos añadido una cláusula adicional "parallel" para acelerar la copia.

Clonado/duplicado Online hacia contenedora remota

En ésta práctica vamos a clonar/duplicar de forma online la PDB “PDBX” que se encuentra en la CDBA en la CDBB.

Para clonar/duplicar una PDB desde una CDB a otra CDB (sea en la misma máquina o en una máquina diferente), es necesario un dblink que conecte ambas CDB$ROOT. Éste dblink no se utiliza para copiar la información, sino únicamente para identificar el destino, y para ganar acceso al mismo.

El comando de clonado utiliza la misma sintáxis que el comando de creación de una PDB (es un “create pluggable database”). Además, el comando de clonado será quien utilice el dblink. Esto quiere decir que ejecutaremos el comando de clonado (con formato “create pluggable database…”) en la CDB destino; en éste caso la CDBB.

Por ésto, debemos crear el DBLINK en CDBB, apuntando hacia CDBA. Por ello comenzaremos editando el tnsnames.ora para crear un alias de conexión contra CDBA . El primer paso consiste en obtener el nombre cualificado del servicio por defecto de CDBA (en éste ejemplo usaremos el servicio por defecto por no alargar el ejercicio, si bien es recomendable emplear un servicio dinámico creado por el usuario):

```
oracle@single19c ~]$ . CDBA.env
[oracle@single19c ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 5 15:15:40 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0


SQL> show parameters service_names

NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
service_names                        string      CDBA_fra2rx.tfexsubdbsys.tfexv
                                                 cndbsys.oraclevcn.com
```

→ el siguiente comando nos permitirá crear los nombres de dblink que deseemos, ya que de lo contrario el dblink debería llamarse de forma igual al nombre de la base de datos a la que se conecta:

```
SQL> alter system set global_names=FALSE scope=both;

System altered

SQL> exit


[oracle@single19c]$ cd $ORACLE_HOME/network/admin

[oracle@single19c admin]$ vi tnsnames.ora


→ introducimos el siguiente texto al final del fichero, reemplazando el nombre del servicio por el que acabamos de obtener y guardamos el cambio
CDBA =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = single19c)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = CDBA_fra2rx.sub12210903070.frivasvcn.oraclevcn.com)
    )
  )


[oracle@single19c admin]$ tnsping CDBA

TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on 05-JAN-2022
14:53:08

Copyright (c) 1997, 2021, Oracle.  All rights reserved.

Used parameter files:
/u01/app/oracle/product/19.0.0.0/dbhome_1/network/admin/sqlnet.ora


Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)(HOST =
single19c)(PORT = 1521)) (CONNECT_DATA = (SERVER = DEDICATED) (SERVICE_NAME =
CDBA)))
OK (10 msec)
```

Además, necesitaremos crear un usuario en CDBA, que será el usuario que emplee el DBLINK para conectarse a la misma CDBA. Asignaremos suficientes grants a éste usuario para poder realizar el clonado :

```
[oracle@single19c ~]$ cd $HOME
[oracle@single19c ~]$ . CDBA.env
[oracle@single19c ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 5 14:58:38 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0

SQL> CREATE USER c##clonados IDENTIFIED BY We1c0m3_We1c0m3_ CONTAINER=ALL;

User created.

SQL> GRANT CREATE SESSION, CREATE PLUGGABLE DATABASE, sysdba, sysoper TO
c##clonados CONTAINER=ALL;

Grant succeeded.

SQL> grant CREATE PLUGGABLE DATABASE, sysdba, sysoper to system container=all;

Grant succeeded.

SQL> exit
Disconnected from Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 -
Production
Version 19.12.0.0.0
```

En este momento, ya tenemos todos los elementos necesarios para crear y probar el dblink desde CDBB hacia CDBA:

```
[oracle@single19c ~]$ . CDBB.env
[oracle@single19c ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 5 15:22:02 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0

→ el siguiente comando nos permitirá crear los nombres de dblink que deseemos,
ya que de lo contrario el dblink debería llamarse de forma igual al nombre de
la base de datos a la que se conecta:

SQL> alter system set global_names=FALSE scope=both;
```

```
SQL>  CREATE DATABASE LINK linkclonado CONNECT TO c##clonados IDENTIFIED BY
We1c0m3_We1c0m3_ USING 'CDBA';

Database link created.

SQL> select sysdate from dual@linkclonado;

SYSDATE
---------
05-JAN-22
```

Disponiendo ya del DBLINK preceptivo para el clonado de PDBs, ejecutamos el sencillo comando de clonado remoto:

```
SQL> show parameters db_name

NAME                                 TYPE     VALUE
------------------------------------ -------- ------------------------------
db_name                              string   CDBB

SQL> show pdbs

    CON_ID CON_NAME                  OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           READ WRITE NO
         4 PDB2                           READ WRITE NO

SQL> create pluggable database PDBX from PDBX@linkclonado;

Pluggable database created.

SQL> alter pluggable database PDBX open;

Pluggable database altered.

SQL> exit
Disconnected from Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 -
Production
Version 19.12.0.0.0
```

Recuperación en el tiempo “PITR” de una PDB

En ésta última práctica vamos a comprobar cómo recuperar una PDB concreta dentro de una CDB, sin afectar a la CDB o a otras PDB de la misma CDB. Además, ésta recuperación será de tipo “Point in time recovery”. Es decir, vamos a recuperar una PDB a un momento en el pasado.

La primera tarea necesaria es generar un backup de RMAN. En éste caso vamos a crear un backup de toda la CDB; esto implica hacer backup tanto de la CDB$ROOT como de todas las PDBs que contenga.

Haremos la práctica con la “PDB1” en el contenedor “CDBB”:

```
[oracle@single19c ~]$ . CDBB.env
[oracle@single19c ~]$ rman target /

Recovery Manager: Release 19.0.0.0.0 - Production on Wed Jan 5 15:58:14 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

connected to target database: CDB19B (DBID=622880706)

RMAN> run{
delete force noprompt backup;
delete force noprompt archivelog all;
allocate channel t01 type disk;
allocate channel t02 type disk;
allocate channel t03 type disk;
backup database plus archivelog;
release channel t01;
release channel t02;
release channel t03;
}→ omitimos la salida de éste comando para facilitar la lectura

RMAN> exit
```

Para tener una referencia fácil a la hora de realizar la recuperación en el tiempo, crearemos una tabla sencilla de control donde insertaremos diferentes valores a lo largo del tiempo:

```
[oracle@single19c ~]$  . CDBB.env
[oracle@single19c ~]$  sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 5 16:03:30 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.
```

```
Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0

SQL> alter session set container=PDB1;

Session altered.

SQL> create table system.prueba (i number) tablespace sysaux;

Table created.

SQL> insert into system.prueba values (77);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from system.prueba order by i;

         I
----------
        77

SQL> select current_scn from v$database;

CURRENT_SCN
-----------
    4086690  → tomaremos nota de éste scn, ya que es al que restauraremos la
PDB posteriormente. A continuación, seguimos insertando datos en la tabla.

SQL>  insert into system.prueba values (99);

1 row created.

SQL> commit;

Commit complete.

SQL> select * from system.prueba;

         I
----------
        77
        99

SQL> select current_scn from v$database;

CURRENT_SCN
-----------
```

```
    4087364 → simplemente confirmamos que el SCN ha crecido, y que la tabla
tiene más registros en la actualidad.

SQL> exit
```

Procedemos a hacer un backup de los archivelogs generados, donde se guardarán éstos cambios realizados desde el backup de la base de datos:

```
[oracle@single19c ~]$ rman target /

Recovery Manager: Release 19.0.0.0.0 - Production on Wed Jan 5 16:09:17 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

connected to target database: CDBB (DBID=622880706)

RMAN> run{
allocate channel t01 type disk;
backup archivelog all;
release channel t01;
} → Omitimos la salida de éste comando para facilitar la lectura
```

Ahora procederemos a restaurar ésta PDB, exclusivamente, a un momento en el tiempo pasado donde la tabla de control únicamente tenía un registro, con un campo de valor “77”. Nos mantenemos dentro de RMAN :

```
RMAN> alter pluggable database pdb1 close immediate;

Statement processed

RMAN> run {
set until scn 4086690;
restore pluggable database PDB1;
recover pluggable database PDB1 ;
}

executing command: SET until clause

Starting restore at 05-JAN-22
allocated channel: ORA_DISK_1
channel ORA_DISK_1: SID=340 device type=DISK

channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00009 to
+DATA/CDB19B/D4D6D6F9E20A45EEE053C900000ACCC7/DATAFILE/sysaux.308.1093183767
```

```
channel ORA_DISK_1: reading from backup piece
+RECO/CDB19B/D4D6D6F9E20A45EEE053C900000ACCC7/BACKUPSET/2022_01_05/nnndf0_tag20
220105t155916_0.270.1093190395
channel ORA_DISK_1: piece
handle=+RECO/CDB19B/D4D6D6F9E20A45EEE053C900000ACCC7/BACKUPSET/2022_01_05/nnndf
0_tag20220105t155916_0.270.1093190395 tag=TAG20220105T155916
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00008 to
+DATA/CDB19B/D4D6D6F9E20A45EEE053C900000ACCC7/DATAFILE/system.298.1093183767
channel ORA_DISK_1: reading from backup piece
+RECO/CDB19B/D4D6D6F9E20A45EEE053C900000ACCC7/BACKUPSET/2022_01_05/nnndf0_tag20
220105t155916_0.273.1093190411
channel ORA_DISK_1: piece
handle=+RECO/CDB19B/D4D6D6F9E20A45EEE053C900000ACCC7/BACKUPSET/2022_01_05/nnndf
0_tag20220105t155916_0.273.1093190411 tag=TAG20220105T155916
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:03
channel ORA_DISK_1: starting datafile backup set restore
channel ORA_DISK_1: specifying datafile(s) to restore from backup set
channel ORA_DISK_1: restoring datafile 00010 to
+DATA/CDB19B/D4D6D6F9E20A45EEE053C900000ACCC7/DATAFILE/undotbs1.273.1093183767
channel ORA_DISK_1: reading from backup piece
+RECO/CDB19B/D4D6D6F9E20A45EEE053C900000ACCC7/BACKUPSET/2022_01_05/nnndf0_tag20
220105t155916_0.277.1093190421
channel ORA_DISK_1: piece
handle=+RECO/CDB19B/D4D6D6F9E20A45EEE053C900000ACCC7/BACKUPSET/2022_01_05/nnndf
0_tag20220105t155916_0.277.1093190421 tag=TAG20220105T155916
channel ORA_DISK_1: restored backup piece 1
channel ORA_DISK_1: restore complete, elapsed time: 00:00:01
Finished restore at 05-JAN-22

Starting recover at 05-JAN-22
using channel ORA_DISK_1


starting media recovery
media recovery complete, elapsed time: 00:00:01

Finished recover at 05-JAN-22

RMAN> exit
```

Cualquier recuperación/restauración PITR es considerada una recuperación incompleta, ya que se descartan los datos posteriores a ese momento en el tiempo. Es por ésta razón por la que tendremos que abrir la PDB con “reset logs”; abriendo una nueva encarnación para ella.

Procedemos a abrir PDB1 y verificar que se ha restaurado/recuperado correctamente a ese momento en el tiempo:

```
[oracle@single19c ~]$ . CDBB.env
[oracle@single19c ~]$  sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 5 16:14:48 2022
Version 19.12.0.0.0

Copyright (c) 1982, 2021, Oracle.  All rights reserved.


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.12.0.0.0

SQL> alter pluggable database PDB1 open resetlogs;

Pluggable database altered.

SQL> alter session set container=PDB1;

Session altered.

SQL> select * from system.prueba order by i;

         I
----------
        77

SQL> exit
Disconnected from Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 -
Production
Version 19.12.0.0.0
```