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

Parte 2 de 3

Contenidos

Multitenant

Online Relocate entre contenedoras

Explicación

La operación de relocate permite mover una PDB desde una CDB hacia otra CDB que puede estar en la misma o en distinta máquina. Gracias a que cada PDB tiene su propio tablespace de UNDO, parte de ésta operación se puede realizar de forma online. Concretamente, la operación de "relocate" consta de dos fases :

A) Copiado inicial de la PDB hacia la CDB destino: Se inicia con el comando "create pluggable database… " ejecutado en la CDB destino. La PDB permanecerá abierta en todo momento en modo read/write en la CDB origen durante toda ésta fase. Por otro lado, la PDB copiada se queda en estado "relocating" en el destino.

B) Finalización/completado del relocate: Se inicia al ejecutar el comando "alter pluggable database x open;" en el destino. Desde ese momento, las DDLs/DMLs se congelarán en la PDB origen, se moverán los archivelogs pendientes desde la copia inicial (paso A) hacia la PDB destino, se aplicarán todos los cambios pendientes, se moverán las sesiones a la nueva PDB, y se cerrará la PDB en origen.

El único elemento necesario para ejecutar el relocate online de una PDB es la creación de un DBLINK desde el destino hacia el origen. Éste dblink se empleará únicamente para identificar la ubicación del origen, ganar acceso a él, y para enviar los archivelogs. No se emplea para realizar la copia de los datafiles; para esto se abren de forma transparentes sockets independientes.

Práctica de relocate

En ésta práctica vamos a hacer un relocate de la PDB1 desde “CDBB” hacia “CDBA”. Como en la práctica de ayer, necesitaremos crear un dblink (y por lo tanto una entrada en el tnsnames.ora y un usuario con suficientes grants en el destino) para realizar la operación. Dado que el comando de “create pluggable…” se ejecuta en el destino, y en éste caso el destino es CDBA, entonces el dblink se crea desde CDBA hacia CDBB.

Comenzaremos editando el tnsnames.ora para crear un alias de conexión contra CDBB . El primer paso consiste en obtener el nombre cualificado del servicio por defecto de CDBB (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 ~]$ . CDBB.env
[oracle@single19c ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 10:57:22 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      CDBB

→ 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 admin]$ cd $ORACLE_HOME/network/admin
```

```
[oracle@single19c admin]$ vi tnsnames.ora


→ introducimos el siguiente texto al final del fichero, adaptado con el nombre
del servicio que acabamos de obtener, y guardamos el cambio. Lo introducimos al
final del fichero.

CDBB =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = single19c)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = CDBB)
    )
  )


[oracle@single19c admin]$ tnsping CDBB

TNS Ping Utility for Linux: Version 19.0.0.0.0 - Production on 07-JAN-2022
11:00:53

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 =
CDB19B)))
OK (0 msec)
```

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

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 11:02:17 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 éste momento, ya tenemos todos los elementos necesarios para crear y probar el dblink desde CDBA hacia CDBB:

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 11:05:11 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 DATABASE LINK linkclonado CONNECT TO c##clonados IDENTIFIED BY
We1c0m3_We1c0m3_ USING 'CDBB';

Database link created.

SQL> select sysdate from dual@linkclonado;

SYSDATE
---------
07-JAN-22

SQL> exit
```

Procedemos a continuación a ejecutar el primer paso de relocate, que realiza la copia inicial de la PDB y la deja preparada para el segundo paso. Éste proceso tardará unos minutos, ya que realizará la copia inicial completa de la PDB:

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 11:26:33 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 PDB1 from PDB1@linkclonado RELOCATE availability
normal ;

Pluggable database created.

SQL> show pdbs

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

SQL> select pdb_name,status from cdb_pdbs where pdb_name='PDB1';

PDB_NAME
--------------------------------------------------------------------------------
-
STATUS
----------
PDB1
RELOCATING


SQL> exit
```

Ahora que la copia inicial está hecha, podemos decidir en qué momento del día (o qué otro día) queremos ejecutar la finalización del relocate. Será en ese momento donde se congelarán los cambios en el origen, se aplicarán en el destino, y se moverán las sesiones. Mientras no se ejecute el segundo paso, la base de datos en origen seguirá plenamente funcional, y se podrán seguir realizando cambios. Para demostrar esto, procederemos a crear una tabla en la ubicación original ahora que aún no hemos terminado el relocate, con el objetivo de verificar que esa tabla existe cuando terminemos el relocate. Es por lo tanto una sencilla tabla de prueba para verificar que la segunda etapa del relocate garantiza que todos los

cambios realizados desde la copia inicial se envían a la nueva ubicación antes de terminar el proceso.

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 11:37:07 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> alter session set container=PDB1;

Session altered.

SQL> create table system.prueba2 (i number) tablespace sysaux;

Table created.

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

Procedemos a terminar el relocate abriendo “PDB1” en CDBA. Con éste comando se aplicarán todos los cambios pendientes, se abrirá PDB1 en CDBA, y se eliminará de CDBB. También verificaremos que la tabla de prueba existe en la nueva ubicación:

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 11:39:25 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 pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
```

```
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           MOUNTED
         4 JSON                           READ WRITE NO
         5 SOE                            READ WRITE NO
         7 PDBX                           READ WRITE NO


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 SOA                            READ WRITE NO
         6 PDB1                           READ WRITE NO

SQL> alter session set container=PDB1;

Session altered.

SQL> desc system.prueba2
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
-
 I                                                  NUMBER

SQL> exit
```

Finalmente verificamos que “PDB1” ya no existe en CDBB:

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 11:43:06 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 pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
```

```
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         4 PDB2                           READ WRITE NO
         5 JSON                           READ WRITE NO

SQL> exit
```

Refreshable PDB

Explicación

Una PDB de tipo “refreshable” es una base de datos que puede estar en modo “mounted” o en modo “read only”, y que se puede refrescar/actualizar en base a una PDB de referencia. De ésta forma, una refreshable PDB actualizada, es una copia de una PDB de referencia que puede estar tanto en la misma CDB, o en otra CDB sea local o remota. Ésta característica se usa especialmente en casos donde necesitemos refrescar periódicamente entornos no productivos, y queramos tener una copia actualizada de producción en el entorno no-productivo, que sirva como bastión para crear las n copias de los entornos previos cuando se requiera un refresco.

Para crear una refreshable PDB necesitaremos igualmente un DBLink, creado desde la CDB donde esté la refreshable PDB, hacia la CDB donde se encuentre la BDD original.

Práctica de Refreshable PDB

Para ésta práctica utilizaremos “PDB1” de la CDBA como base de datos de referencia/origen. Y crearemos “REFRE” como su refreshable PDB en CDBB.

Dado que en las prácticas anteriores ya creamos dblinks entre los contenedores (en ambos sentidos), pasaremos directamente a la creación de la refreshable PDB:

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 11:56:29 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 refre from PDB1@linkclonado refresh mode manual;

Pluggable database created.
```

La refreshable PDB se quedará en estado “mounted” inicialmente, y podremos abrirla en modo “read only” para trabajar con ella cuando se requiera. En éste caso la pondremos en estado read only para verificar ésta posibilidad, y volveremos a dejarla en estado mounted para practicar un refresco:

```
SQL> show pdbs

    CON_ID CON_NAME                  OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                  READ ONLY  NO
         3 REFRE                     MOUNTED
         5 JSON                      READ WRITE NO
SQL> alter pluggable database refre open read only;

Pluggable database altered.

SQL> show pdbs

    CON_ID CON_NAME                  OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                  READ ONLY  NO
         3 REFRE                     READ ONLY  NO
         5 JSON                      READ WRITE NO
SQL> alter pluggable database REFRE close;

Pluggable database altered.

SQL> exit
```

A continuación, crearemos en la BDD origen/referencia una nueva tabla, y posteriormente ejecutaremos un refresco de la refreshable PDB para verificar que se ha puesto al día con los cambios de la BDD de referencia.

Creamos la tabla en origen:

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 12:00:48 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 session set container=PDB1;

Session altered.

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

Table created.
```

```
SQL> exit
```

Refrescamos “REFRE” y verificamos que está al día :

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 12:01:54 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 pdbs

    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         4 PDB2                           READ WRITE NO
         5 JSON                           READ WRITE NO
         6 REFRE                          MOUNTED

SQL> alter pluggable database REFRE refresh;

Pluggable database altered.

SQL> alter pluggable database refre open read only;

Pluggable database altered.

SQL> alter session set container=refre;

Session altered.

SQL> desc system.refre
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
-
 I                                                  NUMBER

SQL> exit
```

Snapshot Clone

Explicación

Una snapshot clone de una PDB es una PDB clon que se puede abrir en lectura/escritura sin requerir el 100% del espacio que ocupa la base de datos original. Los dos principales condicionantes para hacer esto posible son:

A) Que la base de datos original/padre permanezca montada o en estado read/only mientras dure la vida del clon.

B) Que tanto la base de datos original como la clonada estén sobre una tecnología de almacenamiento que permita thin provisioning (como ACFS).

El caso de uso principal de esta tecnología se da cuando necesitamos generar múltiples copias de una misma base de datos (ej. una base de datos de desarrollo por cada desarrollador) sin proveer el 100% del espacio por cada copia. Concretamente, el único espacio que requiere la snapshot clon es aquel destinado a su propio UNDO y temporal. Los datafiles de usuario se apuntan a la base de datos original/padre. En caso de que se haga alguna modificación en la snapshot clone (que es lectura-escritura), dicho bloque se almacenará de forma local a esa PDB.

Gracias a esto se pueden crear, por ejemplo, 5 copias de una base de datos de 200 TB ocupando un total de – a modo de ejemplo – 200,1 TB (20 Gb de undo y temporal por cada bdd).

Práctica de Snapshot Clone

Para la siguiente práctica hemos precreado un punto de montaje ACFS llamado "/miacfs", donde ubicaremos tanto la base de datos original, como la copia "snapshot clone".

Antes de comenzar vamos a obtener el nivel de ocupación de ese filesystem. Con éste dato podremos confirmar la ocupación del snapshot clone:

```
[oracle@single19c ~]$ /sbin/acfsutil info fs /miacfs | egrep -i "size|free"
    metadata block size:   4096
    total size:   4294967296  (   4.00 GB )
    total free:   2900525056  (   2.70 GB )
        logical sector size:   512
        size:                  4294967296  (   4.00 GB )
        free:                  2900525056  (   2.70 GB )
        ADVM resize increment: 67108864
```

Vamos a proceder a crear la PDB de origen/padre y verificar que sus datafiles están en ACFS:

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 15:56:15 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 A admin user a identified by a
create_file_dest='/miacfs';

Pluggable database created.

SQL> alter pluggable database A 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 JSON                           READ WRITE NO
         5 SOE                            READ WRITE NO
         6 A                              READ WRITE NO
         7 PDBX                           READ WRITE NO

SQL> set linesize 999
SQL> select * from cdb_data_files where con_id=6; -- atención, se debe
reemplazar el identificador por el que acabamos de obtener

FILE_NAME

                                 FILE_ID TABLESPACE_NAME                  BYTES
BLOCKS STATUS    RELATIVE_FNO AUT   MAXBYTES  MAXBLOCKS INCREMENT_BY USER_BYTES
USER_BLOCKS ONLINE_ LOST_WR         CON_ID
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------------------ ---------- ----------------------------
```

```
-- ---------- ---------- --------- ------------ --- ---------- ---------- -----
------- ---------- ----------- ------- ------- ----------
/miacfs/CDBA_FRA17W/D501086D1E493E4EE053C900000A4B7B/datafile/o1_mf_undotbs1_jx
jrotf7_.dbf
                                         30 UNDOTBS1                    47185920
          5760 AVAILABLE        8 YES 3.4360E+10    4194302         640
46137344       5632 ONLINE  OFF         3
/miacfs/CDBA_FRA17W/D501086D1E493E4EE053C900000A4B7B/datafile/o1_mf_sysaux_jxjr
otf3_.dbf
                                         29 SYSAUX                     429916160
          52480 AVAILABLE        4 YES 3.4360E+10    4194302        1280
428867584     52352 ONLINE  OFF         3
/miacfs/CDBA_FRA17W/D501086D1E493E4EE053C900000A4B7B/datafile/o1_mf_system_jxjr
otf0_.dbf
                                         28 SYSTEM                     356515840
          43520 AVAILABLE        1 YES 3.4360E+10    4194302        1280
355467264     43392 SYSTEM  OFF         3
```

Ahora que hemos verificado que los datafiles están creados en ACFS y que la BDD se ha abierto correctamente, la ponemos en read-only, y procedemos a crear la Snapshot Clone:

```
SQL> show con_name

CON_NAME
------------------------------
CDB$ROOT
SQL> alter pluggable database A close immediate;

Pluggable database altered.

SQL> alter pluggable database A open read only;

Pluggable database altered.

SQL> create pluggable database B from A create_file_dest='/miacfs' snapshot
copy;

Pluggable database created.

SQL> alter pluggable database B 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 JSON                     READ WRITE NO
         5 SOE                      READ WRITE NO
         6 A                        READ ONLY  NO
```

```
         7 PDBX                         READ WRITE NO
         8 B                            READ WRITE NO
```

Vamos a confirmar que sus datafiles también están en ACFS, y posteriormente, vamos a comprobar el grado de ocupación del filesystem:

```
SQL> set linesize 999
SQL> select * from cdb_data_files where con_id=8; -- atención, se debe
reemplazar el identificador por el que acabamos de obtener

FILE_NAME

                                FILE_ID TABLESPACE_NAME                    BYTES
BLOCKS STATUS    RELATIVE_FNO AUT   MAXBYTES  MAXBLOCKS INCREMENT_BY USER_BYTES
USER_BLOCKS ONLINE_ LOST_WR         CON_ID
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
------------------------------------ ---------- ----------------------------
-- ---------- ---------- --------- ------------ --- ---------- ---------- -----
------- ---------- ----------- ------- ------- ----------
/miacfs/CDBA_FRA17W/D50123DB88105941E053C900000A88D9/datafile/o1_mf_undotbs1_jx
js46fo_.dbf
                                     33 UNDOTBS1                   47185920
           5760 AVAILABLE        8 YES 3.4360E+10    4194302         640
46137344       5632 ONLINE  OFF        7
/miacfs/CDBA_FRA17W/D50123DB88105941E053C900000A88D9/datafile/o1_mf_sysaux_jxjs
46fm_.dbf
                                     32 SYSAUX                     429916160
          52480 AVAILABLE        4 YES 3.4360E+10    4194302        1280
428867584     52352 ONLINE  OFF        7
/miacfs/CDBA_FRA17W/D50123DB88105941E053C900000A88D9/datafile/o1_mf_system_jxjs
46fh_.dbf
                                     31 SYSTEM                     356515840
          43520 AVAILABLE        1 YES 3.4360E+10    4194302        1280
355467264     43392 SYSTEM  OFF        7

SQL> exit
Disconnected from Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 -
Production
Version 19.12.0.0.0
[oracle@single19c ~]$ /sbin/acfsutil info fs /miacfs | egrep -i "size|free"
    metadata block size:   4096
    total size:   4294967296  (   4.00 GB )
    total free:   2678231040  (   2.49 GB )
        logical sector size:   512
        size:                  4294967296  (   4.00 GB )
```

```
        free:                  2678231040  (   2.49 GB )
        ADVM resize increment: 67108864
```

Antes de la creación de ambas PDBS, el espacio libre era de 2,70Gb. Actualmente el espacio libre es de 2,49 Gb, por lo que ambas PDBs ocupan de forma conjunta 0,21 Gb. Vamos a comprobar qué proporción de ese espacio está ocupando la snapshot :

```
[oracle@single19c ~]$ /sbin/acfsutil info fs /miacfs | grep -i snapshot
    number of snapshots:  1
    snapshot space usage: 397312  ( 388.00 KB )
```

ASM Split Mirror

Explicación

Gracias a los diskgroups de tipo “FLEX”, podemos crear un clon de una PDB en la misma contenedora de forma instantánea y sin depender de los datafiles de la PDB original. Es decir, es un clon completo e instantáneo de la PDB original.

Ésta tecnología se basa en dos etapas:

A) Preparación de la copia: De forma previa (pueden ser días o semanas antes) al momento en el que se necesite la copia instantánea, se ordena a la base de datos original que comience a preparar una copia de la base de datos. Desde ese momento, ASM comenzará a duplicar los Allocation Units (unidad de almacenamiento en un diskgroup de ASM) de dicha base de datos, como si fuera un aumento temporal de su redundancia. Una vez finalice esa copia inicial de los AUs, la base de datos original mantendrá la copia en estado “Prepared”; sincronizada en todo momento y “sine die” hasta que se consuma.

B) Consumo de la copia: En el momento en el que se requiera la copia instantánea, se ordena la creación de la nueva PDB empleando la copia preparada. La operación finalizará en pocos segundos de forma independiente al tamaño que tenga la base de datos original.

Práctica de ASM Split Mirror

Emplearemos la base de datos “PDBX” para realizar ésta práctica, ya que previamente la hemos creado en el diskgroup “RECO”, que a su vez ha sido creado con redundancia “FLEX”.

Comenzamos lanzando el primer paso, que es la preparación de la copia:

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 16:28:51 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=PDBX;

Session altered.
```

```
SQL> show con_name

CON_NAME
------------------------------
PDBX

SQL> alter pluggable database prepare mirror copy test;

Pluggable database altered.
```

El prompt se devuelve de forma inmediata, pero el trabajo de duplicado se queda funcionando en background. Vamos a comprobar cómo evoluciona la preparación :

```
SQL> set linesize 999
SQL> column MIRRORCOPY_NAME format a20
SQL> column DBCLONE_STATUS format a20
SQL> column PARENT_DBNAME format a20
SQL> select group_number, mirrorcopy_name, dbclone_status, parent_dbname from
v$asm_dbclone_info;

GROUP_NUMBER MIRRORCOPY_NAME       DBCLONE_STATUS       PARENT_DBNAME
------------ -------------------- -------------------- --------------------
           2 TEST                 PREPARING            CDBA_FRA2RX_PDBX
```

Dado que es una base de datos pequeña, si esperamos un minuto veremos que la copia ya ha finalizado:

```
SQL> select group_number, mirrorcopy_name, dbclone_status, parent_dbname from
v$asm_dbclone_info;

GROUP_NUMBER MIRRORCOPY_NAME       DBCLONE_STATUS       PARENT_DBNAME
------------ -------------------- -------------------- --------------------
           2 TEST                 PREPARED             CDBA_FRA2RX_PDBX
```

Una vez preparada la copia instantánea, volvemos a la CDB$ROOT, y la usamos para crear una nueva PDB:

```
SQL> alter session set container=CDB$ROOT;

Session altered.

SQL> create pluggable database PDBX2 from PDBX using mirror copy test ;

Pluggable database created.

SQL> show pdbs
```

```
    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
         6 A                              READ ONLY  NO
         7 PDBX                           READ WRITE NO
         8 B                              READ WRITE NO
        10 PDBX2                          MOUNTED

SQL> alter pluggable database PDBX2 open;

Pluggable database altered.

SQL> exit
```

El comando de clonado ha finalizado de forma casi instantánea, y ya tenemos a nuestra disposición la copia completa de la BDD y en modo read/write.

Para terminar, recomendamos ejecutar los siguientes pasos, que eliminarán las PDBs innecesarias para los siguientes ejercicios

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

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jan 7 16:35:50 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 all close immediate;

Pluggable database altered.

SQL> drop pluggable database PDB1 including datafiles;

Pluggable database dropped.

SQL> drop pluggable database A including datafiles;

Pluggable database dropped.

SQL> drop pluggable database B including datafiles;
```

```
Pluggable database dropped.

SQL> drop pluggable database including datafiles;

Pluggable database dropped.

SQL> drop pluggable database PDBX2 including datafiles;

Pluggable database dropped.


SQL> alter pluggable database all 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
SQL> exit
```

Finalmente, detendremos la contenedora CDBB para liberar recursos de la máquina :

[oracle@single19c ~]$. CDBB.env
[oracle@single19c ~]$ srvctl stop database -d CDBB -o immediate

JSON

Preparación de entradas tnsnames.ora

Antes de comenzar la práctica añadiremos dos nuevas entradas al final del tnsnames.ora de nuestro Oracle Home 19c:

```
[oracle@single19c ~]$  vi $ORACLE_HOME/network/admin/tnsnames.ora

JSON =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = single19c)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = JSON.tfexsubdbsys.tfexvcndbsys.oraclevcn.com)
    )
  )

SOE =
  (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = single19c)(PORT = 1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = SOE.tfexsubdbsys.tfexvcndbsys.oraclevcn.com)
    )
  )
```

Creación de un modelo de datos políglota, incluir ORDS en una PDB e inyección de documentos JSON en la base de datos (35 min)

En primer lugar, hay que crear el usuario sodauser, que tendrá activados todos los servicios REST, entre ellos SODA (Simple Oracle Document Access) que es el que se emplea en esta primera parte de la sección JSON.

Para ello se va a crear el tablespace users, se va a crear el usuario sodauser y se le van a dar los permisos necesarios para las siguientes operativas.

```
$ cd $HOME
$ . CDBA.env
$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 12 09:51:25 2022
Version 19.13.0.0.0

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


Connected to:
```

```
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.13.0.0.0

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

SQL> ALTER SESSION SET CONTAINER=json;

Session altered.

SQL> create tablespace users;

Tablespace created.

SQL> CREATE USER sodauser IDENTIFIED BY sodauser1
  DEFAULT TABLESPACE users QUOTA UNLIMITED ON users;

User created.

SQL> GRANT CREATE SESSION, CREATE TABLE TO sodauser;

Grant succeeded.

SQL> grant connect, resource to sodauser;

Grant succeeded.

SQL> grant create view to sodauser;

Grant succeeded.

SQL> GRANT SODA_APP TO sodauser;

Grant succeeded.

GRANT CREATE ANY DIRECTORY TO sodauser;

Grant succeeded.
```

A continuación, hay que conectar con el usuario sodauser a la PDB con el nombre JSON.

```
SQL> CONN sodauser/sodauser1@json

Connected.
```

Se activa el esquema sodauser para los servicios REST:

```
SQL> DECLARE
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN

    ORDS.ENABLE_SCHEMA(p_enabled => TRUE,
                       p_schema => 'SODAUSER',
                       p_url_mapping_type => 'BASE_PATH',
                       p_url_mapping_pattern => 'sodauser',
                       p_auto_rest_auth => FALSE);
    commit;
END;
/

PL/SQL procedure successfully completed.
```

Para este taller se van a eliminar las restricciones de seguridad. Este paso no es recomendado en entornos de producción. Solo se hace por motivos de simplicidad y en un entorno de demostración controlado.

```
SQL> BEGIN
ORDS.delete_privilege_mapping('oracle.soda.privilege.developer','/soda/*');
COMMIT;
END;
/

PL/SQL procedure successfully completed.
```

A partir de este momento, ya se puede interactuar con este usuario a través del REST API que está escuchando en el puerto 8080 del servidor de base de datos. Para componer la URL es necesario el nombre de la PDB (json) y del esquema (sodauser), como se muestra en los ejemplos a continuación.

Crear una colección mediante REST API usando la herramienta *curl,* que realiza una petición HTTP de tipo PUT:

```
$ curl -i -X PUT
http://single19c:8080/ords/json/sodauser/soda/latest/TestCollectionREST

HTTP/1.1 201 Created
Date: Wed, 12 Jan 2022 10:01:23 GMT
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0
Location:
http://single19c:8080/ords/json/sodauser/soda/latest/TestCollectionREST/
Content-Length: 0
```

Comprobar cómo se ha creado la colección en la base de datos:

```
[oracle@single19c ~]$ sqlplus sodauser/sodauser1@json

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 12 10:02:28 2022
Version 19.13.0.0.0

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

Last Successful login time: Wed Jan 12 2022 09:57:41 +00:00

Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.13.0.0.0

SQL> desc "TestCollectionREST"

Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
-
 ID                                        NOT NULL VARCHAR2(255)
 CREATED_ON                                NOT NULL TIMESTAMP(6)
 LAST_MODIFIED                             NOT NULL TIMESTAMP(6)
 VERSION                                   NOT NULL VARCHAR2(255)
 JSON_DOCUMENT                                      BLOB
```

Refs:

https://docs.oracle.com/en/database/oracle/simple-oracle-document-access/adsdi/soda-collection-metadata-components-reference.html#GUID-127AC6E0-B27D-4261-B1C2-39E59A7F3C6D

Listar colecciones colección mediante REST API usando la herramienta *curl,* que realiza una petición HTTP de tipo GET:

```
$ curl -i -X GET http://single19c:8080/ords/json/sodauser/soda/latest

HTTP/1.1 200 OK
Date: Wed, 12 Jan 2022 10:05:09 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0
Content-Length: 634

{"items":[{"name":"TestCollectionREST","properties":{"schemaName":"SODAUSER","t
ableName":"TestCollectionREST","keyColumn":{"name":"ID","sqlType":"VARCHAR2","m
axLength":255,"assignmentMethod":"UUID"},"contentColumn":{"name":"JSON_DOCUMENT
","sqlType":"BLOB","compress":"NONE","cache":true,"encrypt":"NONE","validation"
:"STANDARD"},"versionColumn":{"name":"VERSION","type":"String","method":"SHA256
"},"lastModifiedColumn":{"name":"LAST_MODIFIED"},"creationTimeColumn":{"name":"
CREATED_ON"},"readOnly":false},"links":[{"rel":"canonical","href":"http://singl
```

```
e19c:8080/ords/json/sodauser/soda/latest/TestCollectionREST"}]}],"hasMore":fals
e}[oracle@single19c ~]$
```

Crear documentos JSON desde tablas

A continuación, crear varios ficheros que contienen un documento JSON con información que procede de una consulta sobre datos estructurados del modelo de datos SOE perteneciente a la herramienta Swingbench.
Tras la ejecución de este script, deben aparecer en la carpeta home del usuario oracle varios ficheros con nombres:
`order_##.json`
compruebe que el contenido de estos ficheros es un documento JSON.

```
[oracle@single19c ~]$ sqlplus sys/We1c0m3_We1c0m3_@SOE as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 12 10:10:17 2022
Version 19.13.0.0.0

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


Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.13.0.0.0

SQL> grant create any directory to SOE;
Grant succeeded.

SQL> conn soe/We1c0m3_We1c0m3_@SOE

Connected.

SQL> create directory ORDERS as '/home/oracle';

Directory created.

SQL> select table_name from user_tables;

TABLE_NAME
--------------------------------------------------------------------------------
-
CUSTOMERS
ADDRESSES
CARD_DETAILS
WAREHOUSES
ORDER_ITEMS
ORDERS
INVENTORIES
```

```
PRODUCT_INFORMATION
LOGON
PRODUCT_DESCRIPTIONS
ORDERENTRY_METADATA

11 rows selected.

SQL> @SQL2JSON.sql

SQL> quit;
```

Para más información se muestra el contenido del script SQL2JSON.sql que genera los ficheros que contienen los documentos JSON.
NOTA: No es necesario ejecutar este bloque puesto que ha sido ejecutado en el paso anterior, se incluye para referencia.

```
spool off;
set verify off;
Set Heading off
set echo off
set feedback off

DECLARE
  fHandle  UTL_FILE.FILE_TYPE;
  c integer := 0;
  CURSOR c_order
  IS
        (select JSON_OBJECT (
      'ORDER_ID' value ORDER_ID,
      'ORDER_DATE' value ORDER_DATE,
      'ORDER_MODE' value ORDER_MODE,
      'CUSTOMER_ID' value CUSTOMER_ID,
      'ORDER_STATUS' value ORDER_STATUS,
      'ORDER_TOTAL' value ORDER_TOTAL,
      'SALES_REP_ID' value SALES_REP_ID,
      'PROMOTION_ID' value PROMOTION_ID,
      'WAREHOUSE_ID' value WAREHOUSE_ID,
      'DELIVERY_TYPE' value DELIVERY_TYPE,
      'COST_OF_DELIVERY' value COST_OF_DELIVERY,
      'WAIT_TILL_ALL_AVAILABLE' value WAIT_TILL_ALL_AVAILABLE,
      'DELIVERY_ADDRESS_ID' value DELIVERY_ADDRESS_ID,
      'CUSTOMER_CLASS' value CUSTOMER_CLASS,
      'CARD_ID' value CARD_ID,
      'INVOICE_ADDRESS_ID' value INVOICE_ADDRESS_ID,
      'ORDER_ITEMS' value (
          select JSON_ARRAYAGG(
              JSON_OBJECT (
              'LINE_ITEM_ID' value LINE_ITEM_ID,
              'PRODUCT_ID' value PRODUCT_ID,
              'UNIT_PRICE' value UNIT_PRICE,
              'QUANTITY' value QUANTITY,
              'DISPATCH_DATE' value DISPATCH_DATE,
              'RETURN_DATE' value RETURN_DATE,
```

```
              'GIFT_WRAP' value GIFT_WRAP,
              'CONDITION' value CONDITION,
              'SUPPLIER_ID' value SUPPLIER_ID,
              'ESTIMATED_DELIVERY' value ESTIMATED_DELIVERY
              )
            ) from ORDER_ITEMS I where O.ORDER_ID = I.ORDER_ID
        )
  ) as jsonorders from ORDERS O where rownum < 20);
BEGIN
  FOR r_order IN c_order
  LOOP
    fHandle := UTL_FILE.FOPEN('ORDERS', 'order_'||c||'.json', 'w',32767);
    UTL_FILE.PUT_LINE(fHandle, r_order.jsonorders);
    UTL_FILE.FCLOSE(fHandle);
    c := c + 1;
  END LOOP;
END;
/
```

Insertar nuevos documentos JSON en una colección mediante REST API usando la herramienta *curl,* que realiza una petición HTTP de tipo POST.
Los documentos JSON que se van a cargar desde ficheros de texto llamados: order_0.json a order_18.json.

```
$ cat order_0.json

{"ORDER_ID":75283,"ORDER_DATE":"2008-04-
02T07:00:00.000000Z","ORDER_MODE":"online","CUSTOMER_ID":62257,"ORDER_STATUS":5
,"ORDER_TOTAL":4677,"SALES_REP_ID":468,"PROMOTION_ID":467,"WAREHOUSE_ID":103,"D
ELIVERY_TYPE":"Standard","COST_OF_DELIVERY":2,"WAIT_TILL_ALL_AVAILABLE":"ship_w
hen_ready","DELIVERY_ADDRESS_ID":4,"CUSTOMER_CLASS":"Occasional","CARD_ID":7577
4,"INVOICE_ADDRESS_ID":4,"ORDER_ITEMS":[{"LINE_ITEM_ID":1,"PRODUCT_ID":347,"UNI
T_PRICE":693,"QUANTITY":3,"DISPATCH_DATE":"2012-02-
03T00:00:00","RETURN_DATE":"2009-01-
21T00:00:00","GIFT_WRAP":"Boy_Birthday","CONDITION":"New","SUPPLIER_ID":347,"ES
TIMATED_DELIVERY":"2001-01-
08T00:00:00"},{"LINE_ITEM_ID":2,"PRODUCT_ID":439,"UNIT_PRICE":877,"QUANTITY":4,
"DISPATCH_DATE":"2012-03-
27T00:00:00","RETURN_DATE":null,"GIFT_WRAP":"None","CONDITION":"Used","SUPPLIER
_ID":439,"ESTIMATED_DELIVERY":"2000-11-
11T00:00:00"},{"LINE_ITEM_ID":3,"PRODUCT_ID":701,"UNIT_PRICE":1401,"QUANTITY":7
,"DISPATCH_DATE":"2012-04-
28T00:00:00","RETURN_DATE":null,"GIFT_WRAP":"Boy_Birthday","CONDITION":"New","S
UPPLIER_ID":701,"ESTIMATED_DELIVERY":"2006-04-
07T00:00:00"},{"LINE_ITEM_ID":4,"PRODUCT_ID":585,"UNIT_PRICE":1170,"QUANTITY":5
,"DISPATCH_DATE":"2012-04-
24T00:00:00","RETURN_DATE":null,"GIFT_WRAP":"Glossy","CONDITION":"New","SUPPLIE
R_ID":585,"ESTIMATED_DELIVERY":"2004-02-
22T00:00:00"},{"LINE_ITEM_ID":5,"PRODUCT_ID":347,"UNIT_PRICE":693,"QUANTITY":3,
```

```
"DISPATCH_DATE":"2012-01-
30T00:00:00","RETURN_DATE":null,"GIFT_WRAP":"Plain","CONDITION":"New","SUPPLIER
_ID":347,"ESTIMATED_DELIVERY":"2008-05-14T00:00:00"}]}


$ curl -i -X POST --data-binary @order_0.json -H "Content-Type:
application/json"
http://single19c:8080/ords/json/sodauser/soda/latest/TestCollectionREST


HTTP/1.1 100 Continue

HTTP/1.1 201 Created
Date: Wed, 12 Jan 2022 10:18:13 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0
Location:
http://single19c:8080/ords/json/sodauser/soda/latest/TestCollectionREST/5DE5ACD
5D86F47AF8A7EAADA23881AB5
Content-Length: 238

{"items":[{"id":"5DE5ACD5D86F47AF8A7EAADA23881AB5","etag":"B6E108BB51924B0D5915
AE9ABEB49D324B95BCB97D08C4B483D9BAACF9627272","lastModified":"2022-01-
12T10:18:14.322332Z","created":"2022-01-
12T10:18:14.322332Z"}],"hasMore":false,"count":1}[oracle@single19c ~]$


$ curl -i -X POST --data-binary @order_18.json -H "Content-Type:
application/json"
http://single19c:8080/ords/json/sodauser/soda/latest/TestCollectionREST
HTTP/1.1 201 Created
Date: Wed, 12 Jan 2022 10:19:09 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0
Location:
http://single19c:8080/ords/json/sodauser/soda/latest/TestCollectionREST/D59B49A
931CF4E5F88FBEE724C661102
Content-Length: 238

{"items":[{"id":"D59B49A931CF4E5F88FBEE724C661102","etag":"C41AE32BF336D15A4EFD
E2D6DCB7DED2431500D12A2023BC7A687F4EA6AEF4E7","lastModified":"2022-01-
12T10:19:09.478288Z","created":"2022-01-
12T10:19:09.478288Z"}],"hasMore":false,"count":1}[oracle@single19c ~]$
```

La operación de inserción de documento JSON devuelve un identificador único. Este identificador se usa en las siguientes operaciones donde debe sustituir la cadena <id_registro>.

Recuperar un listado completo de los ID de los documentos JSON insertados en la colección:

```
$ curl -i -X GET
http://single19c:8080/ords/json/sodauser/soda/latest/TestCollectionREST?fields=
id

HTTP/1.1 200 OK
Date: Wed, 12 Jan 2022 10:21:53 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0
Content-Length: 479

{"items":[{"id":"5DE5ACD5D86F47AF8A7EAADA23881AB5","etag":"B6E108BB51924B0D5915
AE9ABEB49D324B95BCB97D08C4B483D9BAACF9627272","lastModified":"2022-01-
12T10:18:14.322332Z","created":"2022-01-
12T10:18:14.322332Z"},{"id":"D59B49A931CF4E5F88FBEE724C661102","etag":"C41AE32B
F336D15A4EFDE2D6DCB7DED2431500D12A2023BC7A687F4EA6AEF4E7","lastModified":"2022-
01-12T10:19:09.478288Z","created":"2022-01-
12T10:19:09.478288Z"}],"hasMore":false,"count":2,"offset":0,"limit":100,"totalR
esults":2}[oracle@single19c ~]$
```

Consultar un documento JSON de una colección mediante REST API usando la herramienta *curl,* que realiza una petición HTTP de tipo GET (sustituir <id_registro> por un identificador válido que se habrá obtenido en el paso anterior):

```
curl -i -X GET
http://single19c:8080/ords/json/sodauser/soda/latest/TestCollectionREST/<id_reg
istro>

HTTP/1.1 200 OK
Date: Wed, 12 Jan 2022 10:23:24 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Cache-Control: no-cache,must-revalidate,no-store,max-age=0
ETag: B6E108BB51924B0D5915AE9ABEB49D324B95BCB97D08C4B483D9BAACF9627272
Last-Modified: Wed, 12 Jan 2022 10:18:14 UTC
Content-Length: 1542

{"ORDER_ID":75283,"ORDER_DATE":"2008-04-
02T07:00:00.000000Z","ORDER_MODE":"online","CUSTOMER_ID":62257,"ORDER_STATUS":5
,"ORDER_TOTAL":4677,"SALES_REP_ID":468,"PROMOTION_ID":467,"WAREHOUSE_ID":103,"D
ELIVERY_TYPE":"Standard","COST_OF_DELIVERY":2,"WAIT_TILL_ALL_AVAILABLE":"ship_w
hen_ready","DELIVERY_ADDRESS_ID":4,"CUSTOMER_CLASS":"Occasional","CARD_ID":7577
4,"INVOICE_ADDRESS_ID":4,"ORDER_ITEMS":[{"LINE_ITEM_ID":1,"PRODUCT_ID":347,"UNI
T_PRICE":693,"QUANTITY":3,"DISPATCH_DATE":"2012-02-
03T00:00:00","RETURN_DATE":"2009-01-
21T00:00:00","GIFT_WRAP":"Boy_Birthday","CONDITION":"New","SUPPLIER_ID":347,"ES
TIMATED_DELIVERY":"2001-01-
```

```
08T00:00:00"},{"LINE_ITEM_ID":2,"PRODUCT_ID":439,"UNIT_PRICE":877,"QUANTITY":4,
"DISPATCH_DATE":"2012-03-
27T00:00:00","RETURN_DATE":null,"GIFT_WRAP":"None","CONDITION":"Used","SUPPLIER
_ID":439,"ESTIMATED_DELIVERY":"2000-11-
11T00:00:00"},{"LINE_ITEM_ID":3,"PRODUCT_ID":701,"UNIT_PRICE":1401,"QUANTITY":7
,"DISPATCH_DATE":"2012-04-
28T00:00:00","RETURN_DATE":null,"GIFT_WRAP":"Boy_Birthday","CONDITION":"New","S
UPPLIER_ID":701,"ESTIMATED_DELIVERY":"2006-04-
07T00:00:00"},{"LINE_ITEM_ID":4,"PRODUCT_ID":585,"UNIT_PRICE":1170,"QUANTITY":5
,"DISPATCH_DATE":"2012-04-
24T00:00:00","RETURN_DATE":null,"GIFT_WRAP":"Glossy","CONDITION":"New","SUPPLIE
R_ID":585,"ESTIMATED_DELIVERY":"2004-02-
22T00:00:00"},{"LINE_ITEM_ID":5,"PRODUCT_ID":347,"UNIT_PRICE":693,"QUANTITY":3,
"DISPATCH_DATE":"2012-01-
30T00:00:00","RETURN_DATE":null,"GIFT_WRAP":"Plain","CONDITION":"New","SUPPLIER
_ID":347,"ESTIMATED_DELIVERY":"2008-05-14T00:00:00"}]}
```

Borrar un documento JSON mediante REST API usando la herramienta *curl,* que realiza una petición HTTP de tipo DELETE (sustituir <id_registro> por un identificador válido que se habrá obtenido en un paso anterior):

```
$ curl -i -X DELETE http://single19c
:8080/ords/json/sodauser/soda/latest/TestCollectionREST/<id_registro>

HTTP/1.1 200 OK
Date: Wed, 12 Jan 2022 10:27:14 GMT
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0
Content-Length: 0

$ curl -i -X GET
http://single19c:8080/ords/json/sodauser/soda/latest/TestCollectionREST//<id_re
gistro>

HTTP/1.1 404 Not Found
Date: Wed, 12 Jan 2022 10:28:34 GMT
Content-Type: application/json
X-Frame-Options: SAMEORIGIN
Transfer-Encoding: chunked

{"type":"http://www.w3.org/Protocols/rfc2616/rfc2616-
sec10.html#sec10.4.5","status":404,"title":"Key
D59B49A931CF4E5F88FBEE724C661102 not found in collection
TestCollectionREST.","o:errorCode":"REST-02001"}
```

Borrar una colección mediante REST API usando la herramienta *curl,* que realiza una petición HTTP de tipo DELETE:

```
$ curl -i -X DELETE http://
single19c:8080/ords/json/sodauser/soda/latest/TestCollectionREST

HTTP/1.1 200 OK
Date: Wed, 12 Jan 2022 10:31:32 GMT
X-Frame-Options: SAMEORIGIN
Cache-Control: private,must-revalidate,max-age=0
Content-Length: 0
```

Operar con SODA mediante comandos PL/SQL

Crear una colección mediante Comandos PL/SQL:

```
$ sqlplus sodauser/sodauser1@json

SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jan 12 10:32:47 2022
Version 19.13.0.0.0

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

Last Successful login time: Wed Jan 12 2022 10:02:29 +00:00

Connected to:
Oracle Database 19c EE Extreme Perf Release 19.0.0.0.0 - Production
Version 19.13.0.0.0

SQL> SET SERVEROUTPUT ON

SQL> DECLARE
  l_collection  SODA_COLLECTION_T;
BEGIN
  l_collection := DBMS_SODA.create_collection('TestCollectionSQL');
  IF l_collection IS NOT NULL THEN
    DBMS_OUTPUT.put_line('Collection ID = ' || l_collection.get_name());
  ELSE
    DBMS_OUTPUT.put_line('Collection does not exist.');
  END IF;
END;
/

Collection ID = TestCollectionSQL

PL/SQL procedure successfully completed.
```

Abrir una colección mediante comandos PL/SQL:

```
SQL> DECLARE
  l_collection  SODA_COLLECTION_T;
BEGIN
  l_collection := DBMS_SODA.open_collection('TestCollectionSQL');
  IF l_collection IS NOT NULL THEN
    DBMS_OUTPUT.put_line('Collection ID = ' || l_collection.get_name());
  ELSE
    DBMS_OUTPUT.put_line('Collection does not exist.');
  END IF;
END;
/

Collection ID = TestCollectionSQL

PL/SQL procedure successfully completed.
```

Listar colecciones existentes mediantes PL/SQL:

```
SQL> DECLARE
  l_coll_list  SODA_COLLNAME_LIST_T;
BEGIN
  l_coll_list := DBMS_SODA.list_collection_names;

  IF l_coll_list.COUNT > 0 THEN
    FOR i IN 1 .. l_coll_list.COUNT LOOP
      DBMS_OUTPUT.put_line(i || ' = ' || l_coll_list(i));
    END LOOP;
  END IF;
END;
/

1 = TestCollectionSQL

PL/SQL procedure successfully completed.
```

Insertar nuevos documentos JSON en una colección mediante PL/SQL:

```
SET SERVEROUTPUT ON

SQL> DECLARE
  l_collection    SODA_COLLECTION_T;
  l_document      SODA_DOCUMENT_T;
  l_document_out  SODA_DOCUMENT_T;
BEGIN
  l_collection := DBMS_SODA.open_collection('TestCollectionSQL');

  l_document := SODA_DOCUMENT_T(
                  b_content =>
UTL_RAW.cast_to_raw('{"employee_number":7521,"employee_name":"WARD"}')
                );

  l_document_out := l_collection.insert_one_and_get(l_document);
```

```
  DBMS_OUTPUT.put_line('key       : ' || l_document_out.get_key);
  DBMS_OUTPUT.put_line('content   : ' ||
UTL_RAW.cast_to_varchar2(l_document_out.get_blob));
  DBMS_OUTPUT.put_line('media_type: ' || l_document_out.get_media_type);
  COMMIT;
END;
/

key        : 7C9E25AC44374FC1BF26CB749BCEB05F
content    :  media_type: application/json

PL/SQL procedure successfully completed.
```

Consultar un documento JSON de una colección mediante PL/SQL:

```
SQL> DECLARE
  l_collection    SODA_COLLECTION_T;
  l_document      SODA_DOCUMENT_T;
BEGIN
  l_collection := DBMS_SODA.open_collection('TestCollectionSQL');

  l_document := l_collection.find_one('&ID_REG');

  DBMS_OUTPUT.put_line('key       : ' || l_document.get_key);
  DBMS_OUTPUT.put_line('content   : ' ||
UTL_RAW.cast_to_varchar2(l_document.get_blob));
  DBMS_OUTPUT.put_line('media_type: ' || l_document.get_media_type);
  COMMIT;
END;
/

Enter value for id_reg: 7C9E25AC44374FC1BF26CB749BCEB05F
old   7:   l_document := l_collection.find_one('&ID_REG');
new   7:   l_document :=
l_collection.find_one('7C9E25AC44374FC1BF26CB749BCEB05F');
key       : 7C9E25AC44374FC1BF26CB749BCEB05F
content   : {"employee_number":7521,"employee_name":"WARD"}
media_type: application/json

PL/SQL procedure successfully completed.
```

Borrar una colección mediantes PL/SQL:

```
SQL> DECLARE
  l_status  NUMBER := 0;
BEGIN
  l_status := DBMS_SODA.drop_collection('TestCollectionSQL');
  DBMS_OUTPUT.put_line('l_status=' || l_status);
END;
/
```

```
l_status=1

PL/SQL procedure successfully completed.
```

Manejo de documentos JSON con SQL

A continuación, se muestra cómo consultar información en documentos JSON desde SQL. En primer lugar, se crea una tabla con una columna de tipo CLOB que almacenará los documentos JSON con una constraint que comprueba su validez y sobre esta tabla se insertan algunos documentos JSON y otros datos:

```
SQL> CONN sodauser/sodauser1@json

Connected.

SQL> CREATE TABLE json_documents (
  id    RAW(16) NOT NULL,
  data  CLOB,
  CONSTRAINT json_documents_pk PRIMARY KEY (id),
  CONSTRAINT json_documents_json_chk CHECK (data IS JSON)
);
INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "John",
          "LastName"       : "Doe",
          "Job"            : "Clerk",
          "Address"        : {
                              "Street"   : "99 My Street",
                              "City"     : "My City",
                              "Country"  : "UK",
                              "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                              "Email"    : "john.doe@example.com",
                              "Phone"    : "44 123 123456",
                              "Twitter"  : "@johndoe"
                             },
          "DateOfBirth"    : "01-JAN-1980",
          "Active"         : true
         }');

INSERT INTO json_documents (id, data)
VALUES (SYS_GUID(),
        '{
          "FirstName"      : "Jayne",
          "LastName"       : "Doe",
          "Job"            : "Manager",
          "Address"        : {
```

```
                              "Street"   : "100 My Street",
                              "City"     : "My City",
                              "Country"  : "UK",
                              "Postcode" : "A12 34B"
                             },
          "ContactDetails" : {
                              "Email"    : "jayne.doe@example.com",
                              "Phone"    : ""
                             },
          "DateOfBirth"    : "01-JAN-1982",
          "Active"         : false
         }');

COMMIT;

Table created.

1 row created.

1 row created.

Commit complete.
```

Una vez introducidos los documentos JSON y los datos, se pueden hacer consultas sobre los atributos de los documentos JSON usando SQL.

```
col firstname format a10
col LASTNAME format a10
col POSTCODE format a10
col TWITTER format a10
col PHONE format a10
col EMAIL format a25

SQL> SELECT a.data.FirstName,
       a.data.LastName,
       a.data.Address.Postcode AS Postcode,
       a.data.ContactDetails.Email AS Email
FROM   json_documents a
ORDER BY a.data.FirstName,
         a.data.LastName;

FIRSTNAME  LASTNAME   POSTCODE   EMAIL
---------- ---------- ---------- -------------------------
Jayne      Doe        A12 34B    jayne.doe@example.com
John       Doe        A12 34B    john.doe@example.com

SQL> SELECT a.data.ContactDetails
FROM   json_documents a;

CONTACTDETAILS
--------------------------------------------------------------------------------
-
{"Email":"john.doe@example.com","Phone":"44 123 123456","Twitter":"@johndoe"}
```

```
{"Email":"jayne.doe@example.com","Phone":""}


SQL> SELECT a.data.FirstName,
       a.data.LastName,
       a.data.ContactDetails.Email AS Email,
       a.data.ContactDetails.Phone AS Phone,
       a.data.ContactDetails.Twitter AS Twitter
FROM   json_documents a
WHERE  a.data.ContactDetails.Phone IS NULL
AND    a.data.ContactDetails.Twitter IS NULL;

FIRSTNAME  LASTNAME   EMAIL                     PHONE      TWITTER
---------- ---------- ------------------------- ---------- ----------
Jayne      Doe        jayne.doe@example.com


-- chequea que exista el elemento Phone, pero tiene un valor nulo

SQL> SELECT a.data.FirstName,
       a.data.LastName,
       a.data.ContactDetails.Email AS Email
FROM   json_documents a
WHERE  JSON_EXISTS(a.data.ContactDetails, '$.Phone' FALSE ON ERROR)
AND    a.data.ContactDetails.Phone IS NULL;

FIRSTNAME  LASTNAME   EMAIL
---------- ---------- -------------------------
Jayne      Doe        jayne.doe@example.com

-- chequea registros donde falta el elemento Twitter .

SQL> SELECT a.data.FirstName,
       a.data.LastName,
       a.data.ContactDetails.Email AS Email
FROM   json_documents a
WHERE  NOT JSON_EXISTS(a.data.ContactDetails, '$.Twitter' FALSE ON ERROR);

FIRSTNAME  LASTNAME   EMAIL
---------- ---------- -------------------------
Jayne      Doe        jayne.doe@example.com
```

Más ejemplos de consultas usando JSON_VALUE:

```
SQL> SELECT JSON_VALUE(a.data, '$.FirstName') AS firstname,
       JSON_VALUE(a.data, '$.LastName') AS lastname
FROM   json_documents a
ORDER BY 1, 2;

FIRSTNAME  LASTNAME
---------- ----------
Jayne      Doe
```

```
John       Doe

--consulta accediendo por notación de punto

SQL> SELECT a.data.ContactDetails
FROM   json_documents a;

CONTACTDETAILS
--------------------------------------------------------------------------------
-
{"Email":"john.doe@example.com","Phone":"44 123 123456","Twitter":"@johndoe"}
{"Email":"jayne.doe@example.com","Phone":""}


--json_value falla cuando el resultado no es un valor escalar

SQL> SELECT JSON_VALUE(a.data, '$.ContactDetails') AS contact_details
FROM   json_documents a
ORDER BY 1;

CONTACT_DETAILS
--------------------------------------------------------------------------------
-

-- misma consulta, pero con gestión de errores para json_value

SQL> SELECT JSON_VALUE(a.data, '$.ContactDetails' ERROR ON ERROR) AS
contact_details
FROM   json_documents a
ORDER BY 1;
ERROR at line 2:
ORA-40456: JSON_VALUE evaluated to non-scalar value
```

Creación de una vista que transforma el contenido del documento JSON para que pueda consultarse usando las columnas de mapeo de la vista.

```
SQL> CREATE OR REPLACE VIEW json_documents_v AS
SELECT jt.firstname,
       jt.lastname,
       jt.job,
       jt.addr_street,
       jt.addr_city,
       jt.addr_country,
       jt.addr_postcode,
       jt.email,
       jt.phone,
       jt.twitter,
       TO_DATE(jt.dob, 'DD-MON-YYYY') AS dob,
       jt.active
FROM   json_documents,
       JSON_TABLE(data, '$'
         COLUMNS (firstname    VARCHAR2(50 CHAR) PATH '$.FirstName',
```

```
                  lastname     VARCHAR2(50 CHAR) PATH '$.LastName',
                  job           VARCHAR2(10 CHAR) PATH '$.Job',
                  addr_street   VARCHAR2(50 CHAR) PATH '$.Address.Street',
                  addr_city     VARCHAR2(50 CHAR) PATH '$.Address.City',
                  addr_country  VARCHAR2(50 CHAR) PATH '$.Address.Country',
                  addr_postcode VARCHAR2(50 CHAR) PATH '$.Address.Postcode',
                  email         VARCHAR2(100 CHAR) PATH
'$.ContactDetails.Email',
                  phone         VARCHAR2(50 CHAR) PATH
'$.ContactDetails.Phone',
                  twitter       VARCHAR2(50 CHAR) PATH
'$.ContactDetails.Twitter',
                  dob           VARCHAR2(11 CHAR) PATH '$.DateOfBirth',
                  active        VARCHAR2(5 CHAR) PATH '$.Active')) jt;

View created.

SQL> SELECT firstname, lastname, dob
FROM   json_documents_v
ORDER BY firstname, lastname;

FIRSTNAME  LASTNAME   DOB
---------- ---------- ---------
Jayne      Doe        01-JAN-82
John       Doe        01-JAN-80
```

Extracción de metadatos de la estructura JSON usando json dataguide (json_dataguide):

```
SQL> set long 1000000
SQL> set linesize 1000

SQL> CREATE SEARCH INDEX json_docs_search_idx ON json_documents (data) FOR
JSON;

Index created.

--formato plano

SQL> SELECT DBMS_JSON.get_index_dataguide(
         'json_documents',
         'data',
         DBMS_JSON.format_flat,
         DBMS_JSON.pretty) AS dg
FROM   dual;

DG
--------------------------------------------------------------------------------
-
[
  {
    "o:path" : "$",
    "type" : "object",
```

```
    "o:length" : 512,
    "o:preferred_column_name" : "DATA$"
  },
  {
    "o:path" : "$.Job",
    "type" : "string",
    "o:length" : 8,

DG
--------------------------------------------------------------------------------
-
    "o:preferred_column_name" : "DATA$Job"
  },
  {
    "o:path" : "$.Active",
    "type" : "boolean",
    "o:length" : 8,
    "o:preferred_column_name" : "DATA$Active"
  },
  {
    "o:path" : "$.Address",
    "type" : "object",

DG
--------------------------------------------------------------------------------
-
    "o:length" : 128,
    "o:preferred_column_name" : "DATA$Address"
  },
  {
    "o:path" : "$.Address.City",
    "type" : "string",
    "o:length" : 8,
    "o:preferred_column_name" : "DATA$City"
  },
  {
    "o:path" : "$.Address.Street",

DG
--------------------------------------------------------------------------------
-
    "type" : "string",
    "o:length" : 16,
    "o:preferred_column_name" : "DATA$Street"
  },
  {
    "o:path" : "$.Address.Country",
    "type" : "string",
    "o:length" : 2,
    "o:preferred_column_name" : "DATA$Country"
  },
  {

DG
```

```
--------------------------------------------------------------------------------
-
    "o:path" : "$.Address.Postcode",
    "type" : "string",
    "o:length" : 8,
    "o:preferred_column_name" : "DATA$Postcode"
  },
  {
    "o:path" : "$.LastName",
    "type" : "string",
    "o:length" : 4,
    "o:preferred_column_name" : "DATA$LastName"
  },

DG
--------------------------------------------------------------------------------
-
  {
    "o:path" : "$.FirstName",
    "type" : "string",
    "o:length" : 8,
    "o:preferred_column_name" : "DATA$FirstName"
  },
  {
    "o:path" : "$.DateOfBirth",
    "type" : "string",
    "o:length" : 16,
    "o:preferred_column_name" : "DATA$DateOfBirth"

DG
--------------------------------------------------------------------------------
-
  },
  {
    "o:path" : "$.ContactDetails",
    "type" : "object",
    "o:length" : 128,
    "o:preferred_column_name" : "DATA$ContactDetails"
  },
  {
    "o:path" : "$.ContactDetails.Email",
    "type" : "string",
    "o:length" : 32,

DG
--------------------------------------------------------------------------------
-
    "o:preferred_column_name" : "DATA$Email"
  },
  {
    "o:path" : "$.ContactDetails.Phone",
    "type" : "string",
    "o:length" : 16,
    "o:preferred_column_name" : "DATA$Phone"
```

```
  },
  {
    "o:path" : "$.ContactDetails.Twitter",
    "type" : "string",

DG
--------------------------------------------------------------------------------
-
    "o:length" : 8,
    "o:preferred_column_name" : "DATA$Twitter"

--formato jerárquico

SQL> SELECT DBMS_JSON.get_index_dataguide(
         'json_documents',
         'data',
         DBMS_JSON.format_hierarchical,
         DBMS_JSON.pretty) AS dg
FROM   dual;

DG
--------------------------------------------------------------------------------
-
{
  "type" : "object",
  "o:length" : 512,
  "properties" :
  {
    "Job" :
    {
      "type" : "string",
      "o:length" : 8,
      "o:preferred_column_name" : "DATA$Job"
    },

DG
--------------------------------------------------------------------------------
-
    "Active" :
    {
      "type" : "boolean",
      "o:length" : 8,
      "o:preferred_column_name" : "DATA$Active"
    },
    "Address" :
    {
      "type" : "object",
      "o:length" : 128,
      "o:preferred_column_name" : "DATA$Address",

DG
--------------------------------------------------------------------------------
-
      "properties" :
```

```
      {
        "City" :
        {
          "type" : "string",
          "o:length" : 8,
          "o:preferred_column_name" : "DATA$City"
        },
        "Street" :
        {
          "type" : "string",

DG
--------------------------------------------------------------------------------
-
          "o:length" : 16,
          "o:preferred_column_name" : "DATA$Street"
        },
        "Country" :
        {
          "type" : "string",
          "o:length" : 2,
          "o:preferred_column_name" : "DATA$Country"
        },
        "Postcode" :
        {

DG
--------------------------------------------------------------------------------
-
          "type" : "string",
          "o:length" : 8,
          "o:preferred_column_name" : "DATA$Postcode"
        }
      }
    },
    "LastName" :
    {
      "type" : "string",
      "o:length" : 4,
      "o:preferred_column_name" : "DATA$LastName"

DG
--------------------------------------------------------------------------------
-
    },
    "FirstName" :
    {
      "type" : "string",
      "o:length" : 8,
      "o:preferred_column_name" : "DATA$FirstName"
    },
    "DateOfBirth" :
    {
      "type" : "string",
```

```
      "o:length" : 16,

DG
--------------------------------------------------------------------------------
-
      "o:preferred_column_name" : "DATA$DateOfBirth"
    },
    "ContactDetails" :
    {
      "type" : "object",
      "o:length" : 128,
      "o:preferred_column_name" : "DATA$ContactDetails",
      "properties" :
      {
        "Email" :
        {

DG
--------------------------------------------------------------------------------
-
          "type" : "string",
          "o:length" : 32,
          "o:preferred_column_name" : "DATA$Email"
        },
        "Phone" :
        {
          "type" : "string",
          "o:length" : 16,
          "o:preferred_column_name" : "DATA$Phone"
        },
        "Twitter" :

DG
--------------------------------------------------------------------------------
-
        {
          "type" : "string",
          "o:length" : 8,
          "o:preferred_column_name" : "DATA$Twitter"
        }
      }
    }
  }
}


--salida en una línea

SQL> SELECT JSON_DATAGUIDE(data) dg_doc
FROM   json_documents;

DG_DOC
--------------------------------------------------------------------------------
-
```

```
[{"o:path":"$","type":"object","o:length":512},{"o:path":"$.Job","type":"string
"
,"o:length":8},{"o:path":"$.Active","type":"boolean","o:length":8},{"o:path":"$
.
Address","type":"object","o:length":128},{"o:path":"$.Address.City","type":"str
i
ng","o:length":8},{"o:path":"$.Address.Street","type":"string","o:length":16},{
"
o:path":"$.Address.Country","type":"string","o:length":2},{"o:path":"$.Address.
P
ostcode","type":"string","o:length":8},{"o:path":"$.LastName","type":"string","
o
:length":4},{"o:path":"$.FirstName","type":"string","o:length":8},{"o:path":"$.
D
ateOfBirth","type":"string","o:length":16},{"o:path":"$.ContactDetails","type":
"
object","o:length":128},{"o:path":"$.ContactDetails.Email","type":"string","o:l
e
ngth":32},{"o:path":"$.ContactDetails.Phone","type":"string","o:length":16},{"o
:
path":"$.ContactDetails.Twitter","type":"string","o:length":8}]
```

--otra forma de presentar la información tipo tabla

```
SQL> WITH dg_t AS (
  SELECT JSON_DATAGUIDE(data) dg_doc
  FROM   json_documents
)
SELECT jt.*
FROM   dg_t,
       json_table(dg_doc, '$[*]'
         COLUMNS
           jpath   VARCHAR2(40) PATH '$."o:path"',
           type    VARCHAR2(10) PATH '$."type"',
           tlength NUMBER       PATH '$."o:length"') jt
ORDER BY jt.jpath;

JPATH                                  TYPE          TLENGTH
-------------------------------------- ---------- ----------
$                                      object            512
$.Active                               boolean             8
$.Address                              object            128
$.Address.City                         string              8
$.Address.Country                      string              2
$.Address.Postcode                     string              8
$.Address.Street                       string             16
$.ContactDetails                       object            128
$.ContactDetails.Email                 string             32
$.ContactDetails.Phone                 string             16
$.ContactDetails.Twitter               string              8

JPATH                                  TYPE          TLENGTH
-------------------------------------- ---------- ----------
$.DateOfBirth                          string             16
```

```
$.FirstName                              string              8
$.Job                                    string              8
$.LastName                               string              4

15 rows selected.
```

Uso de columnas virtuales creadas a partir de los metadatos extraídos de los documentos JSON por json_dataguide.

```
SQL> BEGIN
  DBMS_JSON.add_virtual_columns(
    tablename  => 'json_documents',
    jcolname   => 'data',
    dataguide  => DBMS_JSON.get_index_dataguide(
                    'json_documents',
                    'data',
                    DBMS_JSON.format_hierarchical));
END;
/

PL/SQL procedure successfully completed.

--observar las nuevas columnas virtuales por cada atributo

SQL> DESC json_documents

ID
NOT NULL RAW(16)
 DATA
CLOB
 DATA$Job
VARCHAR2(8)
 DATA$Active
VARCHAR2(8)
 DATA$City
VARCHAR2(8)
 DATA$Street
VARCHAR2(16)
 DATA$Country
VARCHAR2(2)
 DATA$Postcode
VARCHAR2(8)
 DATA$LastName
VARCHAR2(4)
 DATA$FirstName
VARCHAR2(8)
 DATA$DateOfBirth
VARCHAR2(16)
 DATA$Email
VARCHAR2(32)
 DATA$Phone
VARCHAR2(16)
```

```
 DATA$Twitter
VARCHAR2(8)

--eliminar las columnas virtuales de la tabla

SQL> BEGIN
  DBMS_JSON.drop_virtual_columns(
    tablename  => 'json_documents',
    jcolname   => 'data');
END;
/

PL/SQL procedure successfully completed.
```