ALUMNO 1: OMAR ELHANI BOTKALA

ORACLE

1. Muestra los espacios de tablas existentes en tu base de datos y la ruta de los ficheros que los componen. ¿Están las extensiones gestionadas localmente o por diccionario?

```
select tablespace_name, file_name from dba_data_files;
```

```
SQL> select tablespace_name, file_name from dba_data_files;

TABLESPACE_NAME
------------------------------
FILE_NAME
--------------------------------------------------------------------------------
USERS
/opt/oracle/oradata/ORCLCDB/users01.dbf

UNDOTBS1
/opt/oracle/oradata/ORCLCDB/undotbs01.dbf

SYSTEM
/opt/oracle/oradata/ORCLCDB/system01.dbf


TABLESPACE_NAME
------------------------------
FILE_NAME
--------------------------------------------------------------------------------
SYSAUX
/opt/oracle/oradata/ORCLCDB/sysaux01.dbf


SQL>
```

Los tablespaces actuales están gestionados localmente.

2. Usa la vista del diccionario de datos v$datafile para mirar cuando fue la última vez que se ejecutó el proceso CKPT en tu base de datos.

```
select distinct checkpoint_change#, to_char(checkpoint_time,'yy/mm/dd HH24:MI')
from v$datafile where checkpoint_time=(select max(checkpoint_time) from
v$datafile);
```

```
SQL> select distinct checkpoint_change#, to_char(checkpoint_time,'yy/mm/dd HH24:MI') from v$datafile where checkpoint_time=(select max(checkpoint_time
) from v$datafile);

CHECKPOINT_CHANGE# TO_CHAR(CHECKP
------------------ -------------
           4813961 22/02/01 03:32

SQL>
```

3. Intenta crear el tablespace TS1 con un fichero de 2M en tu disco que crezca automáticamente cuando sea necesario. ¿Puedes hacer que la gestión de extensiones sea por diccionario? Averigua la razón.

```
SQL> create tablespace TS1 datafile 'ts1.dbf' size 2M autoextend on;

Tablespace creado.

SQL>
```

El tablespace por defecto está hecho en local y no por diccionario por lo que no podemos modificar las cláusulas de almacenamiento.

4. Averigua el tamaño de un bloque de datos en tu base de datos. Cámbialo al doble del valor que tenga.

```
select distinct bytes/blocks from user_segments;
```

```
SQL> select distinct bytes/blocks from user_segments;

BYTES/BLOCKS
------------
        8192

SQL>
```

5. Realiza un procedimiento MostrarObjetosdeUsuarioenTS que reciba el nombre de un tablespace y el de un usuario y muestre qué objetos tiene el usuario en dicho tablespace y qué tamaño tiene cada uno de ellos.

```
create or replace procedure MostrarObjetosdeUsuarioenTS (p_tablespace varchar2,
p_user varchar2)
is
cursor c_objetos_usuarios
is
select segment_name object, bytes
from dba_segments
where tablespace_name=p_tablespace
and owner=p_user;
begin
dbms_output.put_line('objeto de usuario'||p_user||'tamanio (bytes)');
for v in c_objetos_usuarios loop
dbms_output.put_line(v.object||chr(9)||v.bytes);
end loop;
end;
/
```

6. Realiza un procedimiento llamado MostrarUsrsCuotaIlimitada que muestre los usuarios que puedan escribir de forma ilimitada en más de uno de los tablespaces que cuentan con ficheros en la unidad C:

```
create or replace procedure MostrarUsrsCuotaIlimitada
is
cursor c_UsuarioSinCuota
is
select username nombre from DBA_TS_QUOTAS
where max_blocks = -1 and max_bytes = -1
and tablespace_name in (select tablespace_name
from DBA_DATA_FILES
where substr(file_name,1,12)='/home/oracle')
group by username having count(tablespace_name) > 1
UNION
select grantee nombre from DBA_SYS_PRIVS
where privilege = 'UNLIMITED TABLESPACE';
begin
dbms_output.put_line('Usuarios cuota ilimitada: ');
for v in c_UsuarioSinCuota loop
dbms_output.put_line(v.nombre);
end loop;
end;
/
```

POSTGRESQL

7. Averigua si existe el concepto de tablespace en Postgres, en qué consiste y las diferencias con los tablespaces de ORACLE.

En **Postgresql** un **tablespace** representa un directorio de almacenamiento de archivos de datos, en estos archivos, se representan objetos de base de datos, como tablas o índices. Cuando accedemos a una tabla, el sistema localiza al archivo de datos correspondiente a través de su espacio de tabla. El **tablespaces** en **PostgreSQL** es diferente de otros sistemas de bases datos, está más inclinado a un concepto físico.

En **PostgreSQL** a diferencia de **Oracle** si no especificas un **tablespace** en la creación de tablas se usará el valor del parámetro **default_tablespace**. Si **default_tablespace** no está establecido o tiene un valor no válido, la tabla se creará en el mismo **tablespace** que la base de datos. Otra diferencia es que no se usan tablespaces donde se definan donde almacenar la base de datos, el esquema , los índices, etc.

En **Oracle** se usan ficheros (datafiles) y en **PostgreSQL** usamos directorios.
Mientras que **Oracle** permite una definición más precisa de estos tablespaces, **PostgreSQL** se queda un poco corto ya que ni siquiera podemos definir un tamaño máximo.
PostgreSQL no tiene cláusulas de almacenamiento, por lo que no podemos asignar cuotas de almacenamiento.

MySQL

8. Averigua si pueden establecerse clausulas de almacenamiento para las tablas o los espacios de tablas en MySQL.

No cuenta con la posibilidad de crear **tablespaces** en circunstancias normales. Para poder disponer de esta funcionalidad, deberíamos de tener instalada la versión de alta disponibilidad en cluster de **MySQL**, es decir la conocida **MySQL NDB**.

MONGODB

9. Averigua si existe el concepto de índice en MongoDB y las diferencias con los índices de ORACLE. Explica los distintos tipos de índice que ofrece MongoDB.

Los índices en **MongoDB** se generan en forma **Árbol-B** o **B-Tree**, es decir que los datos se guardan en forma de árbol, pero manteniendo los nodos balanceados. Esto incrementa la velocidad a la hora de buscar y también a la hora de devolver resultados ya ordenados.
MongoDB es capaz de recorrer los índices en ambos sentidos, por lo que con un solo índice, podemos conseguir ordenación tanto ascendente como descendente.

Tipos de índices en **MongoDB**:

Índices simples o de un solo campo: Estos índices se aplican a **un solo campo** de nuestro colección.
Índices compuestos: El índice se generará sobre **varios campos**. Podemos usarlos para consultar uno o varios campos, sin que sea necesario incluirlos todos.
Índices únicos: Los índice simples o múltiples, pueden estar obligados a contener valores únicos.
Índices sparse: Para crear índices que solo incluyan los documentos cuyo campo indexado existe, utilizaremos la opción **sparse**.

Otros tipos de índices:

Intersección de índices: De esta manera MongoDB puede u**sar varios índices** en una consulta, para así mejorar el rendimiento de la misma.
Indexación de subdocumentos: Solo nos ayudaría en el caso de que se realizaran consultas sobre el subdocumento completo y con los campos en el mismo orden.
Indexación de arrays: Indexar arrays también es algo posible con **MongoDB** aunque es algo que deberemos hacer con cuidado y teniendo en cuenta algunas limitaciones.
Consultas totalmente cubiertas por los índices: Aunque tengamos varios índices creados, no todas las consultas van a ser igual de eficientes. Las consultas más rápidas serán las denominadas consultas totalmente cubiertas. Son aquellas consultas cuyos campos consultados y devueltos están incluidas en el índice.

Alumna 2: LARA PRUNA TERNERO

ORACLE

1. Establece que los objetos que se creen en el TS1 (creado por Alumno 1) tengan un tamaño inicial de 200K, y que cada extensión sea del doble del tamaño que la anterior. El número máximo de extensiones debe ser de 3.

Una vez añadido el tablespace **TS1,** lo ponemos fuera de línea para poder realizar operaciones con él:

```
ALTER TABLESPACE TS1 OFFLINE;
```

Le modificamos para que el tamaño inicial tenga **200K** y el siguiente **400K**. Para que la siguiente sea el doble y así sucesivamente, en el parámetro **PCTINCREASE** especificaremos que las siguientes extensiones aumenten en un **100%** de las anteriores (**400K** más el **100%** del mismo serían **800K**):

```
ALTER TABLESPACE TS1
DEFAULT STORAGE (
INITIAL 200K
NEXT 400K
PCTINCREASE 100
MAXEXTENTS 3);
```

Aunque en teoría sería así, al tratar de modificar el espacio de tabla nos aparece un error:

```
ALTER TABLESPACE TS1
  2  DEFAULT STORAGE (
  3  INITIAL 200K
  4  NEXT 400K
  5  PCTINCREASE 100
  6  MAXEXTENTS 3);
ALTER TABLESPACE TS1
*
ERROR en linea 1:
ORA-25143: la clausula de almacenamiento por defecto no es compatible con la
politica de asignacion
```

Esto se debe a que, a la hora de instalar la base de datos, elegimos gestionar los tablespaces localmente en lugar de por diccionario, lo cual implica que no podamos usar las cláusulas de almacenamiento de los espacios de tablas. Podemos confirmar que es así consultando el tipo de gestión de las extensiones del tablespace **SYSTEM**, en la tabla **DBA_TABLESPACES**:

```
SQL> SELECT EXTENT_MANAGEMENT FROM DBA_TABLESPACES WHERE TABLESPACE_NAME='SYSTEM';

EXTENT_MAN
----------
LOCAL
```

Por desgracia, tampoco podríamos cambiar la gestión para que fuera por diccionario, puesto que se decide en la propia instalación.

2. Crea dos tablas en el tablespace recién creado e inserta un registro en cada una de ellas. Comprueba el espacio libre existente en el tablespace. Borra una de las tablas y comprueba si ha aumentado el espacio disponible en el tablespace. Explica la razón.

Lo primero que haremos es volver a poner en línea el tablespace **TS1**:

```
ALTER TABLESPACE TS1 ONLINE;
```

Antes de crear las tablas, comprobamos que el tablespace cuenta con 1MB de espacio libre:

```
SQL> SELECT BYTES FROM DBA_FREE_SPACE WHERE TABLESPACE_NAME='TS1';

     BYTES
----------
   1048576
```

A continuación, creamos las tablas y añadimos un registro a cada una:

```
CREATE TABLE T1 (
     CAMPO VARCHAR2(10)
) TABLESPACE TS1;

CREATE TABLE T2 (
     CAMPO VARCHAR2(10)
) TABLESPACE TS1;

INSERT INTO T1 VALUES ('REGISTRO1');
INSERT INTO T2 VALUES ('REGISTRO2');
```

Al volver a comprobar el espacio libre del tablespace, observamos que ha bajado ligeramente de **1MB** a **896KB**:

```
SQL> SELECT BYTES FROM DBA_FREE_SPACE WHERE TABLESPACE_NAME='TS1';

     BYTES
----------
    917504
```

Ahora vamos a borrar una de las tablas:

```
DROP TABLE T2;
```

Ahora, si echamos otro vistazo al espacio libre del tablespace, veremos que hay un nuevo registro en lugar de cambiar el que ya existía:

```
SQL> SELECT BYTES FROM DBA_FREE_SPACE WHERE TABLESPACE_NAME='TS1';

     BYTES
----------
    917504

SQL> DROP TABLE T2;

Tabla borrada.

SQL> SELECT BYTES FROM DBA_FREE_SPACE WHERE TABLESPACE_NAME='TS1';

     BYTES
----------
    917504
     65536
```

La razón de ello es que, al haber borrado la tabla, se han liberado las extensiones del segmento correspondiente a dicha tabla, que ocupan **65536 bytes**.

3. Convierte a TS1 en un tablespace de sólo lectura. Intenta insertar registros en la tabla existente. ¿Qué ocurre? Intenta ahora borrar la tabla. ¿Qué ocurre? ¿Por qué crees que pasa eso?

Convertimos el tablespace TS1 en solo lectura de la siguiente manera:

```
ALTER TABLESPACE TS1 READ ONLY;
```

Si intentamos insertar registros en la tabla T1 que hemos creado antes, lógicamente no se nos permitirá, porque significaría realizar un proceso de escritura en el tablespace que ahora es de solo lectura:

```
SQL> ALTER TABLESPACE TS1 READ ONLY;

Tablespace modificado.

SQL> INSERT INTO T1 VALUES ('REGISTRO2');
INSERT INTO T1 VALUES ('REGISTRO2')
            *
ERROR en linea 1:
ORA-00372: el archivo 19 no se puede modificar en este momento
ORA-01110: archivo de datos 19: '/opt/oracle/product/19c/dbhome_1/dbs/ts1.dbf'
```

Sin embargo, a la hora de borrar la tabla no nos encontramos ningún tipo de impedimento:

```
SQL> DROP TABLE T1;

Tabla borrada.
```

El motivo de que podamos **borrar (y crear)** tablas en un **tablespace de solo lectura** es que, para hacerlo, no se necesita escribir nada en los tablespaces donde se encuentran los segmentos de las tablas. En realidad no se ha borrado la tabla, lo único que hace la operación **DROP** es actualizar el diccionario de datos y colocar la tabla en la papelera de reciclaje (**recycle bin**) de **Oracle**, por si en un futuro queremos volver a crearla y añadirla al tablespace.

4. Crea un espacio de tablas TS2 con dos ficheros en rutas diferentes de 1M cada uno no autoextensibles. Crea en el citado tablespace una tabla con la cláusula de almacenamiento que quieras. Inserta registros hasta que se llene el tablespace. ¿Qué ocurre?

Creamos el **espacio de tablas** con los dos ficheros **no autoextensibles:**

```
CREATE TABLESPACE TS2
DATAFILE 'ts2_1.dbf' SIZE 1M,
'ts2_2.dbf' SIZE 1M
AUTOEXTEND OFF;
```

A continuación, **creamos una tabla** con una cláusula de almacenamiento:

```
CREATE TABLE T3
(CAMPO VARCHAR2(2))
STORAGE
(INITIAL 20K)
TABLESPACE TS2;
```

Para no tener que **insertar los registros** de uno en uno, he creado un procedimiento que inserta 10000 cada vez:

```
CREATE OR REPLACE PROCEDURE Insertar10000RegistrosT3
IS BEGIN
    FOR i IN 1..10000 LOOP
        INSERT INTO T3 VALUES ('1','2','3','4');
    END LOOP;
END;
/
```

Finalmente, lo ejecutamos hasta que el tablespace se llene:

```
SQL> EXEC INSERTAR10000REGISTROST3;

Procedimiento PL/SQL terminado correctamente.

SQL> EXEC INSERTAR10000REGISTROST3;
BEGIN INSERTAR10000REGISTROST3; END;

*
ERROR en linea 1:
ORA-01653: no se ha podido ampliar la tabla SYS.T3 con 128 en el tablespace TS2
ORA-06512: en "SYS.INSERTAR10000REGISTROST3", linea 4
ORA-06512: en linea 1


SQL> SELECT COUNT(*) FROM T3;

  COUNT(*)
----------
     60075
```

El sistema nos avisa de que no podemos insertar más registros, dado que el **tablespaces** se ha llenado por **completo**, y nosotros lo hemos configurado para que no se pueda ampliar automáticamente.

5. Hacer un procedimiento llamado *MostrarUsuariosporTablespace* que muestre por pantalla un listado de los tablespaces existentes con la lista de usuarios que tienen asignado cada uno de ellos por defecto y el número de los mismos, así:

```
Tablespace xxxx:

   Usr1
   Usr2
   ...

Total Usuarios Tablespace xxxx: n1

Tablespace yyyy:
```

```
   Usr1
   Usr2
   ...

Total Usuarios Tablespace yyyy: n2
....
Total Usuarios BD: nn
```

No olvides incluir los tablespaces temporales y de undo.

```
-- Función que acepta como parámetro el nombre de un tablespace --
-- y devuelve el número de usuarios que lo tienen por defecto. --
CREATE OR REPLACE FUNCTION ContarUsuariosDelTablespace(
    p_tablespace VARCHAR2
) RETURN NUMBER
IS
    v_usuarios NUMBER;
BEGIN
    SELECT COUNT(USERNAME) INTO v_usuarios
    FROM DBA_USERS
    WHERE DEFAULT_TABLESPACE = p_tablespace;
    RETURN v_usuarios;
END;
/


-- Procedimiento en el que se inserta un tablespace como parámetro
--
-- y muestra el nombre de cada usuario que lo tiene por defecto. --
CREATE OR REPLACE PROCEDURE MostrarUsuariosDelTablespace(
    p_tablespace VARCHAR2
) IS
    CURSOR c_usuarios IS SELECT USERNAME
    FROM DBA_USERS
    WHERE DEFAULT_TABLESPACE = p_tablespace;
    v_reg c_usuarios%ROWTYPE;
BEGIN
```

```
    FOR v_reg IN c_usuarios LOOP
        DBMS_OUTPUT.PUT_LINE(CHR(9)||v_reg.USERNAME);
    END LOOP;
END;
/
-- Procedimiento final donde se aplican todos los anteriores. --
CREATE OR REPLACE PROCEDURE MostrarUsuariosPorTablespace
IS
    CURSOR c_tablespaces IS SELECT TABLESPACE_NAME
    FROM DBA_TABLESPACES;
    v_reg c_tablespaces%ROWTYPE;
    v_acum NUMBER:=0;
    v_usuarios NUMBER:=0;
BEGIN
    FOR v_reg IN c_tablespaces LOOP

v_usuarios:=ContarUsuariosDelTablespace(v_reg.TABLESPACE_NAME);
        DBMS_OUTPUT.PUT_LINE('Tablespace '||
v_reg.TABLESPACE_NAME||':');
        DBMS_OUTPUT.PUT_LINE(CHR(9));
        MostrarUsuariosDelTablespace(v_reg.TABLESPACE_NAME);
        DBMS_OUTPUT.PUT_LINE('Total Usuarios Tablespace '||
v_reg.TABLESPACE_NAME||': '||v_usuarios);
DBMS_OUTPUT.PUT_LINE('-----------------------------------------------
');
        v_acum:=v_acum+v_usuarios;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Total Usuarios BD: '||v_acum);
    DBMS_OUTPUT.PUT_LINE(CHR(9));
END;
/
```

A continuación mostramos la prueba de funcionamiento. En la lista de tablespaces no solo se tienen en cuenta los permanentes, sino también los temporales (**TEMP**) y los de **UNDO** (**UNDOTBS1**):

```
SQL> EXEC MOSTRARUSUARIOSPORTABLESPACE;
Tablespace SYSTEM:

        SYS
        SYSTEM
        XS$NULL
        OJVMSYS
        LBACSYS
        OUTLN
        SYS$UMF
        C##PEPE
Total Usuarios Tablespace SYSTEM: 8
-----------------------------------------------
Tablespace SYSAUX:

        DBSNMP
        APPQOSSYS
        DBSFWUSER
        GGSYS
        ANONYMOUS
        CTXSYS
        DVSYS
        DVF
        GSMADMIN_INTERNAL
        MDSYS
        OLAPSYS
        XDB
        WMSYS
Total Usuarios Tablespace SYSAUX: 13
-----------------------------------------------
Tablespace UNDOTBS1:

Total Usuarios Tablespace UNDOTBS1: 0
```

```
Tablespace TEMP:

Total Usuarios Tablespace TEMP: 0
-----------------------------------------------
Tablespace USERS:

        GSMCATUSER
        C##LPRUNA
        C##LARA
        C##CLARA
        MDDATA
        SYSBACKUP
        REMOTE_SCHEDULER_AGENT
        C##BECARIO
        GSMUSER
        SYSRAC
        GSMROOTUSER
        C##LARA1
        SI_INFORMTN_SCHEMA
        C##MANOLO
        C##AYUDANTE
        AUDSYS
        DIP
        ORDPLUGINS
        SYSKM
        ORDDATA
        ORACLE_OCM
        C##SCOTT
        SYSDG
        ORDSYS
        C##JUAN
Total Usuarios Tablespace USERS: 25
-----------------------------------------------
Tablespace TBS_01:

Total Usuarios Tablespace TBS_01: 0
-----------------------------------------------
```

```
Tablespace TS2:

        C##PRUEBA
        C##USUARIOLARA
Total Usuarios Tablespace TS2: 2
-----------------------------------------------
Tablespace TS_PRODUCCION:

        C##LIDIA
        C##JAIME
Total Usuarios Tablespace TS_PRODUCCION: 2
-----------------------------------------------
Tablespace TS_VENTAS:

        C##ANA
        C##EVA
Total Usuarios Tablespace TS_VENTAS: 2
-----------------------------------------------
Tablespace TS1:

Total Usuarios Tablespace TS1: 0
-----------------------------------------------
Total Usuarios BD: 52


Procedimiento PL/SQL terminado correctamente.
```

6. Realiza un procedimiento llamado *MostrarDetallesIndices* que reciba el nombre de una tabla y muestre los detalles sobre los índices que hay definidos sobre las columnas de la misma.

```
CREATE OR REPLACE PROCEDURE MostrarColumnasDelIndice(
    p_indice VARCHAR2
) IS
    CURSOR c_columnas IS SELECT DISTINCT COLUMN_NAME,DESCEND
            FROM DBA_IND_COLUMNS WHERE INDEX_NAME=p_indice;
    v_reg c_columnas%ROWTYPE;
BEGIN
    FOR v_reg IN c_columnas LOOP
        DBMS_OUTPUT.PUT_LINE(CHR(9)||v_reg.COLUMN_NAME||'('||
v_reg.DESCEND||')');
    END LOOP;
END;
/


CREATE OR REPLACE PROCEDURE MostrarDetallesIndices(
    p_tabla VARCHAR2,
    p_propietario VARCHAR2
) IS
    CURSOR c_indices IS SELECT DISTINCT
INDEX_NAME,TABLESPACE_NAME,NUM_ROWS,INI_TRANS,MAX_TRANS,STATUS
    FROM DBA_INDEXES WHERE TABLE_NAME=UPPER(p_tabla) AND
TABLE_OWNER=UPPER(p_propietario);
    v_reg c_indices%ROWTYPE;
BEGIN
    FOR v_reg IN c_indices LOOP
        DBMS_OUTPUT.PUT_LINE('Indice: '||v_reg.INDEX_NAME);
        DBMS_OUTPUT.PUT_LINE('Columnas:');
        MostrarColumnasDelIndice(v_reg.INDEX_NAME);
        DBMS_OUTPUT.PUT_LINE('Espacio de tabla usado: '||
v_reg.TABLESPACE_NAME);
        DBMS_OUTPUT.PUT_LINE('Numero de filas: '||v_reg.NUM_ROWS);
        DBMS_OUTPUT.PUT_LINE('Numero inicial de transferencias: '||
v_reg.INI_TRANS);
```

```
        DBMS_OUTPUT.PUT_LINE('Numero maximo de transferencias: '||
v_reg.MAX_TRANS);
        DBMS_OUTPUT.PUT_LINE('Estado: '||v_reg.STATUS);
        DBMS_OUTPUT.PUT_LINE('-----------------------------------');
    END LOOP;
END;
/
```

```
SQL> EXEC MostrarDetallesIndices('T3','SYS');
Indice: I_CAMPOS1_2
Columnas:
        SYS_NC00005$(DESC)
        CAMPO2(ASC)
Espacio de tabla usado: SYSTEM
Numero de filas: 60075
Numero inicial de transferencias: 2
Numero maximo de transferencias: 255
Estado: VALID
-----------------------------------

Procedimiento PL/SQL terminado correctamente.

SQL> EXEC MostrarDetallesIndices('EMP','C##SCOTT');
Indice: I_ENAME
Columnas:
        ENAME(ASC)
Espacio de tabla usado: SYSTEM
Numero de filas: 13
Numero inicial de transferencias: 2
Numero maximo de transferencias: 255
Estado: VALID
-----------------------------------
Indice: PK_EMP
Columnas:
        CIF(ASC)
        EMPNO(ASC)
Espacio de tabla usado: USERS
Numero de filas: 13
Numero inicial de transferencias: 2
Numero maximo de transferencias: 255
Estado: VALID
-----------------------------------

Procedimiento PL/SQL terminado correctamente.
```

POSTGRESQL

7. Averigua si existe el concepto de segmento y el de extensión en Postgres, en qué consiste y las diferencias con los conceptos correspondientes de ORACLE.

Los conceptos de **segmento** y extensión en **PostgreSQL** son **diferentes** a los de **Oracle**, puesto que, en lugar de espacios de tablas, se emplea el sistema de ficheros del propio sistema operativo como almacenamiento para la base de datos. De hecho, un espacio de tabla en **PostgreSQL** es básicamente un directorio en el que se almacenan ficheros de datos. Cada objeto tiene su propio fichero de datos, llamado **segmento** y con **1GB** de **espacio límite**. Si el objeto es más grande, tendrá varios ficheros (lo que en Oracle equivaldría a una **extensión**).
En la siguiente tabla, veremos más clara la diferencia entre Oracle y PostgreSQL:

Oracle	PostgreSQL
Tablespace	Sistema de Ficheros
Fichero de datos	Volumen Lógico/Físico
Segmento	Conjunto de ficheros de un objeto
Extensión	Segmento / fichero de datos

MySQL

8. Averigua si existe el concepto de espacio de tablas en MySQL y las diferencias con los tablespaces de ORACLE.

El concepto del espacio de tablas en **MySQL** es muy distinto del de **Oracle**. Cuando creamos una tabla usando el motor **InnoDB**, los datos de esa tabla se almacenan en un fichero de datos llamado **espacio de tabla**, que contiene tanto datos como índices. Con el parámetro **“innodb_file_per_table”** habilitado, **InnoDB** usará un único fichero por tabla, que tendrá la extensión **fichero.ibd** .De lo contrario, almacenará todas las tablas en el **espacio de tablas del sistema**, que también es un único fichero (por defecto, es **“ibdata1”**, localizado en el directorio de datos de **MySQL**). Este único fichero tiene un espacio límite de **12MB**, pero se puede aumentar su tamaño si llega a llenarse.

A partir de la **versión 5.7 de MySQL**, **InnoDB** soporta los llamamos espacios de tablas generales, que se crean con la directiva **“CREATE TABLESPACE”.** Sin embargo, **MariaDB Server** no soporta estos espacios de tablas ni reconoce el comando mencionado.

MongoDB

9. Averigua si existe la posibilidad en MongoDB de decidir en qué archivo se almacena una colección.

Siempre y cuando el motor de almacenamiento que se esté empleando sea **WiredTiger**, en el fichero de configuración de **MongoDB** (**/etc/mongod.conf en Debian**), se puede indicar mediante el parámetro **storage.dbPath** el directorio donde se guardarán todos los ficheros de

datos, incluidas las colecciones. Por defecto, el directorio es **/var/lib/mongodb**.

En lugar de editar el fichero de configuración, también se puede cambiar el directorio de almacenamiento desde línea de comando:

```
mongod --storageEngine wiredTiger --dbpath <ruta>
```

Si el motor de almacenamiento elegido fuera **"In-Memory"**, en dicho directorio se guardarían **ficheros** pequeños de **metadatos y de diagnóstico**, así como archivos temporales para generar índices de gran tamaño.

Por desgracia, no parece que sea posible guardar diferentes colecciones en diferentes directorios, puesto que las únicas opciones que ofrece **MongoDB** es guardarlo todo en un directorio (por defecto o a elección del usuario), o bien en memoria con **In-Memory** (solo disponible en **MongoDB Enterprise**).

ALUMNO 3: MIGUEL CÓRDOBA ARANDA

ORACLE

1. Muestra los objetos a los que pertenecen las extensiones del tablespace TS2 (creado por Alumno 2) y el tamaño de cada una de ellas.

Para mostrar los objetos del tablespace TS2 he realizado la siguiente consulta:

select segment_name,segment_type,blocks,bytes from dba_extents where tablespace_name='ts2';

```
SQL> SELECT SEGMENT_NAME,SEGMENT_TYPE,BYTES,BLOCKS,BYTES FROM DBA_EXTENTS WHERE TABLESPACE_NAME='TS2';
SEGMENT_NAME
--------------------------------------------------------------------------------
SEGMENT_TYPE           BLOCKS      BYTES
------------------ ---------- ----------
T3
TABLE                       8      65536

T3
TABLE                       8      65536

T3
TABLE                       8      65536
```

2. Borra la tabla que está llenando TS2 consiguiendo que vuelvan a existir extensiones libres. Añade después otro fichero de datos a TS2.

Primero he tenido que consultar que tablas pertenecen a TS2 con la siguiente consulta:

select table_name from dba_tables where tablespace_name='TS2';

Al solo haber una sola tabla la he borrado:

drop table T3;

```
SQL> select table_name from dba_tables where tablespace_name='TS2';

TABLE_NAME
------------------------------------------------------------------------
T3

SQL> drop table T3;

Tabla borrada.
```

Para añadir un fichero nuevo he usado el siguiente comando

```
ALTER TABLESPACE TS2
  ADD DATAFILE '/home/oracle/Tablespace/ts2-1.dbf' SIZE 100M
  AUTOEXTEND ON NEXT 1000K MAXSIZE UNLIMITED;
```

```
SQL> ALTER TABLESPACE TS2
  ADD DATAFILE '/home/oracle/Tablespace/ts2-1.dbf' SIZE 100M
  AUTOEXTEND ON NEXT 1000K MAXSIZE UNLIMITED;  2    3

Tablespace modificado.
[oracle@miguelito ~]$ cd Tablespace/
[oracle@miguelito Tablespace]$ ls
ts2-1.dbf  ts3.dbf
```

3. Crea el tablespace TS3 gestionado localmente con un tamaño de extension uniforme de 128K y un fichero de datos asociado. Cambia la ubicación del fichero de datos y modifica la base de datos para que pueda acceder al mismo. Crea en TS3 dos tablas e inserta registros en las mismas. Comprueba que segmentos tiene TS3, qué extensiones tiene cada uno de ellos y en qué ficheros se encuentran.

Para crear el **tablespaces TS3** he ejecutado el siguiente comando:

```
CREATE TABLESPACE TS3 DATAFILE '/home/oracle/ts3.dbf' SIZE 10M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;
```

```
SQL> CREATE TABLESPACE TS3 DATAFILE '/home/oracle/ts3.dbf' SIZE 10M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;  2

Tablespace creado.
```

Para cambiar el tablespace **TS3** de sitio con la base de datos arrancada he tenido que seguir una serie de pasos:

1. Se **desactivar** el tablespaces:

```
ALTER TABLESPACE TS3 OFFLINE;
```

2. Se **cambia la ubicación** del fichero en el **sistema**.

Para este paso he creado un directorio llamado Tablespace y he introducido el fichero dentro para posteriormente cambiarle la ubicación en Oracle.

```
#mv ts3.dbf Tablespace
```

3. Ahora **cambiaremos la ubicación** del fichero en **Oracle.**

```
ALTER TABLESPACE TS3 RENAME DATAFILE '/home/oracle/ts3.dbf' TO
    '/home/oracle/Tablespace/ts3.dbf';
```

4. Una vez cambiada la ubicación volveremos a **activar el tablespace.**

```
ALTER TABLESPACE TS3 ONLINE;
```

```
SQL> ALTER TABLESPACE TS3 OFFLINE;

Tablespace modificado.

SQL> ALTER TABLESPACE TS3 RENAME DATAFILE '/home/oracle/ts3.dbf' TO '/home/oracle/Tablespace/ts3.dbf';

Tablespace modificado.

SQL> ALTER TABLESPACE TS3 ONLINE;

Tablespace modificado.
```

El siguiente paso es crear dos tablas en **TS3**, las tablas que he creado son a partir de las tablas **EMP** y **DEPT** del usuario **SCOTT**.

Para ello he usado el comando

```
create table tabla1
TABLESPACE TS3
as
select *
from C##SCOTT.EMP;

create table tabla2
TABLESPACE TS3
as
```

```
SQL> create table tabla1
TABLESPACE TS3
as
select *
from C##SCOTT.EMP;

Tabla creada.

SQL> create table tabla2
TABLESPACE TS3
as
select *
from C##SCOTT.DEPT;

Tabla creada.
```

```
select *
from C##SCOTT.EMP;
```

Por último he comprobado los **segmentos** que tiene **TS3**, que **extensiones** tiene cada uno de ellos y en que **fichero** se encuentran.

```
select segment_type,segment_name,file_name
from dba_extents e,dba_data_files f
where e.tablespace_name=f.tablespace_name and
e.tablespace_name='TS3';
```

```
SQL> select segment_type,segment_name,file_name
from dba_extents e,dba_data_files f
where e.tablespace_name=f.tablespace_name and e.tablespace_name='TS3';

SEGMENT_TYPE
------------------
SEGMENT_NAME
--------------------------------------------------------------------------------
FILE_NAME
--------------------------------------------------------------------------------
TABLE
TABLA2
/home/oracle/Tablespace/ts3.dbf

TABLE
TABLA1
/home/oracle/Tablespace/ts3.dbf
```

4. Redimensiona los ficheros asociados a los tres tablespaces que has creado de forma que ocupen el mínimo espacio posible para alojar sus objetos.

5. Realiza un procedimiento llamado InformeRestricciones que reciba el nombre de una tabla y muestre los nombres de las restricciones que tiene, a qué columna o columnas afectan y en qué consisten exactamente.

```
create or replace procedure InformeRestricciones(p_nombretabla
varchar2)
as
cursor c_restricciones is
select
column_name,a1.constraint_name,constraint_type,search_condition
from all_constraints a1,all_cons_columns a2
where a1.table_name=a2.table_name and
a1.constraint_name=a2.constraint_name
and a1.table_name=p_nombretabla;
begin
```

```
for v_info in c_restricciones loop
dbms_output.put_line('NOMBRE RETRICCION: '||
v_info.constraint_name||'    COLUMNA: '||v_info.column_name||'
TIPO: '||v_info.constraint_type||' DESCRIPCION: '||
v_info.search_condition);
end loop;
end;
/
```

```
SQL> exec InformeRestricciones('EMP');
NOMBRE RETRICCION: PK_EMP    COLUMNA: EMPNO   TIPO: P DESCRIPCION:
NOMBRE RETRICCION: FK_DEPTNO    COLUMNA: DEPTNO   TIPO: R DESCRIPCION:
NOMBRE RETRICCION: SALARIO    COLUMNA: SAL   TIPO: C DESCRIPCION: sal>100

Procedimiento PL/SQL terminado correctamente.
```

6. Realiza un procedimiento llamado MostrarAlmacenamientoUsuario que reciba el nombre de un usuario y devuelva el espacio que ocupan sus objetos agrupando por dispositivos y archivos:

Usuario: *NombreUsuario*

Dispositivo:xxxx

Archivo: xxxxxxx.xxx

Tabla1.....nnn K
…
TablaN.....nnn K
Indice1.....nnn K
…
IndiceN.....nnn K

Total Espacio en Archivo xxxxxxx.xxx: nnnnn K

Archivo:...
…

Total Espacio en Dispositivo xxxx: nnnnnn K

Dispositivo: yyyy
…

Total Espacio Usuario en la BD: nnnnnnn K

POSTGRESQL

7. Averigua si es posible establecer cuotas de uso sobre los tablespaces en Postgres.

En **PostgreSQL no es posible** de establecer cuotas porque aunque se hayan implementado los tablespaces **no tienen cláusulas de almacenamiento**. Por lo que los **tablespaces** estarán en una **partición** o **disco**, cuando ocupe la totalidad de la partición hay que crear una nueva y si está en disco hay que insertar un disco nuevo.

Sintaxis para crear un **tablespace** en **PostgreSQL**:

```
CREATE TABLESPACE nombredeltablespace LOCATION '/ruta disco o /ruta partición';
```

MySQL

8. Averigua si existe el concepto de extensión en MySQL y si coincide con el existente en ORACLE.

El concepto de extensión si existe en **MySQL** pero a lo que se refiere es a un complemento (**PLUGIN**) para adquirir una funcionalidad según el servicio que requiera a **MySQL**.

Ejemplo:

Si instalas un **MySQL** y un **WordPress**, tienes que instalar una extensión de **MySQL** para **PHP** para que pueda funcionar con el **WordPress**.

MONGODB

9. Averigua si en MongoDB puede saberse el espacio disponible para almacenar nuevos documentos.

No es algo que normalmente se haga,o simplemente no se hace. Pero **MongoDB** proporciona una serie de funciones que puede usar para **comprobar el tamaño de la colección** y algunos servicios que demuestran cómo ejecutar las funciones en cada base de datos en el servidor.

Las **funciones** son ajustables para su mejor legibilidad enumerando el tamaño y haciendo un recuento de todas las colecciones en una base de datos.

ALUMNO 4: DANIEL MESA MEJÍAS

ORACLE

1. Crea un tablespace de undo e intenta crear una tabla en él.

```
CREATE UNDO TABLESPACE TUNDO DATAFILE 'tablespace_undo1.dbf'
SIZE 200M AUTOEXTEND ON;
```

```
SQL> CREATE UNDO TABLESPACE TUNDO DATAFILE 'tablespace_undo1.dbf'       SIZE   2
00M     AUTOEXTEND ON;

Tablespace creado.

SQL>
```

```
CREATE TABLE tabla_undo (id number(12)) TABLESPACE TUNDO;
```

```
SQL> CREATE table tabla_undo (id number(12)) tablespace TUNDO;
CREATE table tabla_undo (id number(12)) tablespace TUNDO
*
ERROR en linea 1:
ORA-30022: No se pueden crear segmentos en un tablespace de deshacer
```

No se pueden crear tablas en los tablespaces dedicados a rollback, ya que almacenan datos automáticamente cuando el usuario realiza una acción, para posteriormente poder hacer rollback si es necesario.

2. Crea un tablespace temporal TEMP2 y escribe una sentencia SQL que genere un script que haga usar TEMP2 a todos los usuarios que tienen USERS como tablespace por defecto.

```
CREATE TEMPORARY TABLESPACE TEMP2 TEMPFILE 'temp2.dbf' SIZE
200M;
```

```
SQL> CREATE TEMPORARY TABLESPACE TEMP2 TEMPFILE  'temp2.dbf' SIZE 200M;

Tablespace creado.

SQL>
```

```
SELECT 'ALTER USER ' ||USERNAME|| ' TEMPORARY TABLESPACE
TEMP2;'
 FROM DBA_USERS WHERE DEFAULT_TABLESPACE = 'USERS'
```

```
SQL> SELECT 'ALTER USER ' ||USERNAME|| ' TEMPORARY TABLESPACE TEMP2;'
FROM DBA_USERS WHERE DEFAULT_TABLESPACE = 'USERS';  2

'ALTERUSER'||USERNAME||'TEMPORARYTABLESPACETEMP2;'
--------------------------------------------------------------------------------
ALTER USER GSMCATUSER TEMPORARY TABLESPACE TEMP2;
ALTER USER MDDATA TEMPORARY TABLESPACE TEMP2;
ALTER USER C##BECARIO TEMPORARY TABLESPACE TEMP2;
ALTER USER REMOTE_SCHEDULER_AGENT TEMPORARY TABLESPACE TEMP2;
ALTER USER SYSBACKUP TEMPORARY TABLESPACE TEMP2;
ALTER USER USRPRACTICA1 TEMPORARY TABLESPACE TEMP2;
ALTER USER GSMUSER TEMPORARY TABLESPACE TEMP2;
ALTER USER SYSRAC TEMPORARY TABLESPACE TEMP2;
ALTER USER GSMROOTUSER TEMPORARY TABLESPACE TEMP2;
ALTER USER PREPASO TEMPORARY TABLESPACE TEMP2;
ALTER USER SI_INFORMTN_SCHEMA TEMPORARY TABLESPACE TEMP2;

'ALTERUSER'||USERNAME||'TEMPORARYTABLESPACETEMP2;'
--------------------------------------------------------------------------------
ALTER USER AUDSYS TEMPORARY TABLESPACE TEMP2;
ALTER USER DANIEL TEMPORARY TABLESPACE TEMP2;
ALTER USER ORDPLUGINS TEMPORARY TABLESPACE TEMP2;
ALTER USER DIP TEMPORARY TABLESPACE TEMP2;
ALTER USER DANIP TEMPORARY TABLESPACE TEMP2;
ALTER USER SYSKM TEMPORARY TABLESPACE TEMP2;
ALTER USER ORDDATA TEMPORARY TABLESPACE TEMP2;
ALTER USER TIENDA TEMPORARY TABLESPACE TEMP2;
ALTER USER ORACLE_OCM TEMPORARY TABLESPACE TEMP2;
ALTER USER SCOTT TEMPORARY TABLESPACE TEMP2;
ALTER USER ORDSYS TEMPORARY TABLESPACE TEMP2;

'ALTERUSER'||USERNAME||'TEMPORARYTABLESPACETEMP2;'
--------------------------------------------------------------------------------
ALTER USER SYSDG TEMPORARY TABLESPACE TEMP2;

23 filas seleccionadas.
```

3. Borra todos los tablespaces creados para esta práctica sin que quede rastro de ellos. Realiza las acciones previas que sean necesarias.

```
DROP TABLESPACE TUNDO INCLUDING CONTENTS AND DATAFILES

CASCADE CONSTRAINTS;

DROP TABLESPACE TEMP2 INCLUDING CONTENTS AND DATAFILES

CASCADE CONSTRAINTS;
```

```
SQL> DROP TABLESPACE TUNDO INCLUDING CONTENTS AND DATAFILES
ASCADE CONSTRAINTS;

Tablespace borrado.

SQL> DROP TABLESPACE TEMP2 INCLUDING CONTENTS AND DATAFILES
                        CASCADE CONSTRAINTS;

Tablespace borrado.

SQL>
```

Con esta intrucción eliminamos los ficheros de datos relacionados con el tablespace y las restricciones vinculadas, y por supuesto el tablespace correspondiente con su contenido.

4. Averigua los segmentos existentes para realizar un ROLLBACK y el tamaño de sus extensiones.

Para ver los segmentos existentes en relación a Rollback tendremos que consultar la tabla DBA_ROLLBACK_SEGS, en ella podremos listar los diferentes segmentos. En cuanto a el tamaño de sus extensiones, tendremos que recurrir a la tabla DBA_EXTENTS, en la cual se almacenan todos los EXTENTS con su respectivo segmento, así que lo que haremos será relacionar los segmentos que aparecen en la tabla DBA_Rollback_Segs con el nombre relacionado al EXTENT en la tabla DBA_EXTENTES, obteniendo así una lista de todos los EXTENTS con su respectivo tamaño y segmento.

La consulta sería la siguiente:

```
SELECT r.SEGMENT_NAME,e.EXTENT_ID, e.BYTES from
DBA_ROLLBACK_SEGS r, DBA_EXTENTS e WHERE
r.SEGMENT_NAME=e.SEGMENT_NAME;
```

```
SQL> SELECT r.SEGMENT_NAME,e.EXTENT_ID, e.BYTES from DBA_ROLLBACK_SEGS r, DBA_EXTENTS e WHERE r.SEGMENT_NAME=e.SEGMENT_NAME;

SEGMENT_NAME                   EXTENT_ID      BYTES
------------------------------ ---------- ----------
SYSTEM                                  0      65536
SYSTEM                                  1      65536
SYSTEM                                  2      65536
SYSTEM                                  3      65536
SYSTEM                                  4      65536
SYSTEM                                  5      65536
SYSTEM                                  6      65536
_SYSSMU1_1261223759$                    0      65536
_SYSSMU1_1261223759$                    1      65536
_SYSSMU1_1261223759$                    2    1048576
_SYSSMU1_1261223759$                    3    1048576

SEGMENT_NAME                   EXTENT_ID      BYTES
------------------------------ ---------- ----------
_SYSSMU1_1261223759$                    4    1048576
_SYSSMU2_27624015$                      0      65536
_SYSSMU2_27624015$                      1      65536
_SYSSMU2_27624015$                      2    1048576
_SYSSMU2_27624015$                      3    1048576
_SYSSMU3_2421748942$                    0      65536
_SYSSMU3_2421748942$                    1      65536
_SYSSMU3_2421748942$                    2    1048576
_SYSSMU3_2421748942$                    3    1048576
_SYSSMU3_2421748942$                    4    1048576
_SYSSMU3_2421748942$                    5    1048576

SEGMENT_NAME                   EXTENT_ID      BYTES
------------------------------ ---------- ----------
_SYSSMU4_625702278$                     0      65536
_SYSSMU4_625702278$                     1      65536
_SYSSMU4_625702278$                     2    1048576
_SYSSMU4_625702278$                     3      65536
_SYSSMU4_625702278$                     4    1048576
_SYSSMU5_2101348960$                    0      65536
_SYSSMU5_2101348960$                    1      65536
_SYSSMU5_2101348960$                    2    1048576
_SYSSMU5_2101348960$                    3    1048576
_SYSSMU5_2101348960$                    4    1048576
_SYSSMU5_2101348960$                    5    1048576

SEGMENT_NAME                   EXTENT_ID      BYTES
------------------------------ ---------- ----------
_SYSSMU6_813816332$                     0      65536
_SYSSMU6_813816332$                     1      65536
_SYSSMU6_813816332$                     2    1048576
_SYSSMU6_813816332$                     3    1048576
_SYSSMU6_813816332$                     4    1048576
```

5. Queremos cambiar de ubicación un tablespace, pero antes debemos avisar a los usuarios que tienen acceso de lectura o escritura a cualquiera de los objetos almacenados en el mismo. Escribe un procedimiento llamado *MostrarUsuariosAccesoTS* que obtenga un listado con los nombres de dichos usuarios .

Para conseguir este fin voy a crear 2 procedimientos, uno principal, que será en el que introduzcamos como parámetro de entrada el tablespace del que queremos obtener la información de los usuarios con permisos sobre sus objetos, y otro que será el encargado de recolectar a los diferentes usuarios.

El procedimiento “usuarios” se encarga de consultar a la tabla “dba_tab_privs” sobre los usuarios que tienen algún permiso de lectura o escritura (DROP,INSERT,SELECT o UPDATE) sobre la tabla introducida como parámetro de entrada, la cual obtendrá del siguiente procedimiento llamado “MostrarUsuariosAccesoTS”, que se encargará de almacenar los objetos que contiene el tablespace introducido como parámetro de entrada, creando así un cursor con el nombre de las tablas que posteriormente se utilizará como parámetro de entrada para llamar al procedimiento “usuarios”.

Aquí puedes ver el código de ambos procedimientos .

```
create or replace procedure usuarios(v_tabla varchar2)
is
cursor c_permisos is select distinct grantee from dba_tab_privs where
table_name=v_tabla and PRIVILEGE='INSERT' or PRIVILEGE='DROP' or
PRIVILEGE='UPDATE' or PRIVILEGE='SELECT' ;
begin
for i in c_permisos loop
dbms_output.put_line('Nombre -> '||i.grantee);
end loop;
end;
/
```

```
create or replace procedure MostrarUsuariosAccesoTS(v_tablespace varchar2)
is
cursor c_objetos is select table_name from dba_tables where
tablespace_name=v_tablespace;
v_objeto c_objetos%rowtype;
begin
open c_objetos;
fetch c_objetos into v_objeto;
dbms_output.put_line('--------------------');
dbms_output.put_line(' USUARIOS ');
dbms_output.put_line('--------------------');
while c_objetos%FOUND loop
usuarios(v_objeto.table_name);
fetch c_objetos into v_objeto;
end loop;
end;
/
```

```
SQL> exec MostrarUsuariosAccesoTS('SCOTTPRUEBA');
--------------------
USUARIOS
--------------------
Nombre -> SYS
Nombre -> APPQOSSYS
Nombre -> GSMCATUSER
Nombre -> SYSTEM
Nombre -> DBSNMP
Nombre -> DATAPATCH_ROLE
Nombre -> ORDADMIN
Nombre -> USRPRACTICA1
Nombre -> WM_ADMIN_ROLE
Nombre -> DBSFWUSER
Nombre -> GATHER_SYSTEM_STATISTICS
Nombre -> CDB_DBA
Nombre -> OLAP_XS_ADMIN
Nombre -> SYSBACKUP
Nombre -> ADM_PARALLEL_EXECUTE_TASK
Nombre -> OEM_MONITOR
Nombre -> GGSYS
Nombre -> CTXAPP
Nombre -> PUBLIC
Nombre -> SELECT_CATALOG_ROLE
Nombre -> AQ_ADMINISTRATOR_ROLE
Nombre -> CTXSYS
Nombre -> SYSRAC
Nombre -> EXP_FULL_DATABASE
Nombre -> APPLICATION_TRACE_VIEWER
Nombre -> EM_EXPRESS_BASIC
Nombre -> BDSQL_ADMIN
Nombre -> OLAP_USER
Nombre -> OLAP_DBA
Nombre -> EXECUTE_CATALOG_ROLE
Nombre -> DV_ADMIN
Nombre -> DV_POLICY_OWNER
Nombre -> AUDSYS
Nombre -> DVSYS
Nombre -> DV_MONITOR
Nombre -> DV_STREAMS_ADMIN
Nombre -> GSMADMIN_INTERNAL
Nombre -> DBA
Nombre -> IMP_FULL_DATABASE
Nombre -> DBFS_ROLE
Nombre -> XS_CACHE_ADMIN
Nombre -> CAPTURE_ADMIN
Nombre -> GSMADMIN_ROLE
Nombre -> DV_SECANALYST
Nombre -> LBACSYS
Nombre -> SYSUMF_ROLE
Nombre -> MDSYS
```

6.Realiza un procedimiento llamado *MostrarInfoTabla* que reciba el nombre de una tabla y muestre la siguiente información sobre la misma: propietario, usuarios que pueden leer sus datos, usuarios que pueden cambiar (insertar, modificar o eliminar) sus datos, usuarios que pueden modificar su estructura, usuarios que pueden eliminarla, lista de extensiones y en qué fichero de datos se encuentran.

Este procedimiento recopila los usuarios con permiso **Select** sobre la tabla introducida como parámetro de entrada.

```
create or replace procedure MostrarUsuariosLectura(v_tabla varchar2, v_propietario
varchar2)
is
cursor c_permisos is select distinct grantee from dba_tab_privs where
table_name=v_tabla and PRIVILEGE='SELECT' and OWNER=v_propietario;
begin
dbms_output.put_line('--------------------------');
dbms_output.put_line(' USUARIOS CON PERMISO LECTURA ');
dbms_output.put_line('--------------------------');
for i in c_permisos loop
dbms_output.put_line('Nombre -> '||i.grantee);
end loop;
end;
/
```

Este procedimiento recopila los usuarios con permiso **Insert, Drop y Update** sobre la tabla introducida como parámetro de entrada.

```
create or replace procedure MostrarUsuariosEscritura(v_tabla varchar2,v_propietario
varchar2)
is
cursor c_permisos is select distinct grantee from dba_tab_privs
where table_name=v_tabla and PRIVILEGE='INSERT' or PRIVILEGE='DROP' or
PRIVILEGE='UPDATE' and OWNER=v_propietario;
begin
dbms_output.put_line('--------------------------');
dbms_output.put_line(' USUARIOS CON PERMISO ESCRITURA ');
dbms_output.put_line('--------------------------');
for i in c_permisos loop
dbms_output.put_line('Nombre -> '||i.grantee);
end loop;
end;
/
```

Este procedimiento recopila los usuarios con permiso de alteración (**ALTER**) sobre la tabla introducida como parámetro de entrada.

```
create or replace procedure MostrarUsuariosAlter(v_tabla varchar2,v_propietario
varchar2)
is
cursor c_permisos is select distinct grantee from dba_tab_privs
where table_name=v_tabla and PRIVILEGE='ALTER' and OWNER=v_propietario;
cursor c_permisos_sys is select distinct grantee from DBA_SYS_PRIVS where
PRIVILEGE='ALTER ANY TABLE';
begin
dbms_output.put_line('--------------------------');
dbms_output.put_line(' USUARIOS CON PERMISO DE ALTERACION ');
dbms_output.put_line('--------------------------');
dbms_output.put_line('Nombre -> '||v_propietario);
for i in c_permisos loop
dbms_output.put_line('Nombre -> '||i.grantee);
end loop;
for i in c_permisos_sys loop
dbms_output.put_line('Nombre -> '||i.grantee);
end loop;
end;
/
```

Este procedimiento recopila los usuarios con permiso de eliminación (**DROP**) sobre la tabla introducida como parámetro de entrada.

```
create or replace procedure MostrarUsuariosDrop(v_tabla varchar2,v_propietario
varchar2)
is
cursor c_permisos_sys is select distinct grantee from DBA_SYS_PRIVS where
PRIVILEGE='DROP ANY TABLE';
begin
dbms_output.put_line('--------------------------');
dbms_output.put_line(' USUARIOS CON PERMISO DE DROP ');
dbms_output.put_line('--------------------------');
dbms_output.put_line('Nombre -> '||v_propietario);
for i in c_permisos_sys loop
if i.grantee = v_propietario then
dbms_output.put_line('');
else
dbms_output.put_line('Nombre -> '||i.grantee);
end if;
end loop;
end;
/
```

Este procedimiento recopila los **Extents, tamaño y fichero** al que pertenecen de la tabla introducida como parámetro de entrada.

```
create or replace procedure MostrarExtents(v_tabla varchar2,v_propietario varchar2)
is
cursor c_extents is select extent_id,bytes,file_id from DBA_EXTENTS where
segment_name=v_tabla and owner=v_propietario;
begin
dbms_output.put_line('--------------------------');
dbms_output.put_line(' EXTENTS ');
dbms_output.put_line('--------------------------');
for i in c_extents loop
dbms_output.put_line('ID -> '||i.extent_id||' SIZE -> '||i.bytes||' ID FICHERO -> '||
i.file_id);
end loop;
end;
/
```

Este procedimiento es el principal, llama a todos los anteriores para mostrar las diferentes informaciones solicitadas.

```
create or replace procedure MostrarInfoTabla(v_tabla varchar2,v_propietario
varchar2)
is
begin
MostrarUsuariosLectura(v_tabla, v_propietario);
MostrarUsuariosEscritura(v_tabla, v_propietario);
MostrarUsuariosAlter(v_tabla, v_propietario);
MostrarUsuariosDrop(v_tabla, v_propietario);
MostrarExtents(v_tabla, v_propietario);
end;
/
```

```
SQL> exec MostrarInfoTabla('EMP','SCOTT');
----------------------------
USUARIOS CON PERMISO LECTURA
----------------------------
Nombre -> USRPRACTICA1
----------------------------
USUARIOS CON PERMISO ESCRITURA
----------------------------
Nombre -> C##BECARIO
Nombre -> ROLPRACTICA1
Nombre -> USRPRACTICA1
----------------------------
USUARIOS CON PERMISO DE ALTERACION
----------------------------
Nombre -> SCOTT
Nombre -> USRPRACTICA1
Nombre -> SYS
Nombre -> GGSYS
Nombre -> DBA
Nombre -> IMP_FULL_DATABASE
Nombre -> GSMADMIN_INTERNAL
Nombre -> DANIEL
Nombre -> DV_REALM_OWNER
Nombre -> SCOTT
Nombre -> WMSYS
----------------------------
USUARIOS CON PERMISO DE DROP
----------------------------
Nombre -> SCOTT
Nombre -> SYS
Nombre -> OLAP_DBA
Nombre -> DBA
Nombre -> IMP_FULL_DATABASE
Nombre -> DANIEL
Nombre -> DV_REALM_OWNER
Nombre -> WMSYS
----------------------------
EXTENTS
----------------------------
ID -> 0  SIZE -> 65536  ID FICHERO -> 7

Procedimiento PL/SQL terminado correctamente.
```

POSTGRESQL

7. Averigua si pueden establecerse claúsulas de almacenamiento para las tablas o los espacios de tablas en Postgres.

En Postgres no disponemos de las cláusulas de almacenamiento, pero podemos usar una función llamada "pg_total_relation_size" que combinada con "pg_size_pretty" podemos obtener el tamaño mucho más amigable y reconocible para el ojo humano.

Vamos a realizar una prueba mostrando el tamaño de cada una de las bases de datos almacenadas en Postgres:

```
SELECT
pg_database.datname,pg_size_pretty(pg_database_size(pg_datab
ase.datname)) AS Tamaño FROM pg_database;
```

```
daniel=# SELECT
    pg_database.datname,
    pg_size_pretty(pg_database_size(pg_database.datname)) AS Tamaño
    FROM pg_database;
  datname  | tamaño
-----------+---------
 postgres  | 7753 kB
 daniel    | 7917 kB
 template1 | 7901 kB
 template0 | 7753 kB
 mundo1    | 7897 kB
 mundo2    | 7825 kB
 aeros     | 8189 kB
(7 filas)
```

Podemos obtener el dato de una Base de datos en concreto:

```
SELECT pg_size_pretty(pg_database_size('daniel')) as
Total_Database;
```

```
daniel=# SELECT
    pg_size_pretty (
        pg_database_size ('daniel')
    ) as Total_Database;
 total_database
----------------
 7917 kB
(1 fila)
```

MySQL

8. Averigua si existe el concepto de índice en MySQL y si coincide con el existente en ORACLE. Explica los distintos tipos de índices existentes.

Los tipos de índices de MySQL:

INDEX (NON-UNIQUE): Este tipo de índice admite valores duplicados para las columnas que componen el índice. No aplica restricciones, simplemente se utiliza para mejorar el rendimiento de consultas.
UNIQUE: Las columnas deben de tener un valor único, no se admiten valores duplicados. La única restricción que se aplica en este tipo de índice es que las columnas deben ser únicas.
PRIMARY: Aplica las restricciones del anterior,agregando la limitación de que solo puede existir un solo índice primario en cada una de las tablas.
FULLTEXT: Este tipo de índices van orientados a las búsquedas de texto. Se componen por todas las palabras que están contenidas en las columnas que contiene el índice. No se aplican restricciones y solo esta soportado por InnoDB y MyISAM.
SPATIAL: Este tipo de índices se utilizan para realizar consultas sobre datos que componen formas geométricas representadas en el espacio. No se aplican restricciones y solo esta soportado por InnoDB y MyISAM.

MongoDB

9. Explica los distintos motores de almacenamiento que ofrece MongoDB, sus características principales y en qué casos es más recomendable utilizar cada uno de ellos.

MMAPv1: Se utiliza principalmente para mapear páginas a memoria y es el motor de almacenamiento original, anteriormente se llamaba MMAP, pero en la versión 2.6 de MongoDB recibió una actualización que implementaba una API, y pasó a llamarse MMAPv1. Es rápido para lecturas pero lento para escrituras y se puede actualizar rápidamente y fácilmente.

Sus principales características son:

- Rendimiento escritura: Bueno
- Rendimiento Lectura: Bueno
- Soporta MongoDB Query Language
- Soporta sistema de réplicas
- Soporta Manager Cloud
- Integra el soporte de índice secundario
- Soporta Sharding
- Dispone de controles de seguridad

WiredTiger: Motor creado por los fundadores de **BerkeleyDB**. Está diseñado para las arquitecturas modernas de servidores (gran capacidad de memoria y núcleos), actualmente es el motor por defecto utilizado en **MongoDB** y es considerado un **"Database Toolkit"**.

Sus principales características son:

- Rendimiento escritura: Excelente
- Rendimiento Lectura: Excelente
- Soporta compresión de disco
- Soporta sistema de réplicas
- Soporta MongoDB Query Language
- Soporta Manager Cloud
- Integra el soporte de índice secundario
- Soporta Sharding
- Dispone de controles de seguridad

In-Memory: Los datos se almacenan en RAM y el acceso a los mismos es muy rápido. Por defecto, asigna el 50% de la RAM total, siendo el mínimo 1GB. Se utiliza para obtener bajas latencias en operaciones prolongadas de lectura/escritura.

Sus principales características son:

- Rendimiento escritura: Excelente
- Rendimiento Lectura: Excelente
- Latencia: Excelente
- Soporta sistema de réplicas
- Soporta MongoDB Query Language
- Soporta Manager Cloud
- Integra el soporte de índice secundario
- Soporta Sharding
- Dispone de controles de seguridad

Encrypted: Está construido sobre WiredTiger, y encripta la información de la base de datos.Se utiliza especialmente si tienes datos sensibles y necesitas de una protección adicional.

Sus principales características son:

- Rendimiento escritura: Bueno
- Rendimiento Lectura: Bueno
- Latencia: Bueno
- Soporta KMIP (Key Management Interoperability Protocol)
- Soporta sistema de réplicas
- Soporta MongoDB Query Language

- Soporta Manager Cloud
- Integra el soporte de índice secundario
- Soporta Sharding
- Dispone de controles de seguridad
- Encriptación nativa

PARTE GRUPAL

ORACLE

1. Crea un índice para la tabla EMP de SCOTT que agilice las consultas por nombre de empleado en un tablespace creado específicamente para índices. ¿Dónde deberías ubicar el fichero de datos asociado al índice? ¿Cómo se os ocurre que podrías probar si el índice resulta de utilidad?

En primer lugar vamos a crear un tablespace que utilizaremos para almacenar índices:

SQL> Create tablespace indices datafile 'ind.dbf' size 50M autoextend on;

```
SQL> Create tablespace indices datafile 'ind.dbf' size 50M autoextend on;

Tablespace creado.
```

En este caso, lo he creado en el mismo disco, pero la mejor opción sería crear el **tablespace** en otro disco, pues el rendimiento se incrementa.

A continuación vamos a crear el índice para la columna **"ENAME"** de la tabla **"EMP"** de **SCOTT**:

SQL> CREATE INDEX nombre_emp ON scott.emp(ENAME) TABLESPACE indices;

```
SQL> CREATE INDEX nombre_emp ON scott.emp(ENAME) TABLESPACE indices;

Indice creado.
```

Activamos la monitorización de uso de índice:

SQL> ALTER INDEX nombre_emp MONITORING USAGE;

```
SQL>     ALTER INDEX nombre_emp MONITORING USAGE;

Indice modificado.
```

Realizamos una consulta para hacer uso del índice

SQL> select deptno from scott.emp where ename='JAMES';

```
SQL> select deptno from scott.emp where ename='JAMES';

    DEPTNO
----------
        30
```

Para ver si se ha utilizado el índice ejecutamos la siguiente consulta:

SQL> select * from v$object_usage where table_name='EMP';

```
SQL> select * from v$object_usage where table_name='EMP';

INDEX_NAME
------------------------------------------------------------------------
TABLE_NAME
------------------------------------------------------------------------
MON USE START_MONITORING    END_MONITORING
--- --- ------------------- -------------------
NOMBRE_EMP
EMP
YES YES 01/25/2022 10:17:23
```

```
SQL> select deptno from scott.emp where ename='JAMES';

    DEPTNO
----------
        30
```

2. Realiza una consulta al diccionario de datos que muestre qué índices existen para objetos pertenecientes al esquema de SCOTT y sobre qué columnas están definidos. Averiguad en qué fichero o ficheros de datos se encuentran las extensiones de sus segmentos correspondientes.

Podemos ver el registro de índices en la tabla **dba_ind_columns**, donde podemos filtrar por el nombre del usuario propietarios de las tablas y mostrar tanto el nombre de los índices como las tablas y las columnas en las que están definidas. Por otro lado, en la tabla **dba_extents** encontraremos información sobre los segmentos, cuyo nombre coincide con el de las tablas. Aquí veremos el campo **File_ID,** que podemos utilizar para encontrar la ruta del fichero correspondiente en la tabla **dba_data_files.**

Dicho esto, procedemos a realizar la consulta:

```
SELECT DISTINCT
C.INDEX_NAME,C.TABLE_NAME,C.COLUMN_NAME,F.FILE_NAME

FROM DBA_IND_COLUMNS C, DBA_DATA_FILES F, DBA_EXTENTS E

WHERE E.SEGMENT_NAME = C.TABLE_NAME

AND E.FILE_ID = F.FILE_ID

AND C.TABLE_OWNER = 'C##SCOTT';
```

```
SELECT DISTINCT C.INDEX_NAME,C.TABLE_NAME,C.COLUMN_NAME,F.FILE_NAME
  2  FROM DBA_IND_COLUMNS C, DBA_DATA_FILES F, DBA_EXTENTS E
  3  WHERE E.SEGMENT_NAME = C.TABLE_NAME
  4  AND E.FILE_ID = F.FILE_ID
  5  AND C.TABLE_OWNER = 'C##SCOTT';


INDEX_NAME
--------------------------------------------------------------------------------
TABLE_NAME
--------------------------------------------------------------------------------
COLUMN_NAME
--------------------------------------------------------------------------------
FILE_NAME
--------------------------------------------------------------------------------
PK_DEPT
DEPT
DEPTNO
/opt/oracle/oradata/ORCLCDB/users01.dbf


INDEX_NAME
--------------------------------------------------------------------------------
TABLE_NAME
--------------------------------------------------------------------------------
COLUMN_NAME
--------------------------------------------------------------------------------
FILE_NAME
--------------------------------------------------------------------------------
PK_AUDITEMP
AUDIT_EMP
FECHA
/opt/oracle/oradata/ORCLCDB/users01.dbf
```

```
INDEX_NAME
--------------------------------------------------------------------------------
TABLE_NAME
--------------------------------------------------------------------------------
COLUMN_NAME
--------------------------------------------------------------------------------
FILE_NAME
--------------------------------------------------------------------------------
PK_AUDITEMP
AUDIT_EMP
TIPO_OPERACION
/opt/oracle/oradata/ORCLCDB/users01.dbf


INDEX_NAME
--------------------------------------------------------------------------------
TABLE_NAME
--------------------------------------------------------------------------------
COLUMN_NAME
--------------------------------------------------------------------------------
FILE_NAME
--------------------------------------------------------------------------------
PK_EMP
EMP
EMPNO
/opt/oracle/oradata/ORCLCDB/users01.dbf
```

3. Cread una secuencia para rellenar el campo deptno de la tabla dept de forma

coherente con los datos ya existentes. Insertad al menos dos registros haciendo uso de la secuencia.

Para crear la secuencia hemos creado un procedimiento que coge el mayor valor del campo **DEPTNO** de la tabla **DEPT** y lo incrementa en 10, para así guardarlo en una variable y tener el campo **START WITH** de la creación de la secuencia, aquí muestro el código:

```
create or replace procedure secuencia_deptno

is

seqval NUMBER;

 BEGIN

 select MAX(deptno)+10

 INTO seqval

 from scott.dept;

 execute immediate('CREATE SEQUENCE increase_deptno START WITH
'||seqval||' INCREMENT BY 10');

END secuencia_deptno;

/
```

```
SQL> create or replace procedure secuencia_deptno
is
seqval NUMBER;
  BEGIN
  select MAX(deptno)+10
  INTO seqval
  from scott.dept;
  execute immediate('CREATE SEQUENCE increase_deptno START WITH '||seqval||' INCREMENT BY 10');
END secuencia_deptno;
/  2    3    4    5    6    7    8    9   10

Procedimiento creado.
```

Ejecutamos el procedimiento y se creará la secuencia, para probar la secuencia insertamos datos en la tabla **DEPT** utilizando la secuencia:

```
SQL> INSERT INTO SCOTT.DEPT VALUES (increase_deptno.nextval, 'PRUEBAS', 'DOS HERMANAS');

1 fila creada.

SQL> INSERT INTO SCOTT.DEPT VALUES (increase_deptno.nextval, 'PRUEBAS2', 'DOS HERMANAS');

1 fila creada.

SQL> select * from scott.dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 PRUEBAS        DOS HERMANAS
        50 PRUEBAS2       DOS HERMANAS
```

4. Queremos limpiar nuestro fichero tnsnames.ora. Averiguad cuáles de sus entradas se están usando en algún enlace de la base de datos.

Podemos ver todos los enlaces que tenemos en la base de datos mediante la tabla **DBA_DB_LINKS**, y es en el campo host donde aparecerá la entrada correspondiente en el fichero **tnsnames.ora** .

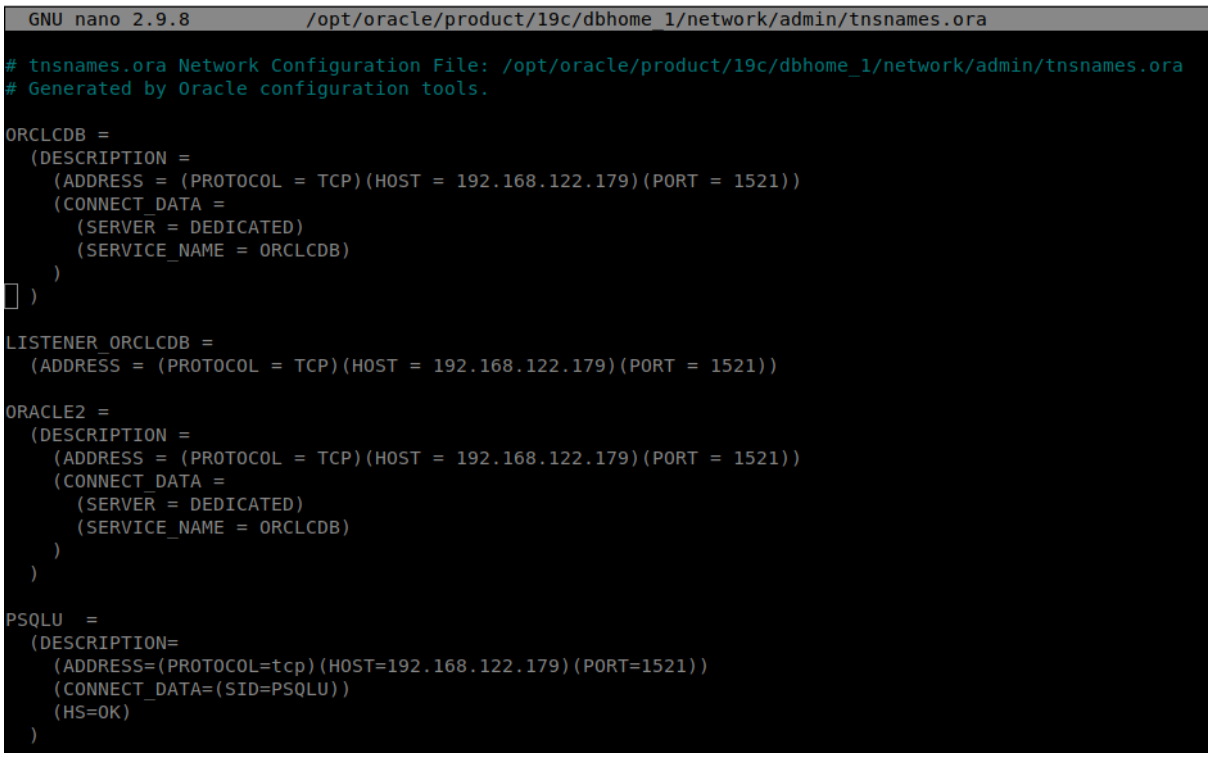
```
SQL> select db_link,host from dba_db_links;

DB_LINK
--------------------------------------------------------------------------------
HOST
--------------------------------------------------------------------------------
SYS_HUB
SEEDDATA

ORACLE2LINK
oracle2

POSTGRESLINK
PSQLU
```

Si abrimos el fichero **tnsnames.ora**, cuya ruta es: **/opt/oracle/product/19c/dbhome_1/network/admin/tnsnames.ora**, vemos que se están usando dos entradas en enlaces de la base de datos (**oracle2 y psqlu**):



Si tuviéramos entradas que no se estuvieran usando, exceptuando las que vienen por defecto, podríamos borrarlas en el fichero.

5. Meted las tablas EMP y DEPT de SCOTT en un cluster.

Para realizar este ejercicio hay que tener en cuenta una cosa muy importante, y es que el cluster tiene que crearse antes de insertar las tablas en él.

Para ello he conformado un script para que primero copie las tablas con y sus datos, luego he creado el cluster, he vuelto a copiar las tablas con “su nombre original” y por último insertaré dichas tablas en el cluster correspondiente.

– CREAR EL CLUSTER Y EL INDICE

```
create cluster cluster_scott (deptno number(2));

create index on cluster cluster_scott;
```

– RENOMBRAR LAS TABLAS

```
create table C##SCOTT.EMP2

as

select *

from C##SCOTT.EMP;

create table C##SCOTT.DEPT2
```

```
as
select *
from C##SCOTT.DEPT;
```

– BORRAR TABLAS EMP Y DEPT

```
drop table c##scott.emp;
drop table c##scott.dept;
```

– INTRODUCIR LAS TABLAS EN EL CLUSTER

```
create table C##SCOTT.EMP
cluster cluster_scott (deptno)
as
select *
from C##SCOTT.EMP2;
create table C##SCOTT.DEPT
cluster cluster_scott (deptno)
as
select *
from C##SCOTT.DEPT2;
```

6. **Realizad un procedimiento llamado *BalanceoCargaTemp* que balancee la carga de usuarios entre los tablespaces temporales existentes. Para ello averiguar cuántos TS existen y asignar los usuarios entre ellos de forma equilibrada. Si es necesario para comprobar su funcionamiento, crea tablespaces temporales nuevos**.

Tenemos 43 usuarios y necesitamos balancear la carga entre todos los **tablespaces** temporales existentes, en este caso 3.

Aquí puedes ver el número de usuarios:

```
SQL> SELECT count(USERNAME) FROM DBA_USERS;

COUNT(USERNAME)
---------------
             43
```

Y aquí los distintos **tablespaces** temporales existentes:

```
SQL> SELECT TABLESPACE_NAME FROM DBA_TABLESPACES WHERE CONTENTS='TEMPORARY';

TABLESPACE_NAME
------------------------------
TEMP1
TEMP2
TEMP3
```

Para balancear la carga hemos creado el siguiente procedimiento:

```
create or replace procedure BalanceoCargaTemp

is

cursor c_usuarios_temp

   is

   SELECT USERNAME FROM DBA_USERS;

v_numero number(12):=1;

v_total number(12);

begin

   SELECT COUNT(TABLESPACE_NAME) into v_total FROM
DBA_TABLESPACES WHERE CONTENTS='TEMPORARY';

   for v_user in c_usuarios_temp loop

   IF v_user.USERNAME!='XS$NULL' then

       execute immediate 'alter user '||v_user.USERNAME||'
temporary tablespace TEMP'||to_char(v_numero);

       v_numero:=v_numero+1;

   end if;

   if v_numero-1=v_total then

       v_numero:=1;

   end if;

   end loop;

end BalanceoCargaTemp;
```

```
/
```

Tras la ejecución del procedimiento comprobamos que realmente ha funcionado:

```
SQL> SELECT TABLESPACE_NAME,COUNT(USERNAME) FROM DBA_USERS RIGHT JOIN DBA_TABLESPACES ON TEMPORARY_TABLESPACE=TABLESPACE_NAME WHERE CONTENTS='TEM
PORARY' group by TABLESPACE_NAME;
alter user SCOTT temporarytablespace TEMP2;
TABLESPACE_NAME                COUNT(USERNAME)
------------------------------ ---------------
TEMP1                                       15
TEMP2                                       14
TEMP3                                       14

SQL>
```

MYSQL

7.Realizad un pequeño artículo o una entrada para un blog técnico explicando las limitaciones que presentan MySQL y Postgres para gestionar el almacenamiento de los datos respecto a ORACLE, si es que presentan alguna.

Mysql es un sistema gestor de base de datos con un sistema de almacenamiento sobre **tablespaces**, usando sus motores de almacenamiento, que son los motes de almacenamiento **InnoDB** y **NDB**.

NDB CLuster es el motor de almacenamiento usado por **MySQL Cluster**, para implementar tablas que se particionan en varias máquinas.

InnoDB es un motor de búsqueda, que proporciona tablas transacciones.

La similitud que podemos encontrar entre **Oracle** y **MYSQL** en el almacenamiento de datos, es que en ambos **SGBD** podemos crear y asignar **tablespaces**, un tamaño a dicha extensión, un tamaño inicial al espacio de tabla y que también el **tablespace** pueda extenderse automáticamente.

Pero cabe destacar que el **SGBD MySQL** tiene una desventaja respecto a Oracle.

La desventaja se encuentra en el motor de almacenamiento **InnoDB**, debido a que no se pueden asignar cuotas de almacenamiento, a los usuarios creados en la base de datos y en cualquier **tablespace** creado, donde en **Oracle** si es posible crear cuotas de almacenamiento para cada usuario y **tablespace** creado en la base de datos.

Los **tablespaces** creados con el motor de almacenamiento **InnoDB**, se asignan como un **tablespace** general, donde dicho **tablespace** es compartido a todos los usuarios, siendo similar al espacio de tabla del sistema.

POSTGRESQL

PostgreSQL no permite añadir cláusulas de almacenamiento de datos como si lo permiten **Oracle** Y **MySQL**.

Pero hay que dejar una cosa clara, y es que hay una función para **PostgreSQL** que nos permite controlar el espacio utilizado en tabla, donde se incluyen las tablas **TOAST** (*The Oversized-Attribute Storage Technique*), y también los índices de tablas almacenadas. La extensión descrita se denomina **pg_total_relation_size(regclass)**.

PostgreSQL no tiene cláusulas de almacenamiento, por lo que no podemos asignar una cuota de almacenamiento, crear **tablespace** y limitar el almacenamiento de una tabla en la base de datos.

También hay que destacar que existe una serie de inconvenientes en **PostgreSQL** con respecto a **Oracle**, y son las siguientes:

- Puntos de recuperación dentro de la transacción. Las transacciones se interrumpen completamente si se encuentra un fallo durante su ejecución.
- No se usan **tablespaces** donde se definan dónde almacenar la base de datos, el esquema, los índices, etc.
- El soporte a orientación a objetos es una simple extensión que ofrece prestaciones como la herencia, no es un soporte completo.

8.Explicad en qué consiste el sharding en MongoDB.

Cuando no es posible almacenar la totalidad de datos en un único servidor. **MongoDB** permite una operación de escalado horizontal. Esto quiere decir que la información puede repetirse en varios servidores, de forma que cada servidor tenga una parte del conjunto completo de datos. Para asegurar la alta disponibilidad del conjunto, cada fragmento se configura como un conjunto replicado. De esta forma, se asegura la tolerancia a fallos de cada shard por separado, independientemente de cuál de ellos almacene un dato concreto.

Para ello, utiliza los metadatos asociados a los datos almacenados en cada fragmento, que están almacenados en un servidor de configuración. Para asegurar la alta disponibilidad de este servidor se utiliza un conjunto replicado.

¿Cuando se utiliza Sharding?

Imaginemos que por necesidades se debe ampliar la memoria RAM del servidor, la ampliación nos supone un desembolso económico de 1o veces el coste inicial para una mejora en rendimiento del doble de rendimiento inicial. En este caos, distribuir los datos en varios fragmentos permitirá una copia más rápida por unidad , al permitir paralelización y un uso menor de ancho de banda. También se debe tener en cuenta que unos ficheros de datos de 15 TB implicará unos fichero de índice de tamaño proporcional. En resumidas cuentas , la primera opción debería ser el escalado vertical, a no ser que los costes no sean proporcionados al incremento de rendimiento obtenido, en cuyo caso se debe optar por un escalado horizontal.

El Sharding también se recomienda cuando el acceso a datos se vea beneficiado por procesos de paralización o cuando los datos se pueden distribuir geográficamente de forma que los clientes puedan acceder a su información estando almacenada en servidores cercanos.

¿Cómo se realiza la comunicación al trabajar con shared clusters (Conjunto fragmentado)?

Cuando la información se almacena en shards, la comunicación del cliente no se realiza directamente con los conjuntos replicados que almacenan la información, sino que se realiza con el proceso MongoS. Este proceso debe averiguar en qué shard se almacena la información solicitada , la localiza y se la devuelve al cliente. Estos metadatos indican que conjuntos de información están almacenados en cada shard, de forma que MongoS sabe a qué servidor debe acudir para obtener la información. Una vez se hayan juntado todos los datos,

se devuelve toda la información del cliente.

¿Qué es el balanceo?

El balanceo es la operación mediante la cual MongoDB distribuye de la manera más uniforme posible los datos entre los distintos shards. Este proceso de balanceo comprueba la distribución de los chunks entre los distintos shards, y analiza los umbrales de migración. Cuando detecta una descompensación, empiezan los rondsa de balanceo. Las rondas de balanceo se repiten tantas veces como sea necesario para obtener una distribución correcta de los chunks.

Cualquier operación de balanceo impacta en el rendimiento. StartBalancer inicia el proceso de balanceo. StopBalancer detiene el proceso de balanceo. En caso de detenerse en mitad de una ronda de balanceo, éste se detiene cuando finaliza dicha ronda.

9.Resolved el siguiente caso práctico en ORACLE:

En nuestra empresa existen tres departamentos: Informática, Ventas y Producción. En Informática trabajan tres personas: Pepe, Juan y Clara. En Ventas trabajan Ana y Eva y en Producción Jaime y Lidia.

a) Pepe es el administrador de la base de datos.
Juan y Clara son los programadores de la base de datos, que trabajan tanto en la aplicación que usa el departamento de Ventas como en la usada por el departamento de Producción.

Ana y Eva tienen permisos para insertar, modificar y borrar registros en las tablas de la aplicación de Ventas que tienes que crear, y se llaman Productos y Ventas, siendo propiedad de Ana.

Jaime y Lidia pueden leer la información de esas tablas pero no pueden modificar la información.

Crea los usuarios y dale los roles y permisos que creas conveniente.

Creación de usuarios y concesión de permisos:

```
create user c##pepe identified  by pepe;

grant dba to c##pepe;

grant create session to c##pepe;



create user c##juan identified  by juan;

create user c##clara identified by clara;
```

```
grant resource to c##juan;

grant resource to c##clara;

grant create session to c##juan;

grant create session to c##clara;


create user c##ana identified by ana;

create user c##eva identified by eva;

grant select,insert,update,delete on c##ana.productos to c##eva;

grant select,insert,update,delete on c##ana.ventas to c##eva;

grant create session to c##ana;

grant create session to c##eva;


create user c##jaime identified by jaime;

create user c##lidia identified by lidia;

grant select on c##ana.ventas to c##jaime;

grant select on c##ana.ventas to c##lidia;

grant select on c##ana.productos to c##jaime;

grant select on c##ana.productos to c##lidia;

grant create session to c##jaime;

grant create session to c##lidia;
```

Las tablas se crearán en el siguiente apartado, para aprovechar la creación de tablespaces.

b) Los espacios de tablas son System, Producción (ficheros prod1.dbf y prod2.dbf) y Ventas (fichero vent.dbf). Los programadores del departamento de Informática pueden crear objetos en cualquier tablespace de la base de datos, excepto en System. Los demás usuarios solo podrán crear objetos en su tablespace correspondiente teniendo un límite de espacio de 30 M los del departamento de Ventas y 100K los del de Producción. Pepe tiene cuota ilimitada en todos los espacios, aunque el suyo por defecto es System.

Tablespaces y tablas:

```
CREATE TABLESPACE ts_produccion
DATAFILE 'prod1.dbf' SIZE 100M,
'prod2.dbf' SIZE 100M AUTOEXTEND ON;

CREATE TABLESPACE ts_ventas
DATAFILE 'vent.dbf'
SIZE 100M AUTOEXTEND ON;

/** Como las tablas son de Ana, le asignaremos el tablespace de ventas. **/
CREATE TABLE c##ana.productos (
	ID VARCHAR2(4),
	NOMBRE VARCHAR2(20),
	PRECIO DECIMAL,
	STOCK NUMBER(3),
	CONSTRAINT pk_productos PRIMARY KEY (ID)
) TABLESPACE ts_ventas;

CREATE TABLE c##ana.ventas (
	ID_PRODUCTO VARCHAR2(4),
	CANTIDAD NUMBER(2),
	FECHA_VENTA DATE,
	DNI_CLIENTE VARCHAR2(9),
	CONSTRAINT pk_ventas PRIMARY KEY
(ID_PRODUCTO,FECHA_VENTA,DNI_CLIENTE),
	CONSTRAINT fk_id FOREIGN KEY (ID_PRODUCTO) REFERENCES
c##ana.productos (ID)
) TABLESPACE ts_ventas;

ALTER USER c##ana DEFAULT TABLESPACE ts_ventas;
ALTER USER c##eva DEFAULT TABLESPACE ts_ventas;
ALTER USER c##jaime DEFAULT TABLESPACE ts_produccion;
ALTER USER c##lidia DEFAULT TABLESPACE ts_produccion;
```

Cuotas en tablespaces:

```
ALTER USER c##ana QUOTA 30M ON ts_ventas;
ALTER USER c##eva QUOTA 30M ON ts_ventas;
ALTER USER c##jaime QUOTA 100K ON ts_produccion;
ALTER USER c##lidia QUOTA 100K ON ts_produccion;

ALTER USER c##pepe DEFAULT TABLESPACE SYSTEM;
GRANT UNLIMITED TABLESPACE TO c##pepe;

/** Juan y Clara son los programadores, por lo que deberán tener un espacio ilimitado
para crear tablas, al igual que Pepe. **/
GRANT UNLIMITED TABLESPACE TO c##juan;
GRANT UNLIMITED TABLESPACE TO c##clara;
```

```
ALTER USER c##juan QUOTA 0 ON SYSTEM;
ALTER USER c##clara QUOTA 0 ON SYSTEM;
```

C) Pepe quiere crear una tabla Prueba que ocupe inicialmente 256K en el tablespace Ventas.

```
CREATE TABLE c##pepe.prueba (
        ID VARCHAR2(2),
        CONSTRAINT pk_prueba PRIMARY KEY (ID)
) STORAGE (INITIAL 256K) TABLESPACE TS_VENTAS;
```

D) Pepe decide que los programadores tengan acceso a la tabla Prueba antes creada y puedan ceder ese derecho y el de conectarse a la base de datos a los usuarios que ellos quieran.

```
GRANT SELECT ON c##pepe.prueba TO c##juan WITH GRANT OPTION;
GRANT SELECT ON c##pepe.prueba TO c##clara WITH GRANT OPTION;
GRANT CONNECT TO c##clara WITH ADMIN OPTION;
GRANT CONNECT TO c##juan WITH ADMIN OPTION;
```

E) Lidia y Jaime dejan la empresa, borra los usuarios y el espacio de tablas correspondiente, detalla los pasos necesarios para que no quede rastro del espacio de tablas.

Eliminamos primero los usuarios, empleando **CASCADE** para que se borre también todo aquello que dependa de dichos usuarios:

```
DROP USER c##lidia CASCADE;
DROP USER c##jaime CASCADE;
```

Después, eliminamos el espacio de tabla de producción junto con su contenido y los ficheros de datos:

```
DROP TABLESPACE ts_produccion INCLUDING CONTENTS AND DATAFILES;
```