

# Parte Individual

## Adrián (Oracle)

**1. Crea un rol ROLPRACTICA1 con los privilegios necesarios para conectarse a la base de datos, crear tablas y vistas e insertar datos en la tabla EMP de SCOTT.**

```
alter session set "_ORACLE_SCRIPT"=true;
create role ROLPRACTICA1;

grant create session to ROLPRACTICA1;

grant create table, create view to ROLPRACTICA1;

grant insert on c##SCOTT.emp to ROLPRACTICA1;
```

```
SQL> alter session set "_ORACLE_SCRIPT"=true;
Sesión modificada.

SQL> create role ROLPRACTICA1;
Rol creado.

SQL> grant create session to ROLPRACTICA1;
Concesión terminada correctamente.

SQL> grant create table, create view to ROLPRACTICA1;
Concesión terminada correctamente.

SQL> grant insert on c##SCOTT.emp to ROLPRACTICA1;
Concesión terminada correctamente.
```

**2. Crea un usuario USRPRACTICA1 con el tablespace USERS por defecto y averigua que cuota se le ha asignado por defecto en el mismo. Sustitúyela por una cuota de 1M.**

```
create user USRPRACTICA1 identified by practica1 default tablespace USERS;
```

```
select tablespace_name, username, bytes, max_bytes
from dba_ts_quotas
where tablespace_name = 'USERS'
and username = 'USRPRACTICA1';
```

```
SQL> create user USRPRACTICA1 identified by practica1 default tablespace USERS;
```

Usuario creado.

```
SQL> select tablespace_name, username, bytes, max_bytes
2  from dba_ts_quotas
3  where username = 'USRPRACTICA1';
```

ninguna fila seleccionada

Como podemos observar, por defecto no se le asigna ninguna cuota al usuario.

### 3. Modifica el usuario USRPRACTICA1 para que tenga cuota 0 en el tablespace SYSTEM.

```
alter user USRPRACTICA1 quota 0 on SYSTEM;
```

### 4. Concede a USRPRACTICA1 el ROLPRACTICA1.

```
grant ROLPRACTICA1 to USRPRACTICA1;
```

### 5. Concede a USRPRACTICA1 el privilegio de crear tablas e insertar datos en el esquema de cualquier usuario. Prueba el privilegio. Comprueba si puede modificar la estructura o eliminar las tablas creadas.

```
grant create any table, insert any table to USRPRACTICA1;

alter user USRPRACTICA1 quota 1M on USERS;

revoke create any table, insert any table from USRPRACTICA1;

CREATE TABLE c##SCOTT.TABLAPRACTICA
(
ID NUMBER(10),
NOMBRE VARCHAR2(20),
APELLIDO VARCHAR2(20),
```

```
EDAD NUMBER(3)
```

```
);
```

```
insert into c##SCOTT.TABLAPRACTICA values (1,'Juan','Perez',20);
```

```
SQL> grant create any table, insert any table to USRPRACTICA1;
Concesi3n terminada correctamente.

SQL> alter user USRPRACTICA1 quota 1M on USERS;
Usuario modificado.

SQL>
```

```
SQL Plus
Introduzca el nombre de usuario: USRPRACTICA1
Introduzca la contrasea:
Hora de 3ltima Conexi3n Correcta: Mar Ene 11 2022 04:44:23 +01:00

Conectado a:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> CREATE TABLE c##SCOTT.TABLAPRACTICA
2 (
3   ID NUMBER(10),
4   NOMBRE VARCHAR2(20),
5   APELLIDO VARCHAR2(20),
6   EDAD NUMBER(3)
7 );

Tabla creada.

SQL> insert into c##SCOTT.TABLAPRACTICA values (1,'Juan','Perez',20);
1 fila creada.

SQL> drop table c##SCOTT.TABLAPRACTICA;
drop table c##SCOTT.TABLAPRACTICA
*
ERROR en l3nea 1:
ORA-01031: privilegios insuficientes

SQL>
```

## 6. Concede a USRPRACTICA1 el privilegio de leer la tabla DEPT de SCOTT con la posibilidad de que lo pase a su vez a terceros usuarios.

```
grant select on c##SCOTT.DEPT to USRPRACTICA1 with grant option;
```

## 7. Comprueba que USRPRACTICA1 puede realizar todas las operaciones previstas en el rol.

```
select * from c##SCOTT.DEPT;
```

```
SQL> select * from c##SCOTT.DEPT;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
40	OPERATIONS	BOSTON
30	SALES	CHICAGO

```
SQL> select user from dual;
```

```
USER
```

```
USRPRAC1ICA1
```

## 8. Quita a USRPRACTICA1 el privilegio de crear vistas. Comprueba que ya no puede hacerlo.

```
revoke create view from USRPRACTICA1;
```

```
SQL> create view c##SCOTT.VISTAPRACTICA as select * from c##SCOTT.DEPT;
create view c##SCOTT.VISTAPRACTICA as select * from c##SCOTT.DEPT
*
ERROR en línea 1:
ORA-01031: privilegios insuficientes

SQL> select user from dual;

USER
-----
USRPRACTICA1
```

## 9. Crea un perfil NOPARESEDECURRAR que limita a dos el número de minutos de inactividad permitidos en una sesión.

```
create profile NOPARESEDECURRAR limit idle_time 2;
```

## 10. Activa el uso de perfiles en ORACLE.

El uso de perfiles en oracle se encuentra activado por defecto, cuando no se especifica un perfil a la hora de crear un usuario, se le asigna el perfil **DEFAULT**.

## 11. Asigna el perfil creado a USRPRACTICA1 y comprueba su correcto funcionamiento.

```
ALTER SYSTEM SET resource_limit = TRUE;

alter user USRPRACTICA1 profile NOPARESEDECURRAR;
```

```
SQL> select to_char(SYSDATE, 'HH24:MI:SS') from dual;

TO_CHAR(
-----
13:49:04

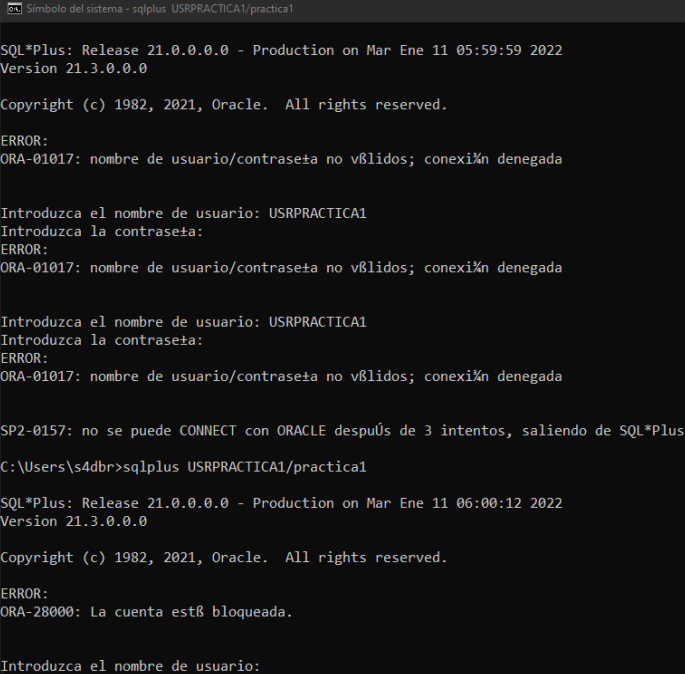
SQL> select to_char(SYSDATE, 'HH24:MI:SS') from dual;
select to_char(SYSDATE, 'HH24:MI:SS') from dual
*
ERROR en línea 1:
ORA-02396: ha excedido el tiempo máximo de inactividad, vuelva a conectarse
```

**12. Crea un perfil CONTRASEÑASEGURA especificando que la contraseña caduca mensualmente y sólo se permiten tres intentos fallidos para acceder a la cuenta. En caso de superarse, la cuenta debe quedar bloqueada indefinidamente.**

```
create profile SECUREPASSWORD limit password_life_time 30 failed_login_attempts
3 password_lock_time unlimited;
```

**13. Asigna el perfil creado a USRPRACTICA1 y comprueba su funcionamiento. Desbloquea posteriormente al usuario.**

```
SQL> create profile SECUREPASSWORD limit password_life_time 30 failed_login_attempts 3 password_lock_time unlimited;
Perfil creado.
SQL> alter user USRPRACTICA1 profile SECUREPASSWORD;
Usuario modificado.
SQL>
```



```
SQL*Plus: Release 21.0.0.0.0 - Production on Mar Ene 11 05:59:59 2022
Version 21.3.0.0.0

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

ERROR:
ORA-01017: nombre de usuario/contraseña no válidos; conexión denegada

Introduzca el nombre de usuario: USRPRACTICA1
Introduzca la contraseña:
ERROR:
ORA-01017: nombre de usuario/contraseña no válidos; conexión denegada

Introduzca el nombre de usuario: USRPRACTICA1
Introduzca la contraseña:
ERROR:
ORA-01017: nombre de usuario/contraseña no válidos; conexión denegada

SP2-0157: no se puede CONNECT con ORACLE después de 3 intentos, saliendo de SQL*Plus
C:\Users\s4dbr>sqlplus USRPRACTICA1/practica1

SQL*Plus: Release 21.0.0.0.0 - Production on Mar Ene 11 06:00:12 2022
Version 21.3.0.0.0

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

ERROR:
ORA-28000: La cuenta está bloqueada.

Introduzca el nombre de usuario:
```

```
alter user USRPRACTICA1 profile SECUREPASSWORD;

alter user USRPRACTICA1 account unlock;
```

**14. Consulta qué usuarios existen en tu base de datos.**

```
SQL> select username from dba_users;
```

USERNAME

-----  
SYS  
SYSTEM  
XS\$NULL  
OJVMSYS  
LBACSYS  
OUTLN  
DBSNMP  
APPQOSSYS  
DBSFUSER  
GGSYS  
ANONYMOUS

USERNAME

-----  
CTXSYS  
DVSYS  
DVF  
AUDSYS  
GSMADMIN\_INTERNAL  
OLAPSYS  
MDSYS  
XDB  
WMSYS  
GSMCATUSER  
MDDATA

USERNAME

-----  
C##EJERCICIO5  
SYSBACKUP  
REMOTE\_SCHEDULER\_AGENT  
GSMUSER  
SYSRAC  
GSMROOTUSER  
C##EJER6  
SI\_INFORMTN\_SCHEMA  
DIP  
ORDPLUGINS  
C##USUARIO2

USERNAME

-----  
SYSKM  
DGPDB\_INT  
ORDDATA

**15. Elige un usuario concreto y consulta qué cuota tiene sobre cada uno de los tablespaces.**

```
select tablespace_name, username, bytes, max_bytes  
from dba_ts_quotas  
where username = 'USRPRACTICA1';
```

```
SQL> select tablespace_name, username, bytes, max_bytes
       2      from dba_ts_quotas
       3      where username = 'USRPRACTICA1';
```

TABLESPACE_NAME	USERNAME	BYTES	MAX_BYTES
USERS	USRPRACTICA1	131072	1048576

## 16. Elige un usuario concreto y muestra qué privilegios de sistema tiene asignados.

```
select granted_role from dba_role_privs where grantee = 'USRPRACTICA1';
select * from dba_sys_privs where grantee='USRPRACTICA1';
```

```
SQL> select granted_role from dba_role_privs where grantee = 'USRPRACTICA1';
```

```
GRANTED_ROLE
```

```
-----
```

```
SQL> select * from dba_sys_privs where grantee='USRPRACTICA1';
```

GRANTEE	PRIVILEGE	ADM	COM	INH
USRPRACTICA1	INSERT ANY TABLE	NO	YES	NO
USRPRACTICA1	CREATE ANY TABLE	NO	YES	NO

## 17. Elige un usuario concreto y muestra qué privilegios sobre objetos tiene asignados.

```
select privilege, table_name from dba_tab_privs where grantee = 'USRPRACTICA1';
```

```
SQL> select privilege, table_name from dba_tab_privs where grantee = 'USRPRACTICA1';
```

```
PRIVILEGE
```

```
TABLE_NAME
```

```
-----
```

```
SELECT
```

```
DEPT
```

## 18. Consulta qué roles existen en tu base de datos.

```
select count(granted_role) from dba_role_privs;
```

```
select granted_role from dba_role_privs;
```

```
SQL> select count(granted_role) from dba_role_privs;
```

```
COUNT(GRANTED_ROLE)
```

```
-----  
161
```

```
SQL> select granted_role from dba_role_privs;
```

```
GRANTED_ROLE
```

```
-----  
GSMROOTUSER_ROLE
```

```
CONNECT
```

```
DBA
```

```
AUDIT_VIEWER
```

```
SELECT_CATALOG_ROLE
```

```
SELECT_CATALOG_ROLE
```

```
EXECUTE_CATALOG_ROLE
```

```
CAPTURE_ADMIN
```

```
CDB_DBA
```

```
AQ_ADMINISTRATOR_ROLE
```

```
AQ_ADMINISTRATOR_ROLE
```

```
GRANTED_ROLE
```

```
-----  
HS_ADMIN_SELECT_ROLE
```

```
GATHER_SYSTEM_STATISTICS
```

```
OPTIMIZER_PROCESSING_RATE
```

```
EM_EXPRESS_ALL
```

```
GDS_CATALOG_SELECT
```

```
XDB_WEBSERVICES_OVER_HTTP
```

```
GSMADMIN_ROLE
```

```
SODA_APP
```

```
DATAPATCH_ROLE
```

```
JAVAUSERPRIV
```

```
ORDADMIN
```

## 19. Elige un rol concreto y consulta qué usuarios lo tienen asignado.

```
select grantee from dba_role_privs where granted_role = 'ROLPRACTICA1';
```

```
SQL> select grantee from dba_role_privs where granted_role = 'ROLPRACTICA1';
```

```
GRANTEE
```

```
-----  
SYS
```

```
USRPRACTICA1
```

## 20. Elige un rol concreto y averigua si está compuesto por otros roles o no.

```
select granted_role from dba_role_privs where grantee = 'ROLPRACTICA2';
```

```
SQL> SELECT GRANTED_ROLE FROM DBA_ROLE_PRIVS WHERE GRANTEE = 'ROLPRACTICA2';
```

```
GRANTED_ROLE
```

```
-----  
ROLPRACTICA3
```

## 21. Consulta qué perfiles existen en tu base de datos.

```
select distinct profile from dba_profiles;
```



```
SQL> select distinct profile from dba_profiles;
```

```
PROFILE
```

```
-----  
NOPARESEDECURRAR  
ORA_STIG_PROFILE  
SECUREPASSWORD  
ORA_CIS_PROFILE  
DEFAULT
```

## 22. Elige un perfil y consulta qué límites se establecen en el mismo.

```
select resource_name, limit from dba_profiles where profile = 'SECUREPASSWORD';
```

```
SQL> select resource_name, limit from dba_profiles where profile = 'SECUREPASSWORD';
```

```
RESOURCE_NAME
```

```
-----  
LIMIT
```

```
-----  
FAILED_LOGIN_ATTEMPTS
```

```
3
```

```
COMPOSITE_LIMIT
```

```
DEFAULT
```

```
SESSIONS_PER_USER
```

```
DEFAULT
```

```
RESOURCE_NAME
```

```
-----  
LIMIT
```

```
-----  
CPU_PER_SESSION
```

```
DEFAULT
```

```
CPU_PER_CALL
```

```
DEFAULT
```

```
LOGICAL_READS_PER_SESSION
```

```
DEFAULT
```

```
RESOURCE_NAME
```

```
-----  
LIMIT
```

```
-----  
LOGICAL_READS_PER_CALL
```

```
DEFAULT
```

```
IDLE_TIME
```

```
DEFAULT
```

```
CONNECT_TIME
```

```
DEFAULT
```

```
RESOURCE_NAME
```

```
-----  
LIMIT
```

## 23. Muestra los nombres de los usuarios que tienen limitado el número de sesiones concurrentes.

```
select username from dba_users where profile in (select profile from
dba_profiles

where resource_name='SESSIONS_PER_USER' and limit!='DEFAULT');
```

```
SQL> select username from dba_users where profile in (select profile from dba_profiles
2 where resource_name='SESSIONS_PER_USER' and limit!='DEFAULT');
```

USERNAME

-----  
SYS  
SYSTEM  
XS\$NULL  
DBSNMP  
GSMCATUSER  
APPOSSYS  
MDDATA  
C##EJERCICIO5  
SYSBACKUP  
REMOTE\_SCHEDULER\_AGENT  
DBSFUSER

USERNAME

-----  
GSMUSER  
GGSYS  
ANONYMOUS  
SYSRAC  
GSMROOTUSER  
C##EJER6  
CTXSYS  
OJVMSYS  
SI\_INFORMTN\_SCHEMA  
DVSYS  
DVF

USERNAME

-----  
AUDSYS  
GSMADMIN\_INTERNAL  
DIP  
ORDPLUGINS  
C##USUARIO2  
OLAPSYS  
MDSYS  
LBACSYS  
SYSKM  
DGPDB\_INT  
ORDDATA

USERNAME

-----  
OUTLN  
ORACLE\_OCM

## 24. Realiza un procedimiento que reciba un nombre de usuario y un privilegio de sistema y nos muestre el mensaje 'SI, DIRECTO' si el usuario tiene ese privilegio concedido directamente, 'SI, POR ROL' si el usuario tiene ese privilegio en alguno de los roles que tiene concedidos y un 'NO' si el usuario no tiene dicho privilegio.

```
create or replace function rolsusuario (p_nombreusuario
dba_role_privs.grantee%type)
return number
is
```

```

v_rolsi number:=0;
begin
select count(grantee) into v_rolsi
from dba_role_privs
where p_nombreusuario=grantee;
return v_rolsi;
end rolusuario;
/

```

```

create or replace procedure privilegusuario (p_nombreusuario
dba_sys_privs.grantee%type, p_privilegio dba_sys_privs.privilege%type)
is
    v_cuenta number:=0;
    v_rol number:=0;
begin
    v_rol := rolusuario(p_nombreusuario);
    select count(grantee) into v_cuenta
    from dba_sys_privs
    where p_nombreusuario=grantee and p_privilegio=privilege;
    if v_cuenta !=0 then
        dbms_output.put_line('si, directo');
    elsif v_cuenta=0 and v_rol!=0 then
        dbms_output.put_line('si, por rol');
    else
        dbms_output.put_line('no');
    end if;
end privilegusuario;
/

```

```

create user USRPRACTICA2 identified by USRPRACTICA2;
create role ROLPRACTICA2;
grant create session to USRPRACTICA2;
grant ROLPRACTICA2 to USRPRACTICA2;

```

```

create role ROLPRACTICA3;
grant CREATE TABLE to ROLPRACTICA3;
grant ROLPRACTICA3 to ROLPRACTICA2;

```

```

create role ROLPRACTICA4;
grant CREATE VIEW to ROLPRACTICA4;
grant DROP PROFILE to ROLPRACTICA4;
grant ROLPRACTICA4 to ROLPRACTICA3;

```

```
SQL> exec ComprobarPrivilegio('USRPRACTICA2','CREATE TABLE');  
sí, por rol
```

Procedimiento PL/SQL terminado correctamente.

```
SQL> exec ComprobarPrivilegio('USRPRACTICA2','CREATE VIEW');  
sí, por rol
```

Procedimiento PL/SQL terminado correctamente.

```
SQL> grant DROP PROFILE to ROLPRACTICA4;
```

Concesión terminada correctamente.

```
SQL> exec ComprobarPrivilegio('USRPRACTICA2','DROP PROFILE');  
sí, por rol
```

Procedimiento PL/SQL terminado correctamente.

**25. Realiza un procedimiento llamado MostrarNumSesiones que reciba un nombre de usuario y muestre el número de sesiones concurrentes que puede tener abiertas como máximo y las que tiene abiertas realmente.**

```
create or replace function ComprobarSesion (p_nombreusuario  
DBA_SYS_PRIVS.GRANTEE%TYPE)  
return number  
is  
    v_sesionesabiertas number:=0;  
begin  
    select count(1) into v_sesionesabiertas  
    from v_$session  
    where username=p_nombreusuario;  
    return v_sesionesabiertas;  
end ComprobarSesion;  
/
```

```
create or replace procedure MostrarNumSesiones(p_nombreusuario  
DBA_SYS_PRIVS.GRANTEE%TYPE)  
is  
    v_maxsesiones DBA_USERS.PROFILE%TYPE;  
    v_sesionesabiertas number:=0;  
begin  
    v_sesionesabiertas:=ComprobarSesion(p_nombreusuario);  
    select PROFILE into v_maxsesiones
```

```

from DBA_USERS
where USERNAME=p_nombreusuario;
if v_maxsesiones='DEFAULT' then
dbms_output.put_line('El numero maximo de sesiones es ilimitado de manera
predeterminada y tiene ' || v_sesionesabiertas || ' sesiones abiertas.');
```

```

else
dbms_output.put_line('El numero maximo de sesiones es ' || v_maxsesiones || '
y tiene ' || v_sesionesabiertas || ' sesiones abiertas.');
```

```

end if;
END MostrarNumSesiones;
/
```

```

SQL> create or replace procedure MostrarNumSesiones(p_nombreusuario DBA_SYS_PRIVS.GRANTEE%TYPE)
2 is
3     v_maxsesiones DBA_USERS.PROFILE%TYPE;
4     v_sesionesabiertas number:=0;
5 begin
6     v_sesionesabiertas:=ComprobarSesion(p_nombreusuario);
7     select PROFILE into v_maxsesiones
8     from DBA_USERS
9     where USERNAME=p_nombreusuario;
10    if v_maxsesiones='DEFAULT' then
11        dbms_output.put_line('El numero maximo de sesiones es ilimitado de manera predeterminada y tiene ' || v_sesionesabiertas || ' sesione
s abiertas.');
```

```

12    else
13        dbms_output.put_line('El numero maximo de sesiones es ' || v_maxsesiones || ' y tiene ' || v_sesionesabiertas || ' sesiones abiertas.
');
```

```

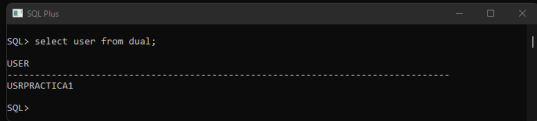
14    end if;
15 END MostrarNumSesiones;
16 /
```

Procedimiento creado.

```

SQL> exec MostrarNumSesiones('USRPRACTICA1');
El numero maximo de sesiones es NOPARESEDECURRAR y tiene 1 sesiones abiertas.
```

Procedimiento PL/SQL terminado correctamente.



# Jesús (Postgres y ORACLE)

## 1) Averigua que privilegios de sistema hay en Postgres y como se asignan a un usuario.

En primer lugar, me gustaría definir que son los privilegios de sistema. En los motores de bases de datos, los privilegios de sistema son los permisos o derechos de poder realizar cierta acción en la base de datos.

En PostgreSQL los permisos son administrados mediante roles. Podemos considerar un rol como un usuario de la base de datos o un grupo de usuarios de la base de datos, dependiendo de como esté configurado. Los roles pueden poseer objetos de la base de datos (por ejemplo, tablas) y pueden asignar privilegios sobre esos objetos a otros roles para controlar quién tiene control sobre estos. Además, es posible conceder permisos dentro de un rol a otro rol, lo cual permite al rol que se le han concedido los privilegios poder usarlos.

El concepto de roles subsume, es decir, incluye los conceptos de "usuarios" y "grupos". En las versiones de PostgreSQL anteriores a la 8.1, los usuarios y los grupos eran distintos tipos de entidades, pero ahora sólo hay roles. Cualquier rol puede actuar como un usuario, un grupo o ambos.

Hecha esta pequeña introducción vamos a ver el objetivo que nos pide el ejercicio, los distintos tipos de privilegios de sistema que existen en PostgreSQL:

- **SUPERUSER o NOSUPERUSER:** determinan si el nuevo rol es un superusuario, que puede anular todas las restricciones de acceso dentro de la base de datos. Debido a la importancia del superusuario deberemos usarlo sólo cuando sea realmente necesario y saber en cada momento quien lo posee. Deberemos ser superusuarios para crear un nuevo superusuario. Si no se especifica, NOSUPERUSER es el predeterminado.
- **CREATEDB o NOCREATEDB:** definen la capacidad de un rol para crear bases de datos. Si se especifica CREATEDB, el rol podrá crear nuevas bases de datos. Especificar NOCREATEDB negará a un rol la capacidad de crear bases de datos. Si no se especifica, NOCREATEDB es el valor predeterminado.
- **CREATEROLE o NOCREATEROLE:** estas cláusulas determinan si se le permitirá a un rol crear nuevos roles. Un rol con el privilegio CREATEROLE también puede modificar y descartar otros roles. Si no se especifica, NOCREATEROLE es el valor predeterminado.
- **INHERIT o NOINHERIT:** determinan si un rol hereda los privilegios de los roles de los que es miembro. Un rol con el atributo INHERIT puede usar automáticamente cualquier privilegio de base de datos que se haya otorgado a todos los roles de los que es miembro directa o indirectamente. Si no se especifica, INHERIT es el valor predeterminado.
- **LOGIN o NOLOGIN:** estas cláusulas determinan si un rol puede iniciar sesión. Un rol que tiene el atributo LOGIN puede considerarse como un usuario. Los roles sin este atributo son útiles para administrar los privilegios de la base de datos, pero no son usuarios en el sentido habitual de la palabra. Si no se especifica, NOLOGIN es el valor predeterminado, excepto cuando se invoca CREATE ROLE a través de su ortografía alternativa CREATE USER.
- **REPLICATION o NOREPLICATION:** determinan si un rol es un rol de replicación, es decir, hará de esclavo del servidor principal. Un rol debe tener este atributo (o ser un superusuario) para poder conectarse al servidor en modo de replicación (replicación física o lógica) y para poder crear o descartar ranuras de replicación. Un rol que tiene el atributo REPLICATION es un rol con muchos privilegios y solo debe usarse en roles que realmente se usan para la replicación. Si no se especifica, NOREPLICATION es el valor predeterminado. Debe ser superusuario para crear un nuevo rol que tenga el atributo REPLICATION.
- **BYPASSRLS o NOBYPASSRLS:** estas cláusulas determinan si un rol omite todas las políticas de seguridad de nivel de fila. NOBYPASSRLS es el valor predeterminado. Debe ser un superusuario para crear un nuevo rol que tenga el atributo BYPASSRLS.
- **CONNECTION LIMIT:** si el rol puede iniciar sesión, esto especifica cuántas conexiones simultáneas puede realizar el rol. -1, el valor predeterminado, significa que no hay límite. Sólo las conexiones normales se cuentan para este límite. Ni las transacciones preparadas ni las conexiones en segundo plano se cuentan para este límite.
- **[ ENCRYPTED ] PASSWORD 'contraseña' o PASSWORD NULL:** establece la contraseña del rol. Una contraseña sólo es usada para roles que tienen el atributo LOGIN, pero aún así se puede definir una para roles sin él. Si no se especifica ninguna contraseña, la contraseña se establecerá como nula y la autenticación de contraseña siempre fallará para ese usuario.

- **VALID UNTIL 'timestamp':** establece una fecha y hora después de la cual la contraseña del rol ya no es válida. Si se omite esta cláusula, la contraseña será válida para todo el tiempo.
- **IN ROLE role\_name:** la cláusula IN ROLE enumera uno o más roles existentes a los que se agregará inmediatamente el nuevo rol como nuevo miembro.
- **IN GROUP role\_name:** sintaxis obsoleta de IN ROLE.
- **ROLE role\_name:** enumera uno o más roles existentes que se agregan automáticamente como miembros del nuevo rol.
- **ADMIN role\_name:** La cláusula ADMIN es como ROLE, pero los roles nombrados se agregan al nuevo rol con ADMIN OPTION, lo que les otorga el derecho de otorgar membresía en este rol a otros.
- **USER role\_name:** sintaxis obsoleta de ROLE.
- **SYSID uid:** esta cláusula es ignorada, pero se acepta por compatibilidad con versiones anteriores.

Para asignar uno de estos privilegios usaremos la siguiente sintaxis:

```
CREATE ROLE nombre_del_nuevo_rol [ WITH ] nombre_del_privilegio;
```

Aunque el ejercicio nos pide los privilegios de sistema vamos a ver también los privilegios sobre objetos con los que contamos:

- **SELECT:** Permite hacer una consulta a cualquier columna o columnas específicas de una tabla, vista u otro objeto similar a una tabla. También nos permite copiar la información seleccionada a otra tabla. Necesitaremos este privilegio si queremos usar las sentencias DELETE o UPDATE haciendo referencia a los datos existentes de una tabla.
- **INSERT:** Permite insertar una nueva fila en una tabla, vista, etc. Se puede otorgar en columnas específicas, en cuyo caso sólo se pueden asignar esas columnas en el comando INSERT (por lo tanto, otras columnas recibirán valores predeterminados).
- **UPDATE:** Permite la actualización de cualquier columna o columnas específicas de una tabla, vista, etc. En la práctica, para casi cualquier uso de UPDATE necesitaremos tener los privilegios SELECT sobre dicha tabla.
- **DELETE:** Permite eliminar una fila de una tabla, vista, etc. De igual manera que en el uso del UPDATE, seguramente necesitaremos permisos de la sentencia SELECT, a no ser que nuestro uso del DELETE sea muy simple.
- **TRUNCATE:** Permite truncar una tabla, es decir, borrar todas sus filas.
- **REFERENCES:** Permite la creación de una foreign key que hace referencia a una tabla o columna(s) específica(s) de una tabla.
- **TRIGGER:** Permite la creación de triggers.
- **CREATE:** - Para bases de datos, permite crear nuevos esquemas y publicaciones dentro de la base de datos y permite instalar extensiones seguras dentro de la base de datos.
  - Para esquemas, permite que se creen nuevos objetos dentro del esquema. Para cambiar el nombre de un objeto existente, debe poseer el objeto y tener este privilegio para el esquema que lo contiene.

- Para tablespace, permite que se creen tablas, índices y archivos temporales dentro del tablespace, y permite que se creen bases de datos que tengan cierto tablespace predeterminado.
- La revocación de este privilegio no alterará la existencia o ubicación de los objetos existentes.
- **CONNECT**: Permite conectarse a la base de datos. Este privilegio se verifica al inicio de la conexión.
- **TEMPORARY**: Permite crear tablas temporales mientras se usa la base de datos.
- **EXECUTE**: Permite llamar a una función o procedimiento, incluido el uso de cualquier operador que se implemente sobre la función.
- **USAGE**: Para lenguajes de programación por procedimientos, permite el uso del lenguaje para la creación de funciones en ese lenguaje. Para los esquemas, permite el acceso a los objetos contenidos en el esquema (suponiendo que también se cumplan los requisitos de privilegios propios de los objetos).

Si queremos conceder uno de estos privilegios usaremos la siguiente forma:

```
GRANT privilegio_a_conceder ON nombre_del_objeto TO nombre_del_rol;
```

Para revocarlo usaremos:

```
REVOKE privilegio_a_revocar ON nombre_del_objeto FROM nombre_del_rol;
```

## 2) Averigua cual es la forma de asignar y revocar privilegios sobre una tabla concreta en Postgres.

Si queremos dar permisos sobre una tabla concreta en Postgres usaremos el siguiente comando:

```
GRANT privilegio_a_conceder
ON nombre_de_la_tabla
TO nombre_del_rol
[ WITH GRANT OPTION ]
```

Lo revocaremos usando la siguiente sintaxis:

```
REVOKE privilegio_a_revocar
ON nombre_de_la_tabla
FROM nombre_del_rol
```

En caso de querer dar permisos sólo sobre ciertas columnas de una tabla, tendremos que crear una vista con dichas columnas y dar permisos al rol deseado en la vista.



### **3) Averigua si existe el concepto de rol en Postgres y señala las diferencias con los roles de ORACLE.**

Al igual que en el primer ejercicio, me gustaría, a modo de introducción, que es en general un rol en un gestor de bases de datos.

Los roles de base de datos simplifican el proceso de gestión de privilegios, ya que se pueden otorgar privilegios a un rol y luego otorgar el rol a distintos usuarios. Cuando desee revocar privilegios para un usuario, simplemente tiene que revocar la autorización de rol del usuario, en vez de revocar cada privilegio individual.

Una vez sabemos que significado puede tener un rol en una base de datos, vamos a ver lo que significan en PostgreSQL y en Oracle.

Como ya hemos visto en el primer apartado en PostgreSQL a partir de la versión 8.1 el rol incluye los conceptos de usuario y grupo, es decir, sólo existe el concepto de rol y este puede actuar como un usuario, como un grupo o como ambos.

Por lo tanto, ahora nos faltaría saber que significa el concepto de rol en Oracle. Lo primero que debemos saber es que en Oracle si podemos diferenciar el rol del usuario, al contrario de PostgreSQL. En Oracle un rol no es más que un conjunto de privilegios que pueden ser otorgados a los usuarios o a otros roles. La segunda diferencia que nos encontramos son los grupos. Mientras que en PostgreSQL podemos crear un grupo con distintos usuarios y conceder un rol a este grupo, por lo que concederemos los privilegios del rol a todos los usuarios del grupo, en Oracle no contaremos con esta opción ya que es enfocado de una manera distinta. En Oracle si tenemos n usuarios y queremos concederles a todos los mismos privilegios crearemos un rol con el conjunto de privilegios y se los otorgaremos a los usuarios.

### **4) Averigua si existe el concepto de perfil como conjunto de límites sobre el uso de recursos o sobre la contraseña en Postgres y señala las diferencias con los perfiles de ORACLE.**

En PostgreSQL no contamos con el concepto de perfil, por lo tanto no nos aporta facilidades para limitar los recursos que consume un usuario, consulta o base de datos en particular, o de manera correspondiente para establecer prioridades de manera que un usuario/consulta/base de datos obtenga más recursos que otros. Respecto a la contraseña tampoco nos dará muchas opciones, podemos destacar PAM(Pluggable Authentication Modules), un mecanismo de autenticación que se usa sólo para validar pares de nombre de usuario/contraseña y, opcionalmente, el nombre de host remoto conectado o la dirección IP. Para poder usar este método el usuario debe existir en la base de datos antes de que se pueda usar PAM para la autenticación. Las únicas opciones parecidas a los perfiles serían CONNECTION LIMIT, para limitar el número de conexiones simultáneas, y VALID UNTIL para establecer la fecha que será válida la contraseña.

Respecto a los perfiles en Oracle es un conjunto de límites de recursos a los que se le asigna un nombre. Los perfiles pueden ser asignados a diferentes usuarios o grupos de usuarios para controlar y limitar el uso de recursos del sistema. Para que estos perfiles

funcionen segun los valores deberemos activar el parámetro RESOURCE\_LIMIT debe estar configurado en TRUE. Para activarlos deberemos ejecutar la siguiente sentencia:

```
ALTER SYSTEM SET RESOURCE_LIMIT=TRUE
```

Si queremos limitar los recursos contamos con varias opciones:

- **SESSIONS\_PER\_USER**: para limitar el número de conexiones simultáneas, muy parecido al CONNECTION LIMIT de PostgreSQL.
- **CPU\_PER\_SESSION**: para limitar el tiempo de CPU para una sesión.
- **CPU\_PER\_CALL**: para limitar el tiempo de CPU por llamada. Las llamadas o call es similar a execute.
- **CONNECT\_TIME**: usaremos esta opción para especificar el límite de tiempo total transcurrido para una sesión.
- **IDLE\_TIME**: limitaremos los períodos permitidos de tiempo inactivo continuo durante una sesión.
- **LOGICAL\_READS\_PER\_SESSION**: número permitido de bloques de datos leídos en una sesión, incluidos los bloques leídos de la memoria y el disco.
- **LOGICAL\_READS\_PER\_CALL**: número permitido de bloques de datos leídos para que una llamada procese una instrucción SQL.
- **PRIVATE\_SGA**: para limitar la cantidad de espacio privado que una sesión puede asignar en el grupo compartido del área global del sistema.
- **COMPOSITE\_LIMIT**: para limitar el costo total de recursos para una sesión, expresado en unidades de servicio. Oracle calculará las unidades de servicio totales como una suma equilibrada de CPU\_PER\_SESSION, CONNECT\_TIME, LOGICAL\_READS\_PER\_SESSION y PRIVATE\_SGA.

Mientras que para el tema de las contraseñas disponemos de las siguientes opciones:

- **FAILED\_LOGIN\_ATTEMPTS**: parámetro para especificar el número de intentos fallidos de iniciar sesión en la cuenta de usuario antes de que se bloquee la cuenta.
- **PASSWORD\_LIFE\_TIME**: número de días que se puede usar la misma contraseña para la autenticación.
- **PASSWORD\_REUSE\_TIME**: número de días que deben pasar antes de poder reutilizar una contraseña.
- **PASSWORD\_REUSE\_MAX**: especifica el número de cambios de contraseña necesarios antes de poder reutilizar la contraseña actual.
- **PASSWORD\_LOCK\_TIME**: cantidad de días que se bloqueará una cuenta después de la cantidad especificada de intentos de inicio de sesión fallidos consecutivos.
- **PASSWORD\_GRACE\_TIME**: número de días después de que comience el período de gracia durante los cuales se emite una advertencia y se permite el inicio de sesión. Si la contraseña no se cambia durante el período de gracia, la contraseña caducará.
- **PASSWORD\_VERIFY\_FUNCTION**: permite pasar un script de verificación de complejidad de contraseña PL/SQL como argumento a la instrucción CREATE PROFILE. Aunque Oracle nos ofrece un script predeterminado, podemos crear el nuestro propio.

Como podemos comprobar Oracle es muy superior a la hora de poder restringir los límites de los recursos del sistema como de las opciones que podemos manejar con las contraseñas que PostgreSQL.

## 5) Realiza consultas al diccionario de datos de Postgres para averiguar todos los privilegios que tiene un usuario concreto.

Para ver los privilegios sobre las tablas:

```
SELECT * FROM information_schema.table_privileges where  
grantee='nombre_del_rol';
```

Para ver si el usuario es superusuario, si puede crear tablas, si tiene la opción replication o la opción bypassrls.

```
SELECT * FROM pg_catalog.pg_user where username='nombre_del_rol';
```

## 6) Realiza consultas al diccionario de datos en Postgres para averiguar qué usuarios pueden consultar una tabla concreta.

Usaremos la siguiente sentencia:

```
SELECT grantee FROM INFORMATION_SCHEMA.TABLE_PRIVILEGES where  
table_name='nombre_de_la_tabla' and privilege_type='SELECT';
```

## 7) Realiza una función de verificación de contraseñas que compruebe que la contraseña difiere en más de tres caracteres de la anterior y que la longitud de la misma es diferente de la anterior. Asígnala al perfil CONTRASEÑASEGURA. Comprueba que funciona correctamente.

```
create or replace function comparar_3_char (p_contrasena varchar2, p_old  
varchar2)  
return boolean  
is  
v_old varchar2(100):=p_old;  
v_longitud_old number:=length(v_old);  
v_letra varchar2(1);  
v_new varchar2(100):=p_contrasena;  
v_longitud_new number:=length(v_new);  
v_letra_nueva varchar2(1);  
v_contador number:=0;  
begin  
for i in 1..v_longitud_old loop  
v_letra:=substr(p_old,i,1);  
v_letra_nueva:=substr(p_contrasena,i,1);  
if v_letra_nueva=v_letra then  
v_contador:=v_contador+1;
```

```

end if;
end loop;
if v_contador>3 then
return FALSE;
else
return TRUE;
end if;
end comparar_3_char;
/

create or replace function comparar_longitud(p_contrasena varchar2,p_old
varchar2)
return boolean
is
v_longitud_old number:=length(p_old);
v_longitud_new number:=length(p_contrasena);
begin
if v_longitud_new=v_longitud_old then
return FALSE;
else
return TRUE;
end if;
end comparar_longitud;
/

create or replace function verificar_contrasena(p_user varchar2,p_contrasena
varchar2,p_oldcontrasena varchar2)
return boolean
is
begin
if (NOT comparar_3_char(p_contrasena,p_oldcontrasena)) then
raise_application_error(-20100, 'La contrasena nueva debe diferir en tres
caracteres con la anterior');
end if;
if (NOT comparar_longitud(p_contrasena,p_oldcontrasena)) then
raise_application_error(-20200, 'La password nueva debe tener un numero de
caracteres diferente a la anterior');
end if;
return TRUE;
end verificar_contrasena;
/

CREATE USER PRUEBA IDENTIFIED BY PRUEBA DEFAULT TABLESPACE USERS TEMPORARY
TABLESPACE TEMP PROFILE DEFAULT;
GRANT CONNECT, RESOURCE TO PRUEBA;
ALTER USER PRUEBA ACCOUNT UNLOCK;
CREATE PROFILE CONTRASENASEGURA LIMIT PASSWORD_VERIFY_FUNCTION
verificar_contrasena;
ALTER USER PRUEBA profile CONTRASENASEGURA;
ALTER USER PRUEBA IDENTIFIED BY PRUEBA REPLACE PRUEBA;

```

## Prueba de funcionamientos.

En primer lugar creamos un usuario llamado PRUEBA y contraseña PRUEBA con permisos de conexión.

```
SQL> CREATE USER PRUEBA IDENTIFIED BY PRUEBA DEFAULT TABLESPACE USERS TEMPORARY TABLESPACE TEMP PROFILE DEFAULT;
User created.

SQL> GRANT CONNECT, RESOURCE TO PRUEBA;
Grant succeeded.

SQL> ALTER USER PRUEBA ACCOUNT UNLOCK;
User altered.
```

Creamos el perfil CONTRASENASEGURA y le asignamos nuestra función que va a verificar la contraseña. A continuación, asignamos el perfil a nuestro usuario.

```
SQL> CREATE PROFILE CONTRASENASEGURA LIMIT PASSWORD_VERIFY_FUNCTION verificar_contraseña;
Profile created.

SQL> ALTER USER PRUEBA profile CONTRASENASEGURA;
User altered.
```

Intentamos cambiar la contraseña poniendo la misma, vemos como no nos lo permite.

```
SQL> ALTER USER PRUEBA IDENTIFIED BY PRUEBA REPLACE PRUEBA;
ALTER USER PRUEBA IDENTIFIED BY PRUEBA REPLACE PRUEBA
*
ERROR at line 1:
ORA-28003: password verification for the specified password failed
ORA-20100: La contraseña nueva debe diferir en tres caracteres con la anterior
```

Intentamos cambiar la contraseña poniendo una de la misma longitud pero con distinto caracteres, tampoco nos permite.

```
SQL> ALTER USER PRUEBA IDENTIFIED BY 123456 REPLACE PRUEBA;
ALTER USER PRUEBA IDENTIFIED BY 123456 REPLACE PRUEBA
*
ERROR at line 1:
ORA-28003: password verification for the specified password failed
ORA-20200: La password nueva debe tener un numero de caracteres diferente a la anterior
```

Por último, la intentamos cambiar por una contraseña de distinta longitud y que difiere en más de 3 caracteres. Ahora sí nos lo permite.

```
SQL> ALTER USER PRUEBA IDENTIFIED BY NUEVAPASSWORD REPLACE PRUEBA;
User altered.
```

## 8) Realiza un procedimiento llamado MostrarPrivilegiosdelRol que reciba el nombre de un rol y muestre los privilegios de sistema y los privilegios sobre objetos que lo componen.

```
CREATE or replace PACKAGE my_public_package IS
type v_array IS varray(200) OF VARCHAR2(100);
lv_array v_array:=v_array();
v_puntero number:=0;
type v_array_duplicate IS varray(200) OF VARCHAR2(100);
lv_array_duplicate v_array_duplicate:=v_array_duplicate();
```

```

v_contador number:=0;
END;
/

create or replace procedure Mostrar_rol_priv_sys(p_rol
DBA_ROLE_PRIVS.GRANTEE%TYPE)
is
    v_privilegios varchar2(30);
    cursor c_privilegios is
    select PRIVILEGE
    from dba_sys_privs
    where grantee=p_rol;
begin
    for i in c_privilegios loop
        my_public_package.v_puntero:=my_public_package.v_puntero+1;
        my_public_package.lv_array.extend;
        my_public_package.lv_array(my_public_package.v_puntero):=i.PRIVILEGE;
        dbms_output.put_line(i.PRIVILEGE);
    end loop;
end Mostrar_rol_priv_sys;
/

create or replace procedure Mostrar_rol_priv_object(p_rol varchar2)
is
    v_privilegios varchar2(30);
    cursor c_privilegios is
    select privilege,table_name
    from dba_tab_privs
    where grantee=p_rol;
begin
    for i in c_privilegios loop
        my_public_package.v_puntero:=my_public_package.v_puntero+1;
        my_public_package.lv_array.extend;
        my_public_package.lv_array(my_public_package.v_puntero):=i.PRIVILEGE
        || ' sobre ' || i.table_name;
        dbms_output.put_line('Privilegio ' || i.privilege || ' sobre
        ' || i.table_name);
    end loop;
end Mostrar_rol_priv_object;
/

create or replace procedure Mostrar_all_priv_associate_rol(p_rol
DBA_ROLE_PRIVS.GRANTEE%TYPE)
is
    cursor c_privilegios is
    select *
    from dba_role_privs
    where grantee=p_rol;
begin
    for v_cursor in c_privilegios loop
        Mostrar_rol_priv_sys(v_cursor.GRANTED_ROLE);
        Mostrar_rol_priv_object(v_cursor.GRANTED_ROLE);
        Mostrar_all_privs_associte_rol(v_cursor.GRANTED_ROLE);
    end loop;
end Mostrar_all_priv_associate_rol;

```

```

/

create or replace procedure Mostrar_all_privs_rol(p_rol
DBA_ROLE_PRIVS.GRANTEE%TYPE)
is
begin
    my_public_package.v_puntero:=0;
    dbms_output.put_line('Estos son los privilegios del rol
'||p_rol||':');
    Mostrar_rol_priv_sys(p_rol);
    Mostrar_rol_priv_object(p_rol);
    Mostrar_all_priv_associate_rol(p_rol);
end Mostrar_all_privs_rol;
/

create or replace procedure ver_duplicados
is
begin
for i in 1..my_public_package.v_puntero loop
    my_public_package.lv_array_duplicate.extend;
    my_public_package.lv_array_duplicate(i):=0;
    for j in 1..my_public_package.v_puntero loop
        if j!=i then
            if (my_public_package.lv_array(i)=my_public_package.lv_array(j)and
(my_public_package.lv_array_duplicate(i)=0)) then
                my_public_package.lv_array_duplicate(j):=1;
            end if;
        end if;
    end loop;
    for i in 1..my_public_package.v_puntero loop
        dbms_output.put_line(my_public_package.lv_array_duplicate(i));
    end loop;
end ver_duplicados;
/

create or replace procedure Mostrar_distinct_privs(p_rol
DBA_ROLE_PRIVS.GRANTEE%TYPE)
is
begin
Mostrar_all_privs_rol(p_rol);
dbms_output.put_line('Estos son los privilegios del rol '||p_rol||' sin
duplicar:');
for i in 1..my_public_package.v_puntero loop
    if my_public_package.lv_array_duplicate(i)=0 then
        dbms_output.put_line(my_public_package.lv_array(i));
    end if;
end loop;
end Mostrar_distinct_privs;
/

create role PRUEBA1;
grant CREATE TRIGGER to PRUEBA1;

create role PRUEBA2;

```

```

grant CREATE TRIGGER to PRUEBA2;
grant PRUEBA2 to PRUEBA1;

create role PRUEBA3;
grant create CLUSTER to PRUEBA3;
grant PRUEBA3 to PRUEBA2;

exec Mostrar_distinct_privs('PRUEBA1');

```

Lo probamos con un rol que viene en Oracle. Como vemos nos muestra en primer lugar todos los privilegios del rol, pero nos surge un problema que también nos aparecen los repetidos. Por ello, para mostrar que funciona correctamente he mostrado por pantalla todos los privilegios incluido los repetidos y justo abajo todos los privilegios pero mostrando una única vez los repetidos.

```

SQL> exec Mostrar_distinct_privs('HR');
Estos son los privilegios del rol HR:
CREATE VIEW
UNLIMITED TABLESPACE
CREATE DATABASE LINK
CREATE SEQUENCE
CREATE SESSION
ALTER SESSION
CREATE SYNONYM
Privilegio EXECUTE sobre DBMS_STATS
CREATE TRIGGER
CREATE SEQUENCE
CREATE TYPE
CREATE PROCEDURE
CREATE CLUSTER
CREATE OPERATOR
CREATE INDEXTYPE
CREATE TABLE
CREATE SESSION
Estos son los privilegios del rol HR sin duplicar:
CREATE VIEW
UNLIMITED TABLESPACE
CREATE DATABASE LINK
ALTER SESSION
CREATE SYNONYM
EXECUTE sobre DBMS_STATS
CREATE TRIGGER
CREATE SEQUENCE
CREATE TYPE
CREATE PROCEDURE
CREATE CLUSTER
CREATE OPERATOR
CREATE INDEXTYPE
CREATE TABLE
CREATE SESSION
PL/SQL procedure successfully completed.

SQL>

```

Como comentamos en clase, voy a crear un "árbol" de roles. He creado el rol PRUEBA1 con permisos de creación de TRIGGERS, el rol PRUEBA2 también tendrá el mismo permiso y asignamos el rol PRUEBA2 a PRUEBA1. Por último creamos el rol PRUEBA3 con permisos de creación de CLUSTERS y se lo asignamos a PRUEBA2.



```

SQL> create role PRUEBA1;
Role created.

SQL> grant CREATE TRIGGER to PRUEBA1;
Grant succeeded.

SQL>
SQL> create role PRUEBA2;
Role created.

SQL> grant CREATE TRIGGER to PRUEBA2;
Grant succeeded.

SQL> grant PRUEBA2 to PRUEBA1;
Grant succeeded.

SQL>
SQL> create role PRUEBA3;
Role created.

SQL> grant create CLUSTER to PRUEBA3;
Grant succeeded.

SQL> grant PRUEBA3 to PRUEBA2;
Grant succeeded.

```

Por lo tanto, el rol PRUEBA1 debería de tener el permiso de creación de TRIGGER, repetido, que mostraremos una sola vez y el permiso de creación de CLUSTER. Comprobamos su funcionamiento.

```

SQL> exec Mostrar_distinct_privs('PRUEBA1');
Estos son los privilegios del rol PRUEBA1:
CREATE TRIGGER
CREATE TRIGGER
CREATE CLUSTER
Estos son los privilegios del rol PRUEBA1 sin duplicar:
CREATE TRIGGER
CREATE CLUSTER

PL/SQL procedure successfully completed.

```

## Dani (MySQL y ORACLE)

### 1. Averigua que privilegios de sistema hay en MySQL y como se asignan a un usuario.

En MariaDB existen seis niveles de privilegios:

- **Privilegios globales:** Es el nivel más alto de privilegios y actúan a nivel de todas las bases de datos. Incluye privilegios para administrar las bases de datos, cuentas de

usuarios, etc. Son almacenados en la tabla "mysql.user".

- **Privilegios de base de datos:** Como en MariaDB podemos tener varias bases de datos, este nivel se refiere a los privilegios que tienen control sobre una base de datos específica (crear tablas, funciones, privilegios para las tablas, etc). Estos privilegios de almacenan en la tabla "mysql.db".
- **Privilegios de tablas:** Son los privilegios que afectan a una tabla de una base de datos en concreto (crear o modificar las filas o registros).
- **Privilegios sobre columnas:** Son los privilegios que afectan a una columna en concreto de una tabla.
- **Privilegios sobre funciones:** Son los privilegios que afectan a la creación y uso de las funciones de una base de datos.
- **Privilegios sobre procedimientos:** Son los privilegios que afectan a la creación y uso de los procedimientos de una base datos.

En MariaDB podemos ver los privilegios existentes si utilizamos "show privileges" en la consola de la base de datos:

Privilege	Context	Comment
Alter the table	Tables	To alter
Alter routine or drop stored functions/procedures	Functions,Procedures	To alter
Create new databases and tables	Databases,Tables,Indexes	To create
Create routine CREATE FUNCTION/PROCEDURE	Databases	To use
Create temporary tables CREATE TEMPORARY TABLE	Databases	To use
Create view new views	Tables	To create
Create user new users	Server Admin	To create
Delete existing rows	Tables	To delete
Delete history versioning table historical rows	Tables	To delete
Drop databases, tables, and views	Databases,Tables	To drop
Event alter, drop and execute events	Server Admin	To create,
Execute stored routines	Functions,Procedures	To execute
File and write files on the server	File access on server	To read
Grant option other users those privileges you possess	Databases,Tables,Functions,Procedures	To give to
Index	Tables	To create

or drop indexes			
Insert	Tables		To insert
data into tables			
Lock tables	Databases		To use
LOCK TABLES (together with SELECT privilege)			
Process	Server Admin		To view
the plain text of currently executing queries			
Proxy	Server Admin		To make
proxy user possible			
References	Databases,Tables		To have
references on tables			
Reload	Server Admin		To reload
or refresh tables, logs and privileges			
Binlog admin	Server		To purge
binary logs			
Binlog monitor	Server		To use
SHOW BINLOG STATUS and SHOW BINARY LOG			
Binlog replay	Server		To use
BINLOG (generated by mariadb-binlog)			
Replication master admin	Server		To monitor
connected slaves			
Replication slave admin	Server		To
start/stop slave and apply binlog events			
Slave monitor	Server		To use
SHOW SLAVE STATUS and SHOW RELAYLOG EVENTS			
Replication slave	Server Admin		To read
binary log events from the master			
Select	Tables		To
retrieve rows from table			
Show databases	Server Admin		To see all
databases with SHOW DATABASES			
Show view	Tables		To see
views with SHOW CREATE VIEW			
Shutdown	Server Admin		To shut
down the server			
Super	Server Admin		To use
KILL thread, SET GLOBAL, CHANGE MASTER, etc.			
Trigger	Tables		To use
triggers			
Create tablespace	Server Admin		To
create/alter/drop tablespaces			
Update	Tables		To update
existing rows			
Set user	Server		To create
views and stored routines with a different definer			
Federated admin	Server		To execute
the CREATE SERVER, ALTER SERVER, DROP SERVER statements			
Connection admin	Server		To bypass
connection limits and kill other users' connections			
Read_only admin	Server		To perform
write operations even if @@read_only=ON			
Usage	Server Admin		No
privileges - allow connect only			
+-----+-----+-----+			

```
-----+  
41 rows in set (0.000 sec)
```

Para asignar estos privilegios a un usuario en concreto, usamos la siguiente sintaxis:

```
GRANT "NOMBRE_PRIVILEGIO" ON "BD-TABLA-OBJETO" TO "USUARIO"@"HOST" [IDENTIFIED BY "CONTRASEÑA"] [WITH GRANT OPTION];
```

Ejemplos:

```
GRANT CREATE VIEWS ON prueba.* to "ejemplo"@"localhost" with grant option; ->  
Da el permiso para crear vistas sobre cualquier tabla de la base de datos  
"prueba" al usuario "ejemplo", con la posibilidad de que pase este privilegio a  
cualquier otro usuario.
```

```
GRANT SHOW DATABASES ON *.* TO "ejemplo"@"localhost"; -> Da el permiso para ver  
todas las bases de datos con "show databases". Si no tuviera este permiso, solo  
podría ver las bases de datos sobre las que tuviera algún tipo de privilegio.
```

## 2. Averigua cual es la forma de asignar y revocar privilegios sobre una tabla concreta en MySQL.

En MariaDB, la forma básica de asignar privilegios sobre una tabla es la siguiente:

```
GRANT "PRIVILEGIO" ON "DB"."NOMBRE_TABLA" TO "USUARIO"@"HOST" [IDENTIFIED BY "CONTRASEÑA"] [WITH GRANT OPTION];
```

Por ejemplo:

```
GRANT SELECT ON prueba.tabla1 to "usuario"@"localhost"; -> Permite hacer  
selects en la tabla "tabla1" de la base de datos "prueba" al usuario "usuario".
```

```
GRANT INSERT ON prueba.tabla2 to "usuario"@"localhost"; -> Permite hacer  
inserts en la tabla "tabla2" de la base de datos "prueba" al usuario "usuario".
```

Para revocar esos privilegios se usa la siguiente sintaxis:

```
REVOKE "PRIVILEGIO" ON "DB"."NOMBRE_TABLA" FROM "USUARIO"@"HOST" [IDENTIFIED BY "CONTRASEÑA"] [WITH GRANT OPTION];
```

Así pues, para revocar los privilegios anteriores se haría de la siguiente forma:

```
REVOKE SELECT ON prueba.tabla1 FROM "usuario"@"localhost"; -> Revoca el
privilegio de hacer selects en la tabla1 de la base de datos "prueba" al
usuario "usuario".
```

```
REVOKE INSERT ON prueba.tabla2 FROM "usuario"@"localhost"; -> Revoca el
privilegio de hacer inserts en la tabla2 de la base de datos "prueba" al
usuario "usuario".
```

Un ejemplo real:

```
MariaDB [prueba]> GRANT SELECT ON prueba.Empleados to "dparrales"@"%";
Query OK, 0 rows affected (0.004 sec)
```

```
MariaDB [prueba]> █
```

```
root@servidormariadb:/home/debian# mysql -u dparrales -p
```

Enter password:

Welcome to the MariaDB monitor. Commands end with ; or \g.

Your MariaDB connection id is 35

Server version: 10.5.12-MariaDB-0+deb11u1 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
MariaDB [(none)]> use prueba;
```

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

```
MariaDB [prueba]> select * from Empleados;
```

DNI	Nombre	Direccion	Telefono	FechaNacimiento
18232747A	Aristarco Caban Meraz	Puerta Nueva, 67	691204722	1994-05-07
90389058R	Joaquin Marrero Covas	C/ Hijuela de Lojo, 22	618385118	1997-01-28

2 rows in set (0.001 sec)

```
MariaDB [prueba]>
```

```
MariaDB [(none)]> REVOKE SELECT ON prueba.Empleados from 'dparrales'@"%";
Query OK, 0 rows affected (0.011 sec)
```

```
root@servidormariadb:/home/debian# mysql -u dparrales -p
```

Enter password:

Welcome to the MariaDB monitor. Commands end with ; or \g.

Your MariaDB connection id is 37

Server version: 10.5.12-MariaDB-0+deb11u1 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
MariaDB [(none)]> use prueba;
```

Reading table information for completion of table and column names

You can turn off this feature to get a quicker startup with -A

Database changed

```
MariaDB [prueba]> select * from Empleados;
```

ERROR 1142 (42000): SELECT command denied to user 'dparrales'@'localhost' for table 'Empleados'

```
MariaDB [prueba]>
```

### 3. Averigua si existe el concepto de rol en MySQL y señala las diferencias con los roles de ORACLE.

Desde la versión 10.0.5, existe en MariaDB el concepto de Rol.

Un rol agrupa un conjunto de privilegios. A estos roles se le pueden añadir otros roles y deben ser asignados a un usuario para que surjan efecto.

Para crear un rol se usa la siguiente sintaxis:

```
CREATE ROLE "NOMBRE_ROL";
```

```
MariaDB [(none)]> CREATE ROLE USUARIO_BASICO;  
Query OK, 0 rows affected (0.011 sec)
```

Ahora le podemos asignar los permisos que queramos;

```
GRANT "PRIVILEGIO" ON "DB"."TABLA" TO "NOMBRE_ROL";
```

```
MariaDB [(none)]> GRANT SHOW DATABASES ON *.* TO USUARIO_BASICO;  
Query OK, 0 rows affected (0.010 sec)
```

Y asignamos el rol a un usuario:

```
GRANT "NOMBRE_ROL" TO "USUARIO"@"HOST";
```

```
MariaDB [(none)]> GRANT USUARIO_BASICO TO "dparrales"@"%";  
Query OK, 0 rows affected (0.009 sec)
```

Con esto hemos asignado el rol "USUARIO\_BASICO" a mi usuario "dparrales". Sin embargo, si entramos como dicho usuario, no podremos hacer uso de dicho rol si no nos lo asignamos. En este punto hago un inciso para explicar un poco los comandos que como usuarios normales podemos usar en relación a los roles:

```
SELECT CURRENT_ROLE; -> Nos muestra el rol que tenemos asignado en este momento.
```

```
SET ROLE "NOMBRE_ROL"; -> Nos asignamos un rol (es necesario para poder usar los privilegios del rol)
```

```
SET DEFAULT ROLE "NOMBRE_ROL"; -> Nos asignamos un rol por defecto para que se nos asigne cada vez que iniciamos sesión (si no nos asignamos uno por defecto, se queda como Null)
```

Explicado esto, vamos a hacer uso del rol que nos hemos asignado antes. Como no nos hemos asignado el rol por defecto, cuando iniciemos sesión no tendremos ninguno asignado:

```

root@servidormariadb:/home/debian# mysql -u dparrales -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 39
Server version: 10.5.12-MariaDB-0+deb11u1 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| aerogeneradores |
| information_schema |
| prueba |
+-----+
3 rows in set (0.001 sec)

MariaDB [(none)]> SELECT CURRENT_ROLE;
+-----+
| CURRENT_ROLE |
+-----+
| NULL |
+-----+
1 row in set (0.000 sec)

```

Como vemos, no nos muestra todas las bases de datos que tenemos, ya que aunque el administrador nos ha asignado un rol, todavía no estamos haciendo uso de él, por lo que solo podemos ver las bases de datos sobre las que tenemos alguna clase de permiso.

```

MariaDB [(none)]> set role USUARIO_BASICO;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| aerogeneradores |
| bookmedik |
| information_schema |
| mysql |
| performance_schema |
| prueba |
+-----+
6 rows in set (0.001 sec)

MariaDB [(none)]> SELECT CURRENT_ROLE;
+-----+
| CURRENT_ROLE |
+-----+
| USUARIO_BASICO |
+-----+
1 row in set (0.000 sec)

MariaDB [(none)]>

```

Al activar el rol, ya podemos ver todas las bases de datos del servidor. También podemos crear otro rol y asignar dicho rol al rol que creamos antes:

```

MariaDB [(none)]> CREATE ROLE CREATR_TABLAS;
Query OK, 0 rows affected (0.009 sec)

MariaDB [(none)]> GRANT CREATE ON prueba.* to CREATR_TABLAS;
Query OK, 0 rows affected (0.009 sec)

MariaDB [(none)]> GRANT CREATR_TABLAS TO USUARIO_BASICO;
Query OK, 0 rows affected (0.014 sec)

```

Ahora si entramos con el usuario "dparrales" tendremos los permisos del rol "USUARIO\_BASICO" y del rol "CREATR\_TABLAS", ya que hemos añadido este último rol al anterior:

```

root@servidormariadb:/home/debian# mysql -u dparrales -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 41
Server version: 10.5.12-MariaDB-0+deb11u1 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> SELECT CURRENT_ROLE;
+-----+
| CURRENT_ROLE |      EL ROL ACTUAL      |
+-----+
| NULL         |                          |
+-----+
1 row in set (0.000 sec)

MariaDB [(none)]> set role USUARIO_BASICO;
Query OK, 0 rows affected (0.000 sec)
NOS ASIGNAMOS EL ROL USUARIO_BASICO

MariaDB [(none)]> use prueba;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
CAMBIAMOS A LA BASE DE DATOS "PRUEBA"

MariaDB [prueba]> create table tabla1 (iden integer(3));
Query OK, 0 rows affected (0.037 sec)
CREAMOS UNA TABLA, YA QUE TENEMOS EL PERMISO NECESARIO

MariaDB [prueba]> drop table tabla1;
ERROR 1142 (42000): DROP command denied to user 'dparrales'@'localhost' for table 'tabla1'
COMO NO TENEMOS PERMISO PARA ELLO, NO PODEMOS ELIMINAR LA TABLA
MariaDB [prueba]>

```

Como vemos los roles en MariaDB funcionan de forma similar a Oracle, con algunas excepciones:

- En Oracle podemos asignar contraseñas a los roles para dotarlos de más seguridad.
- También existen roles locales, externos o globales en Oracle.

Hay que mencionar que debido a que MariaDB pertenece a Oracle, cada vez se parecen más los roles entre los dos gestores de bases de datos.

## 4. Averigua si existe el concepto de perfil como conjunto de límites sobre el uso de recursos o sobre la contraseña en MySQL y señala las diferencias con los perfiles de ORACLE.

En MariaDB no existe el concepto de perfil tal y como existe en Oracle. En MariaDB, la forma de limitar el uso de recursos o la contraseña es a través de variables que se asignan a la hora de crear usuarios o al modificar dichos usuarios. Las variables referentes a los recursos que podemos modificar son las siguientes:

- **MAX\_QUERIES\_PER\_HOUR:** Número de consultas que puede ejecutar un usuario por hora (incluyendo updates).
- **MAX\_UPDATES\_PER\_HOUR:** Número de updates (no consultas) que puede ejecutar un usuario por hora.



- **MAX\_CONNECTIONS\_PER\_HOUR:** Número de conexiones que puede abrir un usuario a la hora.
- **MAX\_USER\_CONNECTIONS:** Número de conexiones simultáneas que puede abrir un usuario (si ponemos 0, no habría límite a no ser que cambiáramos otro parámetro adicional).
- **MAX\_STATEMENT\_TIME:** Tiempo en segundos que se permite como máximo a un usuario para que ejecute consultas.

Por ejemplo:

```
GRANT USAGE ON *.* TO 'someone'@'localhost' WITH
    MAX_USER_CONNECTIONS 0
    MAX_QUERIES_PER_HOUR 200;
```

Para modificar aspectos referentes a las contraseñas tenemos lo siguiente:

- **default\_password\_lifetime:** Establece el tiempo por defecto que tendrá de tiempo de vida la contraseña. Por defecto, a no ser que lo hayamos cambiado esta a 0 (no caducan). Podemos cambiar manualmente el tiempo de vida de una contraseña a la hora de crear un usuario añadiendo **PASSWORD EXPIRE INTERVAL** y el número de días que queramos.
- **disconnect\_on\_expired\_password:** Establece si una vez que ha caducado una contraseña, se permite que el usuario acceda a la base de datos en modo sandbox (con privilegios limitados) o no.

Como vemos, se diferencia bastante con Oracle en el sentido en el que Oracle permite una mayor personalización de los límites de recursos y aspectos de las contraseñas. En Oracle podemos modificar muchos más aspectos de límites de recursos y límites y restricciones con respecto a las contraseñas. También está el aspecto de que en Oracle podemos crear perfiles y después asignar dichos perfiles a los usuarios, mientras que en MariaDB debemos asignar esos límites usuario por usuario, por lo que a la larga se vuelve una tarea monumental si la base de datos tiene muchos usuarios.

## 5. Realiza consultas al diccionario de datos de MySQL para averiguar todos los privilegios que tiene un usuario concreto.

Para ello, MariaDB nos ofrece el siguiente comando, para que de forma rápida y eficaz, podamos ver todos los privilegios que posee un usuario en cualquier momento:

```
SHOW GRANTS FOR "NOMBRE_USUARIO"@"HOST"
```

```
MariaDB [(none)]> SHOW GRANTS FOR "dparrales"@'%';
+-----+
| Grants for dparrales@% |
+-----+
| GRANT `USUARIO_BASICO` TO `dparrales`@`%` |
| GRANT USAGE ON *.* TO `dparrales`@`%` IDENTIFIED BY PASSWORD '*44DC68FA32CB5CA6D691AC83FDB9B131E07E0515' |
| GRANT USAGE ON `prueba`.* TO `dparrales`@`%` WITH GRANT OPTION |
| GRANT ALL PRIVILEGES ON `aerogeneradores`.* TO `dparrales`@`%` WITH GRANT OPTION |
| GRANT DELETE ON `scott`.`dept` TO `dparrales`@`%` |
| GRANT DELETE ON `scott`.`emp` TO `dparrales`@`%` |
+-----+
6 rows in set (0.000 sec)
```

Como vemos, nos muestra los privilegios y roles que tiene asignado mi usuario. Si además queremos ver los privilegios que tienen asignado dichos roles, podemos usar el comando anterior, pero sustituyendo el nombre del usuario por el nombre del rol:

```
MariaDB [(none)]> SHOW GRANTS FOR "USUARIO_BASICO";
+-----+
| Grants for USUARIO_BASICO |
+-----+
| GRANT `CREAR_TABLAS` TO `USUARIO_BASICO` |
| GRANT SHOW DATABASES ON *.* TO `USUARIO_BASICO` |
| GRANT USAGE ON *.* TO `CREAR_TABLAS` |
| GRANT CREATE ON `prueba`.* TO `CREAR_TABLAS` |
+-----+
4 rows in set (0.000 sec)
```

## 6. Realiza consultas al diccionario de datos en MySQL para averiguar qué usuarios pueden consultar una tabla concreta.

Para ello tenemos en MariaDB la tabla `mysql.tables_priv` en la cual aparecen los privilegios que tienen los usuarios sobre las tablas de las diversas bases de datos. También podemos observar los mismos datos en la tabla `information_schema.table_privileges`:

```
MariaDB [(none)]> select * from mysql.tables_priv;
+-----+
| Host | Db | User | Table_name | Grantor | Timestamp | Table_priv | Column_priv |
+-----+
| localhost | mysql | mariadb.sys | global_priv | root@localhost | 0000-00-00 00:00:00 | Select,Delete | |
| % | scott | dparrales | emp | root@localhost | 0000-00-00 00:00:00 | Delete | |
| % | scott | dparrales | dept | root@localhost | 0000-00-00 00:00:00 | Delete | |
+-----+
3 rows in set (0.001 sec)

MariaDB [(none)]> select * from information_schema.table_privileges;
+-----+
| GRANTEE | TABLE_CATALOG | TABLE_SCHEMA | TABLE_NAME | PRIVILEGE_TYPE | IS_GRANTABLE |
+-----+
| 'mariadb.sys'@'localhost' | def | mysql | global_priv | SELECT | NO |
| 'mariadb.sys'@'localhost' | def | mysql | global_priv | DELETE | NO |
| 'dparrales'@'%' | def | scott | dept | DELETE | NO |
| 'dparrales'@'%' | def | scott | emp | DELETE | NO |
+-----+
4 rows in set (0.000 sec)

MariaDB [(none)]>
```

Esto nos muestra información de todas las tablas y los permisos que tienen los usuarios sobre ellas. Podemos concretar más la información si usamos sentencias sql:

```
select * from mysql.tables_priv where Db="scott" and Table_name="emp"; -> Nos muestra los usuarios que tienen permisos sobre la tabla "emp" de la base de datos "scott"
```

```
MariaDB [(none)]> select * from mysql.tables_priv where Db="scott" and Table_name="emp";
+-----+-----+-----+-----+-----+-----+-----+-----+
| Host | Db   | User   | Table_name | Grantor   | Timestamp           | Table_priv | Column_priv |
+-----+-----+-----+-----+-----+-----+-----+-----+
| %    | scott | dparrales | emp       | root@localhost | 0000-00-00 00:00:00 | Delete    |             |
+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set (0.001 sec)
```

## Oracle

### 1. Realiza un procedimiento llamado **PermisosdeAsobreB** que reciba dos nombres de usuario y muestre los permisos que tiene el primero de ellos sobre objetos del segundo.

Primero creamos un procedimiento que busque los privilegios que se han asignado directamente al usuario que hemos introducido:

```
CREATE OR REPLACE PROCEDURE P_BUSCAR_PERMISOS_DIRECTOS (P_USUARIO1 VARCHAR2,
P_USUARIO2 VARCHAR2)
IS
CURSOR C_PRIV_OBJ IS SELECT PRIVILEGE, TABLE_NAME
                      FROM DBA_TAB_PRIVS
                      WHERE GRANTEE = P_USUARIO1 AND
                      OWNER = P_USUARIO2;
V_REG C_PRIV_OBJ%ROWTYPE;
BEGIN
FOR V_REG IN C_PRIV_OBJ LOOP
    DBMS_OUTPUT.PUT_LINE('EL USUARIO QUE HAS INTRODUCIDO TIENE PRIVILEGIO DE
    '||V_REG.PRIVILEGE||' SOBRE LA TABLA '||V_REG.TABLE_NAME||' DEL USUARIO
    '||P_USUARIO2);
END LOOP;
END P_BUSCAR_PERMISOS_DIRECTOS;
/
```

A continuación creamos otro procedimiento que llama al anterior para que busque entre los roles que han sido asignados a dicho usuario si hay privilegios que nos interesen. Lo hacemos recursivo para que de esta forma también busque si hay roles asignados a roles con privilegios que nos interesen:

```
CREATE OR REPLACE PROCEDURE P_BUSCAR_ROLES_POR_USUARIO (P_USUARIO1 VARCHAR2,
P_USUARIO2 VARCHAR2)
IS
CURSOR C_ROLES IS SELECT GRANTED_ROLE FROM DBA_ROLE_PRIVS WHERE GRANTEE =
P_USUARIO1;
V_REG C_ROLES%ROWTYPE;
BEGIN
FOR V_REG IN C_ROLES LOOP
    P_BUSCAR_PERMISOS_DIRECTOS(V_REG.GRANTED_ROLE,P_USUARIO2);
    P_BUSCAR_ROLES_POR_USUARIO(V_REG.GRANTED_ROLE,P_USUARIO2);
    P_BUSCAR_PRIVILEGIOS_SISTEMA_OBJETOS(V_REG.GRANTED_ROLE);
END LOOP;
```

```
END P_BUSCAR_ROLES_POR_USUARIO;  
/
```

Ahora vamos a crear un procedimiento que busque los privilegios del sistema que permitan modificar los objetos de cualquier esquema, ya que de esta forma un usuario también tendría privilegios sobre los objetos de otro usuario:

```
CREATE OR REPLACE PROCEDURE P_BUSCAR_PRIVILEGIOS_SISTEMA_OBJETOS (P_USUARIO1  
VARCHA2)  
IS  
CURSOR C_SYS_PRIVS IS SELECT PRIVILEGE FROM DBA_SYS_PRIVS WHERE GRANTEE =  
P_USUARIO1;  
V_REG C_SYS_PRIVS%ROWTYPE;  
BEGIN  
FOR V_REG IN C_SYS_PRIVS LOOP  
CASE V_REG.PRIVILEGE  
WHEN 'SELECT ANY TABLE' THEN  
DBMS_OUTPUT.PUT_LINE('Tiene el privilegio de Select en cualquier  
tabla');  
WHEN 'UPDATE ANY TABLE' THEN  
DBMS_OUTPUT.PUT_LINE('Tiene el privilegio de Update en cualquier  
tabla');  
WHEN 'INSERT ANY TABLE' THEN  
DBMS_OUTPUT.PUT_LINE('Tiene el privilegio de Insert en cualquier  
tabla');  
WHEN 'DELETE ANY TABLE' THEN  
DBMS_OUTPUT.PUT_LINE('Tiene el privilegio de Delete en cualquier  
tabla');  
WHEN 'DROP ANY TABLE' THEN  
DBMS_OUTPUT.PUT_LINE('Tiene el privilegio de Drop en cualquier  
tabla');  
ELSE  
DBMS_OUTPUT.PUT_LINE('');  
END CASE;  
END LOOP;  
END P_BUSCAR_PRIVILEGIOS_SISTEMA_OBJETOS;  
/
```

Creamos al final el procedimiento final que llama a los anteriores:

```
CREATE OR REPLACE PROCEDURE PERMISOSDEASOBREB (P_USUARIO1 VARCHA2, P_USUARIO2  
VARCHA2)  
IS  
BEGIN  
P_BUSCAR_PERMISOS_DIRECTOS(P_USUARIO1,P_USUARIO2);  
P_BUSCAR_ROLES_POR_USUARIO(P_USUARIO1,P_USUARIO2);  
P_BUSCAR_PRIVILEGIOS_SISTEMA_OBJETOS(P_USUARIO1);  
END PERMISOSDEASOBREB;  
/
```

Podemos ver que nos muestra todos los privilegios del usuario "C##BECARIO" sobre los objetos del usuario "C##SCOTT" ejecutando la siguiente sentencia:

```
EXEC PERMISOSDEASOBREB('C##BECARIO','C##SCOTT');
```

```
SQL> EXEC PERMISOSDEASOBREB('C##BECARIO','C##SCOTT');
EL USUARIO QUE HAS INTRODUCIDO TIENE PRIVILEGIO DE INSERT SOBRE LA TABLA EMP DEL
USUARIO C##SCOTT
EL USUARIO QUE HAS INTRODUCIDO TIENE PRIVILEGIO DE DELETE SOBRE LA TABLA DEPT
DEL USUARIO C##SCOTT
EL USUARIO QUE HAS INTRODUCIDO TIENE PRIVILEGIO DE INSERT SOBRE LA TABLA DEPT
DEL USUARIO C##SCOTT
EL USUARIO QUE HAS INTRODUCIDO TIENE PRIVILEGIO DE SELECT SOBRE LA TABLA DEPT
DEL USUARIO C##SCOTT
EL USUARIO QUE HAS INTRODUCIDO TIENE PRIVILEGIO DE UPDATE SOBRE LA TABLA DEPT
DEL USUARIO C##SCOTT
Tiene el privilegio de Drop en cualquier tabla
Tiene el privilegio de Delete en cualquier tabla
Tiene el privilegio de Update en cualquier tabla
EL USUARIO QUE HAS INTRODUCIDO TIENE PRIVILEGIO DE DELETE SOBRE LA TABLA EMP DEL
USUARIO C##SCOTT
EL USUARIO QUE HAS INTRODUCIDO TIENE PRIVILEGIO DE UPDATE SOBRE LA TABLA EMP DEL
USUARIO C##SCOTT
Tiene el privilegio de Select en cualquier tabla

Procedimiento PL/SQL terminado correctamente.
```

MUESTRA LOS PRIVILEGIOS TANTO DEL SISTEMA COMO SOBRE LOS OBJETOS DE LOS USUARIOS QUE LE HE PASADO COMO PARÁMETROS.

ESTOS PRIVILEGIOS ESTÁN ASIGNADOS DIRECTAMENTE AL USUARIO O A OTROS ROLES, COMPROBADOS HASTA UN TERCER NIVEL DE RECURSIVIDAD.

## 2. Realiza un procedimiento llamado MostrarInfoPerfil que reciba el nombre de un perfil y muestre su composición y los usuarios que lo tienen asignado.

Primero vamos a crear una función que nos avise de si el perfil existe o no:

```
CREATE OR REPLACE FUNCTION F_EXISTE_PERFIL (P_NOMBRE DBA_PROFILES.PROFILE%TYPE)
RETURN NUMBER
IS
V_EXISTE NUMBER:=0;
BEGIN
SELECT COUNT(*) INTO V_EXISTE FROM DBA_PROFILES WHERE PROFILE=P_NOMBRE;
RETURN V_EXISTE;
END F_EXISTE_PERFIL;
/
```

Ahora creamos un procedimiento que muestre la composición de dicho perfil:

```
CREATE OR REPLACE PROCEDURE P_MOSTRAR_PERFIL_INFO (P_NOMBRE
DBA_PROFILES.PROFILE%TYPE)
IS
CURSOR C_RECURSOS_PERFIL IS SELECT RESOURCE_NAME, LIMIT FROM DBA_PROFILES WHERE
PROFILE = P_NOMBRE;
V_REG C_RECURSOS_PERFIL%ROWTYPE;
BEGIN
DBMS_OUTPUT.PUT_LINE('Información del perfil '||P_NOMBRE);
DBMS_OUTPUT.PUT_LINE(CHR(9));
FOR V_REG IN C_RECURSOS_PERFIL LOOP
DBMS_OUTPUT.PUT_LINE(RPAD('Recurso:',10,' ')||RPAD(V_REG.RESOURCE_NAME,30,'
')||LPAD('Límites:',10,' ')||V_REG.LIMIT);
END LOOP;
END P_MOSTRAR_PERFIL_INFO;
/
```

A continuación crearemos otro procedimiento que nos indique que usuarios están haciendo uso de dicho perfil:

```
CREATE OR REPLACE PROCEDURE P_MOSTRAR_USUARIOS_POR_PERFIL (P_NOMBRE
DBA_PROFILES.PROFILE%TYPE)
IS
CURSOR C_USUARIO_PERFIL IS SELECT USERNAME FROM DBA_USERS WHERE PROFILE =
P_NOMBRE;
V_REG C_USUARIO_PERFIL%ROWTYPE;
BEGIN
DBMS_OUTPUT.PUT_LINE(CHR(9));
DBMS_OUTPUT.PUT_LINE('Usuarios con ese perfil: ');
DBMS_OUTPUT.PUT_LINE(CHR(9));
FOR V_REG IN C_USUARIO_PERFIL LOOP
    DBMS_OUTPUT.PUT_LINE(LPAD(V_REG.USERNAME,15,' '));
END LOOP;
END P_MOSTRAR_USUARIOS_POR_PERFIL;
/
```

Finalmente, ya podemos crear el procedimiento principal que llamará a todos los anteriores que hemos creado:

```
CREATE OR REPLACE PROCEDURE MOSTRARINFOPERFIL (P_NOMBRE
DBA_PROFILES.PROFILE%TYPE)
IS
V_ERROR NUMBER:=0;
BEGIN
V_ERROR:=F_EXISTE_PERFIL(P_NOMBRE);
IF V_ERROR > 0 THEN
    P_MOSTRAR_PERFIL_INFO(P_NOMBRE);
    P_MOSTRAR_USUARIOS_POR_PERFIL(P_NOMBRE);
ELSE
    RAISE_APPLICATION_ERROR(-20036,'Ese perfil no existe');
END IF;
END MOSTRARINFOPERFIL;
/
```

Ahora comprobamos si este procedimiento funciona:

```
EXEC MOSTRARINFOPERFIL('DEFAULT');
```

```
SQL> EXEC MOSTRARINFOPERFIL('DEFAULT');
Informaci??n del perfil DEFAULT
Recurso:  COMPOSITE_LIMIT           L??mites:UNLIMITED
Recurso:  SESSIONS_PER_USER         L??mites:UNLIMITED
Recurso:  CPU_PER_SESSION           L??mites:UNLIMITED
Recurso:  CPU_PER_CALL               L??mites:UNLIMITED
Recurso:  LOGICAL_READS_PER_SESSION L??mites:UNLIMITED
Recurso:  LOGICAL_READS_PER_CALL    L??mites:UNLIMITED
Recurso:  IDLE TIME                  L??mites:UNLIMITED
Recurso:  CONNECT_TIME              L??mites:UNLIMITED
Recurso:  PRIVATE_SGA                L??mites:UNLIMITED
Recurso:  FAILED_LOGIN_ATTEMPTS     L??mites:10
Recurso:  PASSWORD_LIFE_TIME         L??mites:180
Recurso:  PASSWORD_REUSE_TIME        L??mites:UNLIMITED
Recurso:  PASSWORD_REUSE_MAX         L??mites:UNLIMITED
Recurso:  PASSWORD_VERIFY_FUNCTION   L??mites:NULL
Recurso:  PASSWORD_LOCK_TIME         L??mites:1
Recurso:  PASSWORD_GRACE_TIME        L??mites:7
Recurso:  INACTIVE_ACCOUNT_TIME     L??mites:UNLIMITED
```

Usuarios con ese perfil:

```
SYS
SYSTEM
XS$NULL
DBSNMP
APPQOSSYS
GSMCATUSER
MDDATA
C##DPARRALES
DBSFUSER
SYSBACKUP
REMOTE_SCHEDULE
C##BECARIO
GGSYS
ANONYMOUS
GSMUSER
SYSRAC
GSMROOTUSER
DANIM
CTXSYS
OJVMSYS
DVSYS
DVF
SI_INFORMTN_SCH
C##DPARRALES1
C##MANOLO
AUDSYS
```

```
EXEC MOSTRARINFOPERFIL('C##CONTRA_LIMITE');
```

```
SQL> EXEC MOSTRARINFOPERFIL('C##CONTRA_LIMITE');
Informaci??n del perfil C##CONTRA_LIMITE
Recurso: COMPOSITE_LIMIT L??mites:DEFAULT
Recurso: SESSIONS_PER_USER L??mites:DEFAULT
Recurso: CPU_PER_SESSION L??mites:DEFAULT
Recurso: CPU_PER_CALL L??mites:DEFAULT
Recurso: LOGICAL_READS_PER_SESSION L??mites:DEFAULT
Recurso: LOGICAL_READS_PER_CALL L??mites:DEFAULT
Recurso: IDLE_TIME L??mites:DEFAULT
Recurso: CONNECT_TIME L??mites:DEFAULT
Recurso: PRIVATE_SGA L??mites:DEFAULT
Recurso: FAILED_LOGIN_ATTEMPTS L??mites:5
Recurso: PASSWORD_LIFE_TIME L??mites:DEFAULT
Recurso: PASSWORD_REUSE_TIME L??mites:DEFAULT
Recurso: PASSWORD_REUSE_MAX L??mites:DEFAULT
Recurso: PASSWORD_VERIFY_FUNCTION L??mites:DEFAULT
Recurso: PASSWORD_LOCK_TIME L??mites:DEFAULT
Recurso: PASSWORD_GRACE_TIME L??mites:DEFAULT
Recurso: INACTIVE_ACCOUNT_TIME L??mites:DEFAULT

Usuarios con ese perfil:
C##SCOTT

Procedimiento PL/SQL terminado correctamente.
```

Ahora comprobemos si funciona el control de errores:

```
EXEC MOSTRARINFOPERFIL('C##NO_EXISTE');
```

```
SQL> EXEC MOSTRARINFOPERFIL('C##NO_EXISTE');
BEGIN MOSTRARINFOPERFIL('C##NO_EXISTE'); END;

*
ERROR en línea 1:
ORA-20036: Ese perfil no existe
ORA-06512: en "SYS.MOSTRARINFOPERFIL", línea 11
ORA-06512: en línea 1

SQL> █
```

Como vemos, nos indica de forma adecuada el error, así que podemos dar por concluido este procedimiento.

## Carlos (MongoDB y ORACLE)

### Mongo

**1. Averigua si existe la posibilidad en MongoDB de limitar el acceso de un usuario a los datos de una colección determinada.**



Como en la primera práctica que se hizo en este curso se creó una base de datos en *MongoDB*, voy a hacer uso de esa misma para ilustrar los ejemplos.

El usuario que tenía creado, llamado *prueba*, tiene privilegios de **administrador**, y por tanto es el que se usará para los ejercicios propuestos. La base de datos, llamada *empresa* nos servirá de ejemplo.

```
crivero@crivero:~$ mongosh -u prueba 192.168.122.51/admin
Enter password: *****
Current Mongosh Log ID: 61c98d7098d46f1afd792cb0
Connecting to:      mongodb://192.168.122.51:27017/admin?directConnection=true
Using MongoDB:      5.0.3
Using Mongosh:      1.1.4

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

admin> use empresa;
switched to db empresa
empresa> show collections;
departamentos
edificios
empleados
empresa>
```

ESTAS SON LAS 3 COLECCIONES QUE TENEMOS DISPONIBLES DENTRO DE LA BASE DE DATOS.

*MongoDB* tiene unos roles predefinidos, que explicaremos más adelante, y además podemos crear roles personalizados, que es lo que haremos para dar respuesta a este ejercicio. Hay que usar la base de datos donde se va a crear el rol.

A este rol lo llamaremos, "*becarios*" y tiene la siguiente estructura:

```
db.createRole(
  {role:"becarios",
   privileges: [
     {
       resource:{
         db:"empresa",
         collection:"empleados"
       },
       actions: ["find"]
     },
   ],
   roles: [ ]
  }
);
```

Lo que hemos creado es un rol que sólo permita leer los datos de la colección *empleados* de la base de datos *empresa*.

```

mongosh mongodb://192.168.122.51:27017/admin?directConnection=true 80x
empresa> db.createRole(
...     {role:"becarios",
.....     privileges: [
.....         {
.....             resource:{
.....                 db:"empresa",
.....                 collection:"empleados"
.....             },
.....             actions: ["find"]
.....         }
.....     ],
.....     roles: [ ]
..... }
... );
{ ok: 1 }
empresa>

```

Tambien se puede borrar un rol de una base de datos, dentro de la misma hay que usar el siguiente comando

```
db.dropRole("nombre_rol");
```

Ahora vamos a crear un usuario "novato" para darle este permiso

```

db.createUser(
  {
    user:"novato",
    pwd:"12345",

    roles: [
      {
        role:"becarios",
        db:"empresa"}
    ]
  }
);

```

```

empresa> db.createUser(
...     {
.....         user:"novato",
.....         pwd:"12345",
.....         roles: [
.....             {
.....                 role:"becarios",
.....                 db:"empresa"}
.....         ]
.....     }
... );
{ ok: 1 }

```

Mostramos la información del usuario que acabamos de crear

```
db.getUser("novato");
```

```
empresa> db.getUser("novato");
{
  _id: 'empresa.novato',
  userId: UUID("1f6569ff-04a4-4d9f-b8d6-335488c75e14"),
  user: 'novato',
  db: 'empresa',
  roles: [ { role: 'becarios', db: 'empresa' } ],
  mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
}
empresa>
```

La herramienta actualizada de Mongosh, permite mostrar los datos como si añadiéramos al final del comando `.pretty()`.

Entramos con el nuevo usuario y comprobamos que podemos ver los datos de la colección *empleados*

```
db.empleados.find();
```

```
crivero@crivero:~$
crivero@crivero:~$ mongosh -u novato 192.168.122.51/empresa
Enter password: ****
Current Mongosh Log ID: 61c99eb1dacf80eb1e00aea6
Connecting to:      mongodb://192.168.122.51:27017/empresa?directConnection=
true
Using MongoDB:      5.0.3
Using Mongosh:      1.1.4

For mongosh info see: https://docs.mongodb.com/mongosh-shell/

empresa> db.empleados.find();          ESTO PUEDE HACERLO
[
  {
    _id: ObjectId("6152da00565b421d3480279a"),
    DNI: '18569841L',
    Nombre: 'Arturo',
    Apellidos: 'Montes Alvarado',
    Salario: '2567',
    ID_Dept: '10'
  },
  {
    _id: ObjectId("6152da01565b421d3480279b"),
    DNI: '18569842L',
    Nombre: 'María',
    Apellidos: 'Montes Alvarado'
```

Pero no nos deja ver los datos de las otras colecciones o insertar datos en cualquiera de

ellas

```
mongosh mongodb://192.168.122.51:27017/empresa?directConnection=true 80x30
empresa> db.edificios.find();
MongoServerError: not authorized on empresa to execute command { find: "edificios", filter: {}, lsid: { id: UUID("795564cd-0702-4090-8062-21e2ae9f0537") }, $db: "empresa" }
empresa> db.departamentos.find();
MongoServerError: not authorized on empresa to execute command { find: "departamentos", filter: {}, lsid: { id: UUID("795564cd-0702-4090-8062-21e2ae9f0537") }, $db: "empresa" }
empresa> 
```

NO PERMITE VER DATOS DE LAS OTRAS COLECCIONES

```
db.empleados.insert({DNI:'18569853L',Nombre:'Rodrigo',Apellidos:'Pilatos Rojas',Salario:'2567',ID_Dept:'10'});
```

```
mongosh mongodb://192.168.122.51:27017/empresa?directConnection=true 80x30
empresa> db.empleados.insert({DNI:'18569853L',Nombre:'Rodrigo',Apellidos:'Pilatos Rojas',Salario:'2567',ID_Dept:'10'});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
Uncaught:
MongoBulkWriteError: not authorized on empresa to execute command { insert: "empleados", documents: [ { DNI: "18569853L", Nombre: "Rodrigo", Apellidos: "Pilatos Rojas", Salario: "2567", ID_Dept: "10", _id: ObjectId('61c9a1ec8fc8d651282a9611') } ], ordered: true, lsid: { id: UUID("795564cd-0702-4090-8062-21e2ae9f0537") }, $db: "empresa" }
Result: BulkWriteResult {
  result: {
    ok: 1,
    writeErrors: [],
    writeConcernErrors: [],
    insertedIds: [ { index: 0, _id: ObjectId("61c9a1ec8fc8d651282a9611") } ],
    nInserted: 0,
    nUpserted: 0,
    nMatched: 0,
    nModified: 0,
    nRemoved: 0,
    upserted: []
  }
}
empresa>
```

## 2. Averigua si en MongoDB existe el concepto de privilegio del sistema y muestra las diferencias más importantes con ORACLE.

En Oracle los privilegios de sistema son permisos para realizar ciertas operaciones en la base de datos. Hay muchos como, *create session*, *create user*, *create role*, etc.

En MongoDB no existen como tal, nuevamente, nos referimos a los roles que otorgan una operatividad específica sobre una o varias bases de datos o colecciones; incluso clústeres. Los roles son un conjunto de acciones que se le otorgan a un usuario y este pueden heredar otras acciones de uno o varios roles.

Hay dos tipos de roles:

- *Roles predefinidos*, estos son los que vienen por defecto en este sistema gestor. Más adelante se verán con más detalle.
- *Roles definidos por el usuario*, como vimos en el primer caso que sirven para acotar las acciones que un usuario puede realizar, a criterio del administrador.

A efectos prácticos, ambos sistemas gestores pueden asignar privilegios a los usuarios según interese; la mayor diferencia es que *Oracle* los asigna individualmente, porque existen así definidos, y se aleja de la necesidad de asignar un rol que si necesita *Mongodb*. Como hemos visto se puede asignar un único "privilegio" a un usuario, pero bajo el paraguas de un rol creado con ese único privilegio.

### 3. Explica los roles por defecto que incorpora MongoDB y como se asignan a los usuarios.

*MongoDB* tiene los siguientes roles integrados:

#### Usuarios de la base de datos

- **read**, permite leer todas las colecciones que no son del sistema.
- **readWrite**, puede leer y modificar todas las colecciones que no sean del sistema.

#### Gestión de una base de datos concreta.

- **dbOwner**, el propietario de la base de datos tiene todos los permisos sobre esa base de datos.
- **dbAdmin**, permite operaciones de administracion de algunos objetos, pero sin permisos de lectura y escritura de la base de datos.
- **userAdmin**, crea y modifica tanto usuarios como roles dentro de esa base de datos y puede dar autoridad a otros usuarios.

#### Gestión de clústeres

- **clusterAdmin**, es la función de gestión más alta de los clústeres. Es una mezcla de *clusterManager*, *clusterMonitor*, *hostManager* y *dropDatabase*.
- **clusterManager**, puede supervisar y administrar conjuntos de replicación y clústeres. Los usuarios que tengan este rol pueden operar bases de datos locales.
- **clusterMonitor**, da privilegios de supervisión sólo de clústeres y conjuntos de réplicas.
- **hostManager**, permite la autoridad para monitorear y administrar el servidor, incluido el nodo de apagado, *"logrotate"*, *repairDatabase*, etc.

#### Para todas las bases de datos

- **readAnyDatabase**, permisos de lectura para todas las bases de datos. Esto no incluye la base de datos aplicadas al clúster.
- **readWriteAnyDatabase**, igual que el anterior pero permitiendo la modificación de los datos de las mismas.

- **userAdminAnyDatabase**, permisos como los de *userAdmin* pero para todas las bases de datos, excepto, claro está, las bases de datos en clústeres.
- **dbAdminAnyDatabase**, permisos para todas las bases de datos como si fuera *dbAdmin*, con la misma exclusión que el resto de este apartado.

## Superadministrador

- **root**, combina *readWriteAnyDatabase*, *dbAdminAnyDatabase*, *userAdminAnyDatabase*, *clusterAdmin* y *restore*.

## Otros

- **backup**, privilegios para hacer copias de respaldo sobre una base de datos.
- **restore**, permisos para recuperar los *backups*.

Como vimos al principio estos roles se pueden asignar al crear el usuario, pero también se pueden asignar más adelante (como Administrador).

```
db.grantRolesToUser(
  "novato",
  [
    {role:"readWrite",
     db:"empresa"}
  ]
);
```

The screenshot shows a terminal window with the title bar "mongosh mongod://192.168.122.51:27017/admin?directConnection=true 80". The terminal content is as follows:

```
empresa> db.grantRolesToUser(
...   "novato",
...   [
...     {role:"readWrite",
...       db:"empresa"}
...   ]
... );
{ ok: 1 }
empresa> db.getUser("novato");
{
  _id: 'empresa.novato',
  _userId: UUID("1f6569ff-04a4-4d9f-b8d6-335488c75e14"),
  user: 'novato',
  db: 'empresa',
  roles: [
    { role: 'readWrite', db: 'empresa' },
    { role: 'becarios', db: 'empresa' }
  ],
  mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
}
empresa> █
```

On the right side of the terminal, there is a red text annotation: "DAMOS EL ROL DE LEER Y MODIFICAR LA BASE DE DATOS EMPRESA".

También podemos quitárselo

```
db.revokeRolesFromUser(
  "novato",
  [
    {role:"becarios",
     db:"empresa"}
  ]
);
```

```

mongosh mongodb://192.168.122.51:27017/admin?directConnection=true 80x30
empresa> db.revokeRolesFromUser(
...   "novato",
...   [
...     {role:"becarios",
...       db:"empresa"}
...   ]
... );
{ ok: 1 }
empresa> db.getUser("novato");
{
  _id: 'empresa.novato',
  userId: UUID("1f6569ff-04a4-4d9f-b8d6-335488c75e14"),
  user: 'novato',
  db: 'empresa',
  roles: [ { role: 'readWrite', db: 'empresa' } ],
  mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
}
empresa>
```

LE QUITAMOS EL ROL QUE CREAMOS EN EL PRIMER EJERCICIO.

SOLO APARECE EL ÚLTIMO QUE LE HEMOS CONCEDIDO

#### 4. Explica como puede consultarse el diccionario de datos de MongoDB para saber que roles han sido concedidos a un usuario y qué privilegios incluyen.

En ejercicios anteriores hemos podido ver información del usuario con el comando `db.getUser()` y este muestra los roles que tiene asignados y por tanto ya hemos respondido a parte de lo que se nos plantea. Pero si queremos ver lo que contiene un rol en específico, debemos hacer uso de otros comandos, se ha creado un rol predefinido más para ilustrar un ejemplo.

```
# Para roles predefinidos de MongoDB

db.getRole("read", {showPrivileges: true} );

# Para roles definidos por el usuario

db.system.roles.find();
```





```

mongosh mongodb://192.168.122.51:27017/admin?directConnection=true
admin> db.getRole( "read", {showPrivileges: true} );
{
  db: 'admin',
  role: 'read',
  roles: [],
  privileges: [
    {
      resource: { db: 'admin', collection: '' },
      actions: [
        'changeStream',
        'collStats',
        'dbHash',
        'dbStats',
        'find',
        'killCursors',
        'listCollections',
        'listIndexes',
        'planCacheRead'
      ]
    },
    {
      resource: { db: 'admin', collection: 'system.js' },
      actions: [
        'changeStream',
        'collStats',
        'dbHash',
        'dbStats',
        'find',
        'killCursors',
        'listCollections',
        'listIndexes',
        'planCacheRead'
      ]
    }
  ],
  inheritedRoles: [],
  inheritedPrivileges: [
    {
      resource: { db: 'admin', collection: '' },
      actions: [

```

```

mongosh mongodb://192.168.122.51:27017/admin?directConnection=true 80x39
admin> db.system.roles.find();
[
  {
    _id: 'empresa.becarios',
    role: 'becarios',
    db: 'empresa',
    privileges: [
      {
        resource: { db: 'empresa', collection: 'empleados' },
        actions: [ 'find' ]
      }
    ],
    roles: []
  },
  {
    _id: 'empresa.prueba',
    role: 'prueba',
    db: 'empresa',
    privileges: [
      {
        resource: { db: 'empresa', collection: 'empleados' },
        actions: [ 'find' ]
      }
    ],
    roles: []
  }
]

```

```
rol: 'prueba',
db: 'empresa',
privileges: [
  {
    resource: { db: 'empresa', collection: '' },
    actions: [ 'find', 'insert', 'update' ]
  }
],
roles: []
}
]
admin>
```

## Oracle

1. Realiza un procedimiento llamado **MostrarObjetosAccesibles** que reciba un nombre de usuario y muestre todos los objetos a los que tiene acceso.
2. Realiza un procedimiento que reciba un nombre de usuario, un privilegio y un objeto y nos muestre el mensaje 'SI, DIRECTO' si el usuario tiene ese privilegio sobre objeto concedido directamente, 'SI, POR ROL' si el usuario lo tiene en alguno de los roles que tiene concedidos y un 'NO' si el usuario no tiene dicho privilegio. **NO FUNCIONA**

Modificar permanentemente el *set serveroutput*

```
cd $ORACLE_HOME/sqlplus/admin
nano glogin.sql
```

- Escribir dentro

```
sql SET SERVEROUTPUT ON;
```

```
GNU nano 2.9.8                                glogin.sql                                Modificado
--
-- Copyright (c) 1988, 2005, Oracle. All Rights Reserved.
--
-- NAME
--   glogin.sql
--
-- DESCRIPTION
--   SQL*Plus global login "site profile" file
--
--   Add any SQL*Plus commands here that are to be executed when a
--   user starts SQL*Plus, or uses the SQL*Plus CONNECT command.
--
-- USAGE
--   This script is automatically run
--
set serveroutput on;
```

Cada vez que se acceda se ejecutará este script.

Vamos a ver las vistas que muestran información sobre privilegios:

VISTA | INFORMACIÓN

---|---

DBA\_SYS\_PRIVS | Privilegios de sistema asignados a usuarios y roles

DBA\_TAB\_PRIVS | Lista de todos los privilegios de todos los objetos de la base de datos

DBA\_COL\_PRIVS | Lista de todos los privilegios aplicados a columnas de la base de datos

SESSION\_PRIVS | Privilegios en activo para el usuario y sesión actuales

USER\_SYS\_PRIVS | Privilegios de sistema asignados al usuario

USER\_TAB\_PRIVS\_MADE | Privilegios de objeto asignados a los objetos del usuario actual

USER\_TAB\_PRIVS\_RECD | Privilegios de objeto (de otros usuarios) concedidos al usuario actual.

USER\_COL\_PRIVS\_MADE | Privilegios de objeto asignados a columnas de objetos del usuario actual

USER\_COL\_PRIVS\_RECD | Privilegios asignados a columnas de objetos (de otros usuarios) y que son concedidos al usuario actual.

Las vistas para examinar los roles son:

VISTA	INFORMACIÓN
DBA_ROLES	Muestra todos los roles de la base de datos
DBA_ROLES_PRIVS	Roles asignados a los usuarios
ROLE_ROLE_PRIVS	Roles asignados a otros roles
ROLE_SYS_PRIVS	Privilegios de sistema asignado a roles
ROLE_TAB_PRIVS	Privilegios de objeto concedidos a roles
SESSION_ROLES	Roles en activo para el usuario actual.

PROCEDIMIENTO PRINCIPAL

```

CREATE OR REPLACE PROCEDURE CONCESION_PRIVILEGIOS(P_USUARIO IN VARCHAR2,
P_PRIVILEGIO VARCHAR2, P_OBJETO VARCHAR2)
IS
    v_aux NUMBER:=0;
BEGIN
    v_aux:= COMPROBAR_PRIVILEGIO(P_USUARIO,P_PRIVILEGIO,P_OBJETO);
    CASE v_aux
        WHEN 0 THEN
            IF v_aux=0 THEN

VER_ROL_DIRECTO(P_USUARIO,P_PRIVILEGIO,P_OBJETO);

            END IF;
        ELSE
            imprimir('SI, DIRECTO');
    END CASE;
END CONCESION_PRIVILEGIOS;
/

```

## FUNCIONES/PROCEDIMIENTOS SECUNDARIOS

```

CREATE OR REPLACE FUNCTION COMPROBAR_PRIVILEGIO(USUARIO IN VARCHAR2, PRIVILEGIO
VARCHAR2, OBJETO VARCHAR2)
RETURN NUMBER
IS
    v_numero NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_numero FROM DBA_TAB_PRIVS
    WHERE GRANTEE=UPPER(USUARIO) AND
    PRIVILEGE=UPPER(PRIVILEGIO) AND
    TABLE_NAME=UPPER(OBJETO);
    RETURN v_numero;
END COMPROBAR_PRIVILEGIO;
/

CREATE OR REPLACE PROCEDURE IMPRIMIR(Respuesta IN VARCHAR2)
IS
BEGIN
    DBMS_OUTPUT.PUT_LINE(LPAD('#',10,'#'));
    DBMS_OUTPUT.PUT_LINE(Respuesta);
    DBMS_OUTPUT.PUT_LINE(LPAD('#',10,'#'));
END IMPRIMIR;
/

CREATE OR REPLACE FUNCTION BUSCAR_SUBROLES (p_nombrerol IN
ROLE_TAB_PRIVS.ROLE%TYPE, p_objeto IN ROLE_TAB_PRIVS.TABLE_NAME%TYPE,
p_privilegio IN ROLE_TAB_PRIVS.PRIVILEGE%TYPE)
RETURN NUMBER
IS
    v_numero NUMBER := 0;
BEGIN
    SELECT COUNT(*) INTO v_numero FROM ROLE_TAB_PRIVS WHERE

```

```

ROLE=p_nombrerol and
    TABLE_NAME=p_objeto AND PRIVILEGE=p_privilegio;
    RETURN v_numero;
END BUSCAR_SUBROLES;
/

CREATE OR REPLACE PROCEDURE VER_SUBROL (P_ROL IN
ROLE_ROLE_PRIVS.GRANTED_ROLE%TYPE, P_OBJETO IN VARCHAR2, P_PRIVILEGIO IN
VARCHAR2)
IS
    v_resp VARCHAR2(20);
    v_numero NUMBER;
    cursor c_subrol
    IS
        SELECT GRANTED_ROLE FROM ROLE_ROLE_PRIVS START WITH ROLE=P_ROL CONNECT
BY PRIOR GRANTED_ROLE=ROLE;
BEGIN
    FOR I IN c_subrol LOOP
        v_numero:=BUSCAR_SUBROLES(I.GRANTED_ROLE, P_OBJETO,
P_PRIVILEGIO);
    END LOOP;
    IF v_numero=0 THEN
        IMPRIMIR('NO');
    ELSE
        IMPRIMIR('SI, POR ROL');
    END IF;
END VER_SUBROL;
/

CREATE OR REPLACE PROCEDURE VER_ROL_DIRECTO (P_USUARIO IN
DBA_ROLE_PRIVS.GRANTEE%TYPE, P_OBJETO IN VARCHAR2, P_PRIVILEGIO IN VARCHAR2)
IS
    cursor c_rol
    IS
        SELECT GRANTED_ROLE FROM DBA_ROLE_PRIVS WHERE GRANTEE='P_USUARIO';
BEGIN
    FOR I IN c_rol LOOP
        VER_SUBROL(I.GRANTED_ROLE, P_OBJETO, P_PRIVILEGIO);
    END LOOP;
END VER_ROL_DIRECTO;
/

```

## PRUEBAS (CON OTRO TIPO DE FUNCIONES, QUE NO FUNCIONABA CON ROL QUE RECIBE UN ROL)

Damos privilegio directo (Debe dar, 'SI, DIRECTO')

```
GRANT INSERT ON C##CRIVMAR.EMP TO C##BECARIO WITH GRANT OPTION;
```

Comprobamos:

```
exec concesion_privilegios('c##becario','insert','emp');
```

```
oracle@oracledb1:~ 80x24
SQL> exec concesion_privilegios('c##becario','insert','emp');
#####
SI, DIRECTO
#####

Procedimiento PL/SQL terminado correctamente.

SQL>
```

Creamos un rol que daremos a c##becario2 (Debe dar, 'SI, POR ROL')

```
CREATE ROLE C##ROL_PRUEBA;

GRANT select, update, alter ON C##CRIVMAR.DEPT TO C##ROL_PRUEBA;

GRANT C##ROL_PRUEBA TO C##BECARIO2;
```

Comprobamos:

```
exec concesion_privilegios('c##becario2','update','dept');
```

```
oracle@oracledb1:~ 80x24
SQL> exec concesion_privilegios('c##becario2','update','dept');
#####
SI, POR ROL
#####

Procedimiento PL/SQL terminado correctamente.

SQL>
```

Eliminamos el rol anterior (Debe dar, 'NO')

```
DROP ROLE C##ROL_PRUEBA;
```

Comprobamos:

```
exec concesion_privilegios('c##becario2','update','dept');
```

```
oracle@oracledb1:~ 80x24
SQL> DROP ROLE C##ROL_PRUEBA;

Rol borrado.

SQL> exec concesion_privilegios('c##becario2','update','dept');
#####
NO
#####

Procedimiento PL/SQL terminado correctamente.

SQL>
```

## CREAR ROLES

```
CREATE ROLE C##ROL_A;
CREATE ROLE C##ROL_B;
CREATE ROLE C##ROL_C;
CREATE ROLE C##ROL_D;
CREATE ROLE C##ROL_E;
```

## ASIGNAR PRIVILEGIOS Y ROLES A ROL\_A

```
GRANT update ON C##CRIVMAR.EMP TO C##ROL_A;
GRANT C##ROL_B TO C##ROL_A;
GRANT C##ROL_C TO C##ROL_A;
```

## ASIGNAR PRIVILEGIOS A ROL\_B

```
GRANT alter ON C##CRIVMAR.DEPT TO C##ROL_B;
```

## ASIGNAR PRIVILEGIOS Y ROLES A ROL\_C

```
GRANT select ON C##CRIVMAR.EMP TO C##ROL_C;
GRANT C##ROL_D TO C##ROL_C;
GRANT C##ROL_E TO C##ROL_C;
```

## ASIGNAR PRIVILEGIOS A ROL\_D

```
GRANT delete ON C##CRIVMAR.EMP TO C##ROL_D;
```

## ASIGNAR PRIVILEGIOS A ROL\_E

```
GRANT insert ON C##CRIVMAR.DEPT TO C##ROL_E;
```

## ASIGNAR ROL A USUARIO

```
GRANT C##ROL_A TO C##BECARIO2;
```

## COMPROBACIÓN

SI, POR ROL

```
exec concesion_privilegios('c##becario2','insert','dept');
```

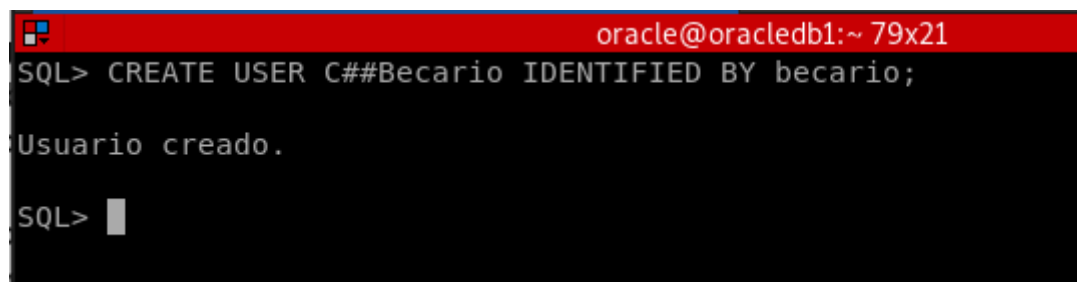
# Parte grupal

## CASO PRÁCTICO 1:

### 1. (ORACLE, Postgres, MySQL) Crea un usuario llamado Becario y, sin usar los roles de ORACLE, dale los siguientes privilegios: (1,5 puntos)

- Conectarse a la base de datos.
- Modificar el número de errores en la introducción de la contraseña de cualquier usuario.
- Modificar índices en cualquier esquema (este privilegio podrá pasarlo a quien quiera)
- Insertar filas en scott.emp (este privilegio podrá pasarlo a quien quiera)
- Crear objetos en cualquier tablespace.
- Gestión completa de usuarios, privilegios y roles.

```
CREATE USER C##BECARIO IDENTIFIED BY BECARIO;
```



```
oracle@oracledb1:~ 79x21
SQL> CREATE USER C##Becario IDENTIFIED BY becario;

Usuario creado.

SQL> █
```

## Conectarse a la base de datos

```
GRANT CREATE SESSION TO C##BECARIO;
```



```
oracle@oracle:/home/debian 80x7

SQL> GRANT CREATE SESSION TO C##BECARIO;

Concesion terminada correctamente.

SQL>
```

Y ahora nos conectamos a la base de datos con dicho usuario:

```
oracle@oracle:/home/debian 80x16
Version 19.3.0.0.0
[oracle@oracle debian]$ sqlplus

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Dec 17 12:27:07 2021
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Enter user-name: C##BECARIO
Enter password:

Conectado a:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> █
```

## Modificar el número de errores en la introducción de la contraseña de cualquier usuario

```
GRANT CREATE PROFILE TO C##BECARIO CONTAINER=ALL;
GRANT ALTER PROFILE TO C##BECARIO CONTAINER=ALL;
GRANT ALTER USER TO C##BECARIO CONTAINER=ALL;
```

```
SQL> GRANT CREATE PROFILE TO C##BECARIO CONTAINER=ALL;

Concesion terminada correctamente.

SQL> GRANT ALTER PROFILE TO C##BECARIO CONTAINER=ALL;

Concesion terminada correctamente.

SQL> GRANT ALTER USER TO C##BECARIO CONTAINER=ALL;

Concesion terminada correctamente.

SQL>
```

Ahora lo probamos:

Creamos el perfil:

```
SQL> CREATE PROFILE C##CONTRA_LIMITE LIMIT FAILED_LOGIN_ATTEMPTS 3;

Perfil creado.

SQL>
```

Modificamos el perfil:

```
SQL> ALTER PROFILE C##CONTRA_LIMITE LIMIT FAILED_LOGIN_ATTEMPTS 5;

Perfil modificado.
```

Asignamos el perfil a un usuario:

```
SQL> ALTER USER C##SCOTT PROFILE C##CONTRA_LIMITE;

Usuario modificado.
```

## Modificar índices en cualquier esquema (este privilegio podrá pasarlo a quien quiera)

```
GRANT ALTER ANY INDEX TO C##Becario WITH ADMIN OPTION;
```

```
oracle@oracledb1:~ 79x21
SQL> GRANT ALTER ANY INDEX TO C##Becario WITH ADMIN OPTION;

Concesion terminada correctamente.

SQL>
```

Creamos el índice

```
CREATE INDEX Prueba ON DEPT(DNAME);
```

```
SQL> CREATE INDEX Prueba ON DEPT(DNAME);

Indice creado.

SQL> █
```

Visualizaremos los índices de la tabla DEPT

```
SELECT INDEX_NAME FROM ALL_INDEXES WHERE TABLE_NAME='DEPT';
```

Crear un usuario para que C##Becario le dé el permiso

```
CREATE USER C##Becario2 IDENTIFIED BY becario2;
GRANT ALTER ANY INDEX TO C##BECARIO2;
```

```
oracle@oracledb1:~ 79x21
[oracle@oracledb1 ~]$ sqlplus c##becario/becario

SQL*Plus: Release 19.0.0.0.0 - Production on Fri Dec 17 12:31:59 2021
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Conectado a:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> GRANT ALTER ANY INDEX TO C##BECARIO2;

Concesion terminada correctamente.

SQL> █
```

`ALTER INDEX C##CRIVMAR.PRUEBA MONITORING USAGE;` (Hay que usar el nombre totalmente cualificado, es decir, que haga referencia al usuario que creó el índice)

```
oracle@oracledb1:~ 79x21
Desconectado de Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
^[A[oracle@oracledb1 ~]$ sqlplus c##becario2/becario2

SQL*Plus: Release 19.0.0.0.0 - Production on Tue Dec 21 09:44:07 2021
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Hora de Ultima Conexion Correcta: Mar Dic 21 2021 09:32:56 +01:00

Conectado a:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> ALTER INDEX C##CRIVMAR.PRUEBA MONITORING USAGE;

Indice modificado.
```

`SELECT * FROM V$OBJECT_USAGE WHERE TABLE_NAME='DEPT';` # debe dar un "SI" en el monitoreo.

```
oracle@oracledb1:~ 79x26
[oracle@oracledb1 ~]$ sqlplus c##crivmar/

SQL*Plus: Release 19.0.0.0.0 - Production on Tue Dec 21 09:47:06 2021
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Hora de Ultima Conexion Correcta: Mar Dic 21 2021 09:36:43 +01:00

Conectado a:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> SELECT * FROM V$OBJECT_USAGE WHERE TABLE_NAME='DEPT';

INDEX_NAME
-----
-
TABLE_NAME
-----
-
MON USE START_MONITORING END_MONITORING
-----
PRUEBA
DEPT
YES NO 12/21/2021 09:44:23
```

## Insertar filas en scott.emp (este privilegio podrá pasarlo a quien quiera)

```
GRANT INSERT ON C##SCOTT.EMP TO C##BECARIO WITH GRANT OPTION;
```

```
SQL> GRANT INSERT ON C##SCOTT.EMP TO C##BECARIO WITH GRANT OPTION;

Concesion terminada correctamente.
```

Vamos a probar a insertar datos en esa tabla:

```
SQL> INSERT INTO C##SCOTT.EMP VALUES(7469, 'SMITH', 'CLERK', 7902, TO_DATE('12-17-1980','MM-DD-YYYY'), 800, NULL, 20)
2 ;

1 fila creada.
```

## Crear objetos en cualquier tablespace

```
GRANT UNLIMITED TABLESPACE TO C##BECARIO;
```

Para probar esto, primero creamos un nuevo tablespace con el usuario sysdba:

```
SQL> CREATE BIGFILE TABLESPACE bigtbs_01
DATAFILE 'bigtbs_f1.dat'
SIZE 20M AUTOEXTEND ON; 2 3

Tablespace creado.
```

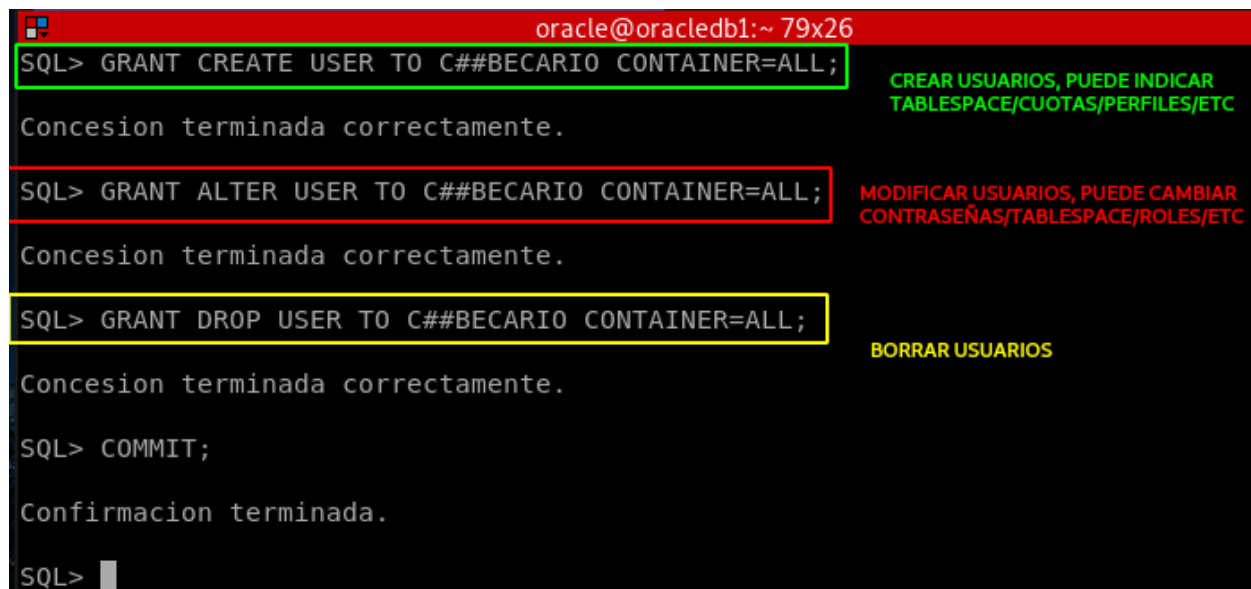
Ahora probemos a crear un objeto en el nuevo tablespace con el usuario C##Becario (antes le hemos dado permiso para crear tablas):

```
SQL> CREATE TABLE PRUEBA (  
IDEN NUMBER,  
NOMBRE VARCHAR2 (20)  
) TABLESPACE bigtbs_01; 2    3    4  
  
Tabla creada.
```

## Gestión completa de usuario, privilegios y roles

### Usuarios

```
GRANT CREATE USER TO C##BECARIO CONTAINER=ALL;  
GRANT ALTER USER TO C##BECARIO CONTAINER=ALL;  
GRANT DROP USER TO C##BECARIO CONTAINER=ALL;
```



The screenshot shows a terminal window titled 'oracle@oracledb1:~ 79x26'. It displays three SQL commands being executed, each followed by a confirmation message. To the right of each command, there is a descriptive note in a different color.

```
SQL> GRANT CREATE USER TO C##BECARIO CONTAINER=ALL;  
Concesion terminada correctamente.  
SQL> GRANT ALTER USER TO C##BECARIO CONTAINER=ALL;  
Concesion terminada correctamente.  
SQL> GRANT DROP USER TO C##BECARIO CONTAINER=ALL;  
Concesion terminada correctamente.  
SQL> COMMIT;  
Confirmacion terminada.  
SQL> █
```

CREAR USUARIOS, PUEDE INDICAR  
TABLESPACE/CUOTAS/PERFILES/ETC

MODIFICAR USUARIOS, PUEDE CAMBIAR  
CONTRASEÑAS/TABLESPACE/ROLES/ETC

BORRAR USUARIOS

**COMPROBAMOS:**

```
oracle@oracledb1:~ 79x26
[oracle@oracledb1 ~]$ sqlplus c##becario/becario;

SQL*Plus: Release 19.0.0.0.0 - Production on Tue Dec 21 10:25:55 2021
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Hora de Ultima Conexion Correcta: Mar Dic 21 2021 10:25:01 +01:00

Conectado a:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> CREATE USER C##PRUEBA IDENTIFIED BY prueba;

Usuario creado.

SQL> ALTER USER C##PRUEBA IDENTIFIED BY prueba REPLACE 12345;

Usuario modificado.

SQL> DROP USER C##PRUEBA;

Usuario borrado.

SQL>
```

## Privilegios

```
GRANT GRANT ANY PRIVILEGE TO C##BECARIO CONTAINER=ALL;
```

```
oracle@oracledb1:~ 79x26
[oracle@oracledb1 ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Tue Dec 21 10:30:47 2021
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Conectado a:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> GRANT GRANT ANY PRIVILEGE TO C##BECARIO CONTAINER=ALL;

Concesion terminada correctamente.

SQL> COMMIT;

Confirmacion terminada.
```

## Comprobamos

```
oracle@oracledb1:~ 79x26
[oracle@oracledb1 ~]$ sqlplus c##becario/becario;

SQL*Plus: Release 19.0.0.0.0 - Production on Tue Dec 21 10:32:23 2021
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Hora de Ultima Conexion Correcta: Mar Dic 21 2021 10:25:55 +01:00

Conectado a:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> GRANT CREATE TABLE TO C##BECARIO2;

Concesion terminada correctamente.

SQL> █
```

## Roles

```
GRANT CREATE ROLE TO C##BECARIO CONTAINER=ALL;
GRANT ALTER ANY ROLE TO C##BECARIO CONTAINER=ALL;
GRANT DROP ANY ROLE TO C##BECARIO CONTAINER=ALL;
GRANT GRANT ANY ROLE TO C##BECARIO CONTAINER=ALL;
```

```
oracle@oracledb1:~ 79x39
[oracle@oracledb1 ~]$ sqlplus / as sysdba

SQL*Plus: Release 19.0.0.0.0 - Production on Tue Dec 21 10:52:11 2021
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle. All rights reserved.

Conectado a:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> GRANT CREATE ROLE TO C##BECARIO CONTAINER=ALL;

Concesion terminada correctamente.

SQL> GRANT ALTER ANY ROLE TO C##BECARIO CONTAINER=ALL;

Concesion terminada correctamente.

SQL> GRANT DROP ANY ROLE TO C##BECARIO CONTAINER=ALL;

Concesion terminada correctamente.

SQL> GRANT GRANT ANY ROLE TO C##BECARIO CONTAINER=ALL;

Concesion terminada correctamente.

SQL> COMMIT;

Confirmacion terminada.

SQL> █
```

## Comprobamos

```
CREATE ROLE C##PRUEBA CONTAINER=ALL;
ALTER ROLE C##PRUEBA IDENTIFIED BY prueba;
```

```
oracle@oracledb1:~ 82x28

SQL> show user;
USER es "C##BECARIO"
SQL> CREATE ROLE C##PRUEBA CONTAINER=ALL;

Rol creado.

SQL> ALTER ROLE C##PRUEBA IDENTIFIED BY prueba;

Rol modificado.

SQL>
```

Para añadir privilegios a un rol habría que tener privilegios sobre el sistema, si quisiéramos que el usuario "Becario" tuviera esa capacidad, habría que darle dichos privilegios



previamente. Aquí se muestra un ejemplo, siendo "Sysdba", de cómo añadir un privilegio a un rol

```
GRANT SELECT ON ANY TABLE TO C##PRUEBA CONTAINER=ALL;
```

```
oracle@oracledb1:~ 82x28
SQL> GRANT SELECT ANY TABLE TO C##PRUEBA CONTAINER=ALL;

Concesion terminada correctamente.

SQL> show user;
USER es "SYS"
SQL> COMMIT;

Confirmacion terminada.

SQL> █
```

```
GRANT C##PRUEBA TO C##BECARIO2 CONTAINER=ALL;
```

```
oracle@oracledb1:~ 82x28
SQL> show user;
USER es "C##BECARIO"
SQL> GRANT C##PRUEBA TO C##BECARIO2 CONTAINER=ALL;

Concesion terminada correctamente.

SQL> █
```

```
DROP ROLE C##PRUEBA;
```

```
SQL> show user;
USER es "C##BECARIO"
SQL> DROP ROLE C##PRUEBA;

Rol borrado.
```

## VER PRIVILEGIOS DE UN ROL

```
SELECT CON_ID, GRANTEE, PRIVILEGE FROM CDB_SYS_PRIVS WHERE GRANTEE='C##PRUEBA'
ORDER BY PRIVILEGE;
```

```
SQL> SELECT CON_ID, GRANTEE, PRIVILEGE FROM CDB_SYS_PRIVS WHERE GRANTEE='C##PRUEBA'
        ORDER BY PRIVILEGE;

        CON_ID
-----
GRANTEE
-----
PRIVILEGE
-----
              1
C##PRUEBA
SELECT ANY TABLE

SQL> █
```

## VER ROLES DEL USUARIO CONECTADO (BECARIO2)

```
SELECT USERNAME, GRANTED_ROLE FROM USER_ROLE_PRIVS;
```

```
SQL> select username, granted_role from user_role_privs;

USERNAME
-----
GRANTED_ROLE
-----
C##BECARIO2
C##PRUEBA

SQL> █
```

## Postgres

### Crear usuario y darle permisos para conectarse a la base de datos:

```
CREATE USER Becario WITH PASSWORD 'becario';
```

```
postgres=# CREATE USER becario WITH PASSWORD 'becario';
CREATE ROLE
```

```
CREATE DATABASE prueba;
GRANT CONNECT ON DATABASE prueba TO Becario;
```

```
postgres=# CREATE DATABASE prueba;
CREATE DATABASE
postgres=# GRANT CONNECT ON DATABASE prueba TO Becario;
GRANT
```

Probamos si se puede conectar:

```
postgres@postgre1:/home/debian$ psql -U becario -d prueba -h localhost
Contraseña para usuario becario:
psql (11.14 (Debian 11.14-0+deb10u1))
conexión SSL (protocolo: TLSv1.3, cifrado: TLS_AES_256_GCM_SHA384, bits: 256, co
mpresión: desactivado)
Digite «help» para obtener ayuda.

prueba=>
```

## Modificar el número de errores en la introducción de la contraseña de cualquier usuario

No existe en PostgreSQL

## Modificar índices en cualquier esquema (este privilegio podrá pasarlo a quien quiera)

En PostgreSQL no existe el privilegio de modificar índices como tal, sino que un usuario podrá crear y modificar índices sobre tablas en las que tenga privilegios. Por esto, para que el usuario pudiera modificar índices en cualquier esquema, tendría que tener permisos sobre todas las tablas. Este permiso por tanto sería bastante peligroso. La sintaxis sería la siguiente:

```
ALTER ROLE becario WITH SUPERUSER;
ALTER INDEX <nombre_indice> ON <nombretabla>;
```

## Insertar filas en scott.emp (este privilegio podrá pasarlo a quien quiera)

Primero nos conectamos a la base de datos scott y después damos el permiso:

```
\CONNECT SCOTT
GRANT INSERT ON EMP TO becario with grant option;
```

```
postgres=# \connect scott
Ahora está conectado a la base de datos «scott» con el usuario «postgres».
scott=# grant insert on emp to becario with grant option;
GRANT
scott=#
```

Probemos si ha funcionado:

```

postgres@postgre1:/home/debian$ psql -U becario -d scott -h localhost
Contraseña para usuario becario:
psql (11.14 (Debian 11.14-0+deb10u1))
conexión SSL (protocolo: TLSv1.3, cifrado: TLS_AES_256_GCM_SHA384, bits: 256, compresión: desactivado)
Digite «help» para obtener ayuda.

scott=> select * from dept;  Sin permisos
ERROR:  permiso denegado a la tabla dept
scott=> insert into emp values (7869, 'SMITH', 'CLERK', 7902, '1980-12-17', 800, NULL, NULL);
ERROR:  el valor null para la columna «deptno» viola la restricción not null
DETALLE:  La fila que falla contiene (empno, ename, job, mgr, hiredate, sal, comm, deptno) = (7869, SMITH, CLERK, 7902, 19
80-12-17, 800.00, null, null).
scott=> insert into emp values (7869, 'SMITH', 'CLERK', 7902, '1980-12-17', 800, NULL, 30);  Sí nos deja
INSERT 0 1
scott=> insert into dept values (50,'IT','Michigan');  Sin permisos
ERROR:  permiso denegado a la tabla dept
scott=>

```

## Crear objetos en cualquier tablespace.

PostgreSQL no tiene un permiso específico que permita a los usuarios crear objetos en cualquier tablespace. En su lugar, tenemos que crear un tablespace y después asignar los permisos sobre ese tablespace al usuario que queramos:

```

postgres=# create tablespace prueba_space location '/mnt/prueba';
CREATE TABLESPACE
postgres=# grant create on tablespace prueba_space to becario;
GRANT
postgres=#

```

Ahora probemos si ese usuario puede crear objetos en ese tablespace:

```

postgres@postgre1:/home/debian$ psql -U becario -d prueba -h localhost
Contraseña para usuario becario:
psql (11.14 (Debian 11.14-0+deb10u1))
conexión SSL (protocolo: TLSv1.3, cifrado: TLS_AES_256_GCM_SHA384, bits: 256, compresión: desactivado)
Digite «help» para obtener ayuda.

prueba=> create table tabla (id integer,nombre varchar) tablespace prueba_space;
CREATE TABLE
prueba=>

```

## Gestión completa de usuarios, privilegios y roles.

En PostgreSQL un rol también es un usuario. Tampoco hay una forma tan granular de otorgar privilegios como la hay en Oracle, por lo que para tener una gestión completa de usuarios y roles, el usuario Becario tendría que tener privilegios de superusuario (administrador). Estos privilegios se otorgarían de la siguiente forma:

```
ALTER ROLE becario WITH SUPERUSER;
```

## MySQL

### Crear usuario

```
create user 'becario' identified by 'becario';
```

```
MariaDB [(none)]> create user 'becario' identified by 'becario';
Query OK, 0 rows affected (0.055 sec)

MariaDB [(none)]> █
```

## Conectarse a la base de datos

En Mariadb no hay un permiso específico para que pueda conectarse a una base de datos. El permiso 'usage' es lo mismo que decirle que no tiene ningún privilegio y por tanto no podrá conectarse a ninguna base de datos. Para que pueda conectarse a SCOTT (por ejemplo) deberíamos darle un privilegio al menos sobre las tablas de ese esquema, en este le daremos simplemente la posibilidad de hacer "select".

```
grant select on SCOTT.* to 'becario'@'%';
```

```
MariaDB [(none)]> grant select on SCOTT.* to 'becario'@'%';
Query OK, 0 rows affected (0.017 sec)

MariaDB [(none)]> commit;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> █
```

Comprobamos:

```
debian@db: ~ 82x28
debian@db:~$ mysql -u becario -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 46
Server version: 10.5.11-MariaDB-1 Debian 11

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use empresa;
ERROR 1044 (42000): Access denied for user 'becario'@'localhost' to database 'empresa'

MariaDB [(none)]> use SCOTT;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [SCOTT]> █
```

## Modificar el número de errores en la introducción de la contraseña de cualquier usuario.

MariaDB no tiene opción de dar un número de intentos de ingresar la contraseña, si fuera MySQL entonces sería:

```
alter user 'becario'@'%' failed_login_attempts <nº_intentos> password_lock_time <tiempo_de_bloqueo>;
```

## Modificar índices en cualquier esquema (este privilegio podrá pasarse a quien quiera)

En MariaDB no hay la opción de alterar un índice como en Oracle, pero si tiene permiso de verlos, sin poder delegarlo a otros. Las otras opciones sería añadir, eliminar y alterar un índice, que son los privilegios de crear, borrar y alterar tablas. Esto sería bastante peligroso ya que le darías las opciones de modificar la base de datos como si fuera un administrador y además le darías la posibilidad de que los repartiera a su criterio.

## Insertar filas en SCOTT.EMP (este privilegio podrá pasarlo a quien quiera)

```
grant insert on SCOTT.EMP to 'becario'@'%' with grant option;
```

```
MariaDB [(none)]> grant insert on SCOTT.EMP to 'becario'@'%' with grant option;  
Query OK, 0 rows affected (0.057 sec)
```

```
MariaDB [(none)]> commit;  
Query OK, 0 rows affected (0.000 sec)
```

```
MariaDB [(none)]> █
```

Comprobamos:

```
insert into EMP values ('7787','LORIS','CLERK',null,'1980-09-22',null,  
null,'10');
```

```
debian@db:~$ mysql -u becarior -p  
Enter password:  
Welcome to the MariaDB monitor.  Commands end with ; or \g.  
Your MariaDB connection id is 66  
Server version: 10.5.11-MariaDB-1 Debian 11  
  
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
MariaDB [(none)]> use SCOTT;  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A
```

```
Database changed  
MariaDB [SCOTT]> insert into EMP values ('7787','LORIS','CLERK',null,'1980-09-22',null, null,'10');  
Query OK, 1 row affected (0.014 sec)
```

```
MariaDB [SCOTT]> insert into DEPT values ('50','ALGO','SITIO');  
ERROR 1142 (42000): INSERT command denied to user 'becario'@'localhost' for table 'DEPT'
```

```
MariaDB [SCOTT]>  
MariaDB [SCOTT]> grant insert on SCOTT.EMP to 'becario2'@'%;  
Query OK, 0 rows affected (0.031 sec)
```

```
MariaDB [SCOTT]>
```

PUEDE EN LA TABLA EMPLEADOS

NO PUEDE EN LA  
TABLA DEPT

DELEGA EL PRIVILEGIO A BECARIO2

## Crear objetos en cualquier tablespace.

En versiones anteriores de MariaDB tenía esta característica heredada de MySQL, pero en versiones más recientes no ha sido incluida.

No se puede crear ni alterar los espacios de tablas.

## Gestión completa de usuarios, privilegios, roles.

En Mariadb todos estos privilegios están sujetos al administrador y por tanto no hay tanta versatilidad de darlos por separado como en oracle; aunque si hay la excepción de permitir crear usuarios. Es por ello que para que pudiera hacer todo, debería obtener TODOS los privilegios como un administrador, pero se lo vamos a dar sólo sobre la base de datos SCOTT para tenerlo más controlado.

```
grant all privileges on SCOTT.* to 'becario'@'%';
```

```
MariaDB [(none)]> grant all privileges on SCOTT.* to 'becario'@'%';  
Query OK, 0 rows affected (0.015 sec)  
  
MariaDB [(none)]>
```

Podemos ver los permisos que tiene un usuario:

```
show grants for 'becario'@'%';
```

```
MariaDB [(none)]> show grants for 'becario'@'%';  
+-----+  
| Grants for becario@% |  
+-----+  
| GRANT USAGE ON *.* TO `becario`@`%` IDENTIFIED BY PASSWORD '*289FA4B539971D43DB630F60B1C72F8335DF1245' WITH GRANT OPTION |  
| GRANT ALL PRIVILEGES ON `SCOTT`.* TO `becario`@`%` |  
| GRANT INSERT ON `SCOTT`.`EMP` TO `becario`@`%` WITH GRANT OPTION |  
+-----+  
3 rows in set (0.000 sec)  
  
MariaDB [(none)]> █
```

## 2. Escribe una consulta que obtenga un script para quitar el privilegio de borrar registros en alguna tabla de SCOTT a los usuarios que lo tengan.

### Oracle

```
select 'revoke delete on C##SCOTT.' || table_name || ' from ' || grantee || ';'   
from DBA_TAB_PRIVS
```

```
where OWNER='C##SCOTT'
and PRIVILEGE='DELETE';
```

```
SQL> select 'revoke delete on C##SCOTT.' || table_name || ' from ' || grantee || ';'
  2   from DBA_TAB_PRIVS
  3   where OWNER='C##SCOTT'
  4   and PRIVILEGE='DELETE';

'REVOKEDELETEONC##SCOTT.' || TABLE_NAME || 'FROM' || GRANTEE || ';'
-----
revoke delete on C##SCOTT.DEPT from C##USUARIO3;
revoke delete on C##SCOTT.EMP from C##S4DBRD;
```

## POSTGRESQL

```
select 'Revoke Delete on ' || table_catalog || '.' || table_name || ' from
' || grantee || ';'
from information_schema.role_table_grants
where table_catalog='scott'
and privilege_type='DELETE'
and table_schema='scott';
```

```
scott=> select 'Revoke Delete on ' || table_catalog || '.' || table_name || ' from ' || grantee || ';'
scott-> from information_schema.role_table_grants
scott-> where table_catalog='scott'
scott-> and privilege_type='DELETE'
scott-> and table_schema='scott';
          ?column?
-----
Revoke Delete on scott.bonus from scott;
Revoke Delete on scott.salgrade from scott;
Revoke Delete on scott.dummy from scott;
Revoke Delete on scott.dept from scott;
Revoke Delete on scott.dept from dparrales1;
Revoke Delete on scott.emp from scott;
Revoke Delete on scott.emp from dparrales1;
(7 filas)

scott=> █
```

## MARIADB

```
select concat('Revoke Delete on ', table_schema, '.', table_name, ' from
', grantee, ';') as script
from information_schema.table_privileges
where table_schema='scott'
and privilege_type='DELETE';
```



```

MariaDB [(none)]> select concat('Revoke Delete on ',table_schema,'.',table_name,' from ',grantee,';') as script
-> from information_schema.table_privileges
-> where table_schema='scott'
-> and privilege_type='DELETE';
+-----+
| script |
+-----+
| Revoke Delete on scott.dept from 'dparrales'@'%'; |
| Revoke Delete on scott.emp from 'dparrales'@'%'; |
+-----+
2 rows in set (0.001 sec)

```

### 3. Crea un tablespace TS2 con tamaño de extensión de 256K. Realiza una consulta que genere un script que asigne ese tablespace como tablespace por defecto a los usuarios que no tienen privilegios para consultar ninguna tabla de SCOTT, excepto a SYSTEM.

Creación del tablespace.

```

CREATE TABLESPACE TS2
DATAFILE 'ts2_data.dbf'
SIZE 256k;

```

```

SQL> CREATE TABLESPACE TS2
DATAFILE 'ts2_data.dbf'
SIZE 256k;
2      3
Tablespace creado.

```

Creación de la consulta.

```

select 'alter user "'||username||'" default tablespace TS2;'
from DBA_USERS
where USERNAME not in (select GRANTEE from DBA_TAB_PRIVS where
PRIVILEGE='SELECT' and OWNER='C##SCOTT')
and USERNAME!='SYSTEM' and USERNAME!='C##SCOTT' ;

```

```

SQL> select 'alter user "'||username||'" default tablespace TS2;'
      from DBA_USERS
      where USERNAME not in (select GRANTEE from DBA_TAB_PRIVS where PRIVILEGE='SELECT' and OWNER='C##SCOTT')
      and USERNAME!='SYSTEM' and USERNAME!='C##SCOTT' ;
 2      3      4
'ALTERUSER"'||USERNAME||'"DEFAULTTABLESPACETS2;'
-----
alter user "XS$NULL" default tablespace TS2;
alter user "SYS" default tablespace TS2;
alter user "OJMSYS" default tablespace TS2;
alter user "LBACSYS" default tablespace TS2;
alter user "OUTLN" default tablespace TS2;
alter user "SYS$UMF" default tablespace TS2;
alter user "DBSNMP" default tablespace TS2;
alter user "APPOSSYS" default tablespace TS2;
alter user "DBSFUSER" default tablespace TS2;
alter user "GGSYS" default tablespace TS2;
alter user "ANONYMOUS" default tablespace TS2;

'ALTERUSER"'||USERNAME||'"DEFAULTTABLESPACETS2;'
-----
alter user "CTXSYS" default tablespace TS2;
alter user "DVSYS" default tablespace TS2;
alter user "DVF" default tablespace TS2;
alter user "GSMADMIN_INTERNAL" default tablespace TS2;
alter user "MDSYS" default tablespace TS2;
alter user "OLAPSYS" default tablespace TS2;
alter user "XDB" default tablespace TS2;
alter user "WMSYS" default tablespace TS2;
alter user "GSMCATUSER" default tablespace TS2;
alter user "MDDATA" default tablespace TS2;
alter user "C##DPARRALES" default tablespace TS2;

'ALTERUSER"'||USERNAME||'"DEFAULTTABLESPACETS2;'
-----
alter user "SYSBACKUP" default tablespace TS2;
alter user "REMOTE_SCHEDULER_AGENT" default tablespace TS2;
alter user "GSMUSER" default tablespace TS2;
alter user "SYSRAC" default tablespace TS2;
alter user "GSMROOTUSER" default tablespace TS2;
alter user "DANIM" default tablespace TS2;
alter user "SI_INFORMTN_SCHEMA" default tablespace TS2;
alter user "C##DPARRALES1" default tablespace TS2;
alter user "AUDSYS" default tablespace TS2;
alter user "DIP" default tablespace TS2;
alter user "ORDPLUGINS" default tablespace TS2;

'ALTERUSER"'||USERNAME||'"DEFAULTTABLESPACETS2;'
-----
alter user "SYSKM" default tablespace TS2;
alter user "ORDDATA" default tablespace TS2;
alter user "ORACLE_OCM" default tablespace TS2;

```

**4. Realiza un procedimiento que reciba un nombre de usuario y nos muestre cuántas sesiones tiene abiertas en este momento. Además, para cada una de dichas sesiones nos mostrará la hora de comienzo y el nombre de la máquina, sistema operativo y programa desde el que fue abierta.**

**ORACLE:**

---

```

create or replace procedure MostrarSesiones(p_user varchar2)
is
    v_numero number:=0;
    cursor c_sesiones
    is
        select to_char(LOGON_TIME, 'HH24:MI') as HORA,
               MACHINE,
               PROGRAM
        from v$session
        where USERNAME=p_user;
    v_cursor c_sesiones%ROWTYPE;
begin
    for v_cursor in c_sesiones loop
        dbms_output.put_line(chr(10)||'Hora: '||v_cursor.HORA||chr(9)||'Maquina: '||v_cursor.MACHINE||chr(9)||'Programa: '||v_cursor.PROGRAM);
        v_numero:=v_numero+1;
    end loop;
    dbms_output.put_line(chr(10)||'Numero de sesiones abiertas: '||v_numero);
end MostrarSesiones;
/

```

```

SQL> create or replace procedure MostrarSesiones(p_user varchar2)
2  is
3  v_numero number:=0;
4  cursor c_sesiones
5  is
6      select to_char(LOGON_TIME, 'HH24:MI') as HORA,
7             MACHINE,
8             PROGRAM
9      from v$session
10     where USERNAME=p_user;
11  v_cursor c_sesiones%ROWTYPE;
12  begin
13      for v_cursor in c_sesiones loop
14          dbms_output.put_line(chr(10)||'Hora: '||v_cursor.HORA||chr(9)||'Maquina: '||v_cursor.MACHINE||chr(9)||'Programa: '||v_cursor.PROGRAM);
15          v_numero:=v_numero+1;
16      end loop;
17      dbms_output.put_line(chr(10)||'Numero de sesiones abiertas: '||v_numero);
18  end MostrarSesiones;
19  /

Procedimiento creado.

SQL> exec MostrarSesiones('SYS');

Hora: 09:28      Maquina: WORKGROUP\DESKTOP-7IUFICD      Programa: sqlplus.exe

Numero de sesiones abiertas: 1

Procedimiento PL/SQL terminado correctamente.

```

## Postgres

```

create or replace function MostrarNumSesiones(p_usuario VARCHAR)
returns void
as
$$
DECLARE
    v_cont numeric:=0;
    c_sesiones CURSOR IS
        select *
        from PG_STAT_ACTIVITY
        where USENAME=lower(p_usuario)
        AND STATE IS NOT NULL;
BEGIN

```

```

FOR v_cursor IN c_sesiones LOOP
    RAISE NOTICE 'Hora: %', to_char(v_cursor.BACKEND_START, 'DD/MM/YYYY
HH24:MI');
    RAISE NOTICE 'Maquina: %', v_cursor.CLIENT_HOSTNAME;
    RAISE NOTICE 'Programa: %', v_cursor.APPLICATION_NAME;
    RAISE NOTICE ' ';
    v_cont:=v_cont+1;
END LOOP;
    RAISE NOTICE 'Numero de sesiones abiertas: %', v_cont;
END;
$$ language plpgsql;

```

```

postgres=# create or replace function MostrarNumSesiones(p_usuario VARCHAR)
postgres=# returns void
postgres=# as
postgres=# $$
postgres$# DECLARE
postgres$#     v_cont numeric:=0;
postgres$#     c_sesiones CURSOR IS
postgres$#         select *
postgres$#         from PG_STAT_ACTIVITY
postgres$#         where USERNAME=lower(p_usuario)
postgres$#         AND STATE IS NOT NULL;
postgres$# BEGIN
postgres$#     FOR v_cursor IN c_sesiones LOOP
postgres$#         RAISE NOTICE 'Hora: %', to_char(v_cursor.BACKEND_START, 'DD/MM/YYYY HH24:MI');
postgres$#         RAISE NOTICE 'Maquina: %', v_cursor.CLIENT_HOSTNAME;
postgres$#         RAISE NOTICE 'Programa: %', v_cursor.APPLICATION_NAME;
postgres$#         RAISE NOTICE ' ';
postgres$#         v_cont:=v_cont+1;
postgres$#     END LOOP;
postgres$#         RAISE NOTICE 'Numero de sesiones abiertas: %', v_cont;
postgres$# END;
postgres$# $$ language plpgsql;
CREATE FUNCTION
postgres=# select MostrarNumSesiones('postgres');
NOTICE:  Hora: 16/01/2022 20:33
NOTICE:  Maquina: <NULL>
NOTICE:  Programa: psql
NOTICE:
NOTICE:  Numero de sesiones abiertas: 1

```

El hecho de que salga <NULL> en la función creo que es porque nos encontramos en una máquina virtual

## 5. Realiza un procedimiento que muestre los usuarios que pueden conceder privilegios de sistema a otros usuarios y cuales son dichos privilegios.

```

create or replace procedure SysPrivs
is
    cursor c_mostrar_userprivs
    is

```

```

        select grantee, privilege
        from dba_sys_privs
        where admin_option = 'YES';
begin
    for usuario in c_mostrar_userprivs loop
        dbms_output.put_line(chr(10)||'Usuario: '||lpad(usuario.grantee,
25)||chr(9)||'Privilegio: '||lpad(usuario.privilege, 30));
    end loop;
end SysPrivs;
/

```

SQL> exec SysPrivs;

Usuario:	SYS	Privilegio:	CREATE ANY RULE SET
Usuario:	SYS	Privilegio:	EXECUTE ANY RULE SET
Usuario:	SYS	Privilegio:	CREATE RULE
Usuario:	SYS	Privilegio:	ALTER ANY RULE
Usuario:	SCHEDULER_ADMIN	Privilegio:	MANAGE SCHEDULER
Usuario:	GSMADMIN_INTERNAL	Privilegio:	ALTER DATABASE LINK
Usuario:	SYSKM	Privilegio:	ADMINISTER KEY MANAGEMENT
Usuario:	SYS	Privilegio:	DROP ANY RULE
Usuario:	AQ_ADMINISTRATOR_ROLE	Privilegio:	CREATE RULE SET
Usuario:	SYS	Privilegio:	CREATE RULE SET
Usuario:	SCHEDULER_ADMIN	Privilegio:	EXECUTE ANY PROGRAM
Usuario:	AQ_ADMINISTRATOR_ROLE	Privilegio:	MANAGE ANY QUEUE
Usuario:	SYS	Privilegio:	CREATE ANY EVALUATION CONTEXT
Usuario:	SYS	Privilegio:	ALTER ANY EVALUATION CONTEXT
Usuario:	SYS	Privilegio:	CREATE ANY RULE
Usuario:	SYS	Privilegio:	EXECUTE ANY RULE
Usuario:	SCHEDULER_ADMIN	Privilegio:	CREATE JOB
Usuario:	SCHEDULER_ADMIN	Privilegio:	CREATE ANY JOB

## Caso Práctico 2

**1. La vida de un DBA es dura. Tras pedirlo insistentemente, en tu empresa han**

# contratado una persona para ayudarte.

## Decides que se encargará de las siguientes tareas:

Creamos el usuario

```
CREATE USER C##AYUDANTE IDENTIFIED BY ayudante;
```

```
SQL> CREATE USER C##AYUDANTE IDENTIFIED BY ayudante;  
Usuario creado.  
SQL> █
```

Y le damos el permiso para iniciar sesión:

```
GRANT CREATE SESSION TO C##AYUDANTE;
```

```
SQL> GRANT CREATE SESSION TO C##AYUDANTE;  
Concesion terminada correctamente.
```

## Resetear los archivos de log en caso de necesidad

Para ello debemos dar al usuario el privilegio de modificar la base de datos (algo que nos parece bastante inseguro):

```
GRANT ALTER DATABASE TO C##AYUDANTE CONTAINER=ALL;
```

```
SQL> GRANT ALTER DATABASE TO C##AYUDANTE CONTAINER=ALL;  
Concesion terminada correctamente.
```

Ahora probemos si el usuario puede resetear los logs:

```
ALTER DATABASE CLEAR LOGFILE GROUP 3;
```

```
SQL> show user
USER es "C##AYUDANTE"
SQL> ALTER DATABASE CLEAR LOGFILE GROUP 3;

Base de datos modificada.
```

Como vemos, ahora el log aparece como "Unused":

```
SQL> select GROUP#,THREAD#,SEQUENCE#,bytes/1024/1024,
MEMBERS,STATUS from v$log; 2
```

GROUP#	THREAD#	SEQUENCE#	BYTES/1024/1024	MEMBERS	STATUS
1	1	70	200	1	INACTIVE
2	1	71	200	1	CURRENT
3	1	0	200	1	UNUSED

## Crear funciones de complejidad de contraseña y asignárselas a usuarios.

Esta parte es imposible de realizar para un usuario que no sea SYS, ya que Oracle no deja asignar funciones de verificación de contraseñas que no hayan sido creadas por el usuario SYS, y no deja asignarlas a nadie que no sea SYS. Aunque no podamos hacerlo con el usuario C##AYUDANTE, los pasos para realizar esto con el usuario SYS son los siguientes:

- Primero creamos la función:

```
SQL> show user
USER es "SYS"
SQL> CREATE OR REPLACE FUNCTION
MY_VERIFICATION_FUNCTION (
  USERNAME VARCHAR2,
  PASSWORD VARCHAR2,
  OLD_PASSWORD VARCHAR2)
RETURN BOOLEAN AS
BEGIN
  IF LENGTH(PASSWORD) < 4 THEN
    RETURN FALSE;
  ELSE
    RETURN TRUE;
  END IF;
END MY_VERIFICATION_FUNCTION;
/ 2 3 4 5 6 7 8 9 10 11 12 13 14

Funcion creada.
```

- Creamos un perfil:

```
SQL> CREATE PROFILE C##PRUEBA12 LIMIT;

Perfil creado.

SQL> SHOW USER
USER es "C##AYUDANTE"
```

- Asignamos la función que hemos creado a dicho perfil:

```
SQL> ALTER PROFILE C##PRUEBA12 LIMIT PASSWORD_VERIFY_FUNCTION MY_VERIFICATION_FUNCTION;

Perfil modificado.
```

- A partir de aquí, ya podemos asignar dicho perfil a los usuarios con el usuario C##AYUDANTE si le damos el siguiente permiso:

```
GRANT ALTER USER TO C##AYUDANTE CONTAINER=ALL;
```

```
SQL> GRANT ALTER USER TO C##AYUDANTE CONTAINER=ALL;

Concesion terminada correctamente.
```

- Asignamos dicho perfil a un usuario:

```
SQL> ALTER USER C##SCOTT PROFILE C##PRUEBA12;

Usuario modificado.

SQL> SHOW USER
USER es "C##AYUDANTE"
```

- Ahora probamos si a ese usuario le funcionan las restricciones que le hemos puesto al perfil (no puede ser una contraseña con menos de cuatro caracteres):

```
Enter user-name: C##SCOTT
Enter password:
Hora de Ultima Conexion Correcta: Mie Nov 24 2021 19:03:13 +01:00

Conectado a:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> ALTER USER C##SCOTT IDENTIFIED BY abc REPLACE TIGER;
ALTER USER C##SCOTT IDENTIFIED BY abc REPLACE TIGER
*
ERROR en línea 1:
ORA-28003: fallo en la verificación de la contraseña especificada
```

Como vemos, la función de verificación de contraseñas funciona bien, pero estamos limitados en cuanto a que el usuario SYS debe ser quien cree la función y quien la asigna. Si intentamos asignar esa función con el usuario C##AYUDANTE nos da el siguiente error:

```
SQL> ALTER PROFILE C##PRUEBA12 LIMIT PASSWORD_VERIFY_FUNCTION MY_VERIFICATION_FUNCTION;
ALTER PROFILE C##PRUEBA12 LIMIT PASSWORD_VERIFY_FUNCTION MY_VERIFICATION_FUNCTION
*
ERROR en línea 1:
ORA-07443: función MY_VERIFICATION_FUNCTION no encontrada
```

**Eliminar la información de rollback. (este privilegio podrá pasarlo a quien quiera)**



```
GRANT DROP ROLLBACK SEGMENT TO C##AYUDANTE WITH ADMIN OPTION;
```

```
SQL> GRANT DROP ROLLBACK SEGMENT TO C##AYUDANTE WITH ADMIN OPTION;  
Concesion terminada correctamente.
```

Probamos si puede borrar la información de rollback:

```
SQL> show user;  
USER es "C##AYUDANTE"  
SQL> DROP ROLLBACK SEGMENT rbs_one;  
  
Segmento de rollback borrado.
```

Y comprobamos si puede dar el permiso a cualquiera:

```
SQL> SHOW USER;  
USER es "C##AYUDANTE"  
SQL> GRANT DROP ROLLBACK SEGMENT TO C##BECARIO;  
  
Concesion terminada correctamente.
```

## Modificar información existente en la tabla dept del usuario scott (este privilegio podrá pasarlo a quien quiera)

Para ello deberá ser capaz de insertar nuevos datos, modificar los datos existentes (para poder hacer un update también debemos darle el privilegio de select) y borrar los datos:

```
GRANT UPDATE ON C##SCOTT.DEPT TO C##AYUDANTE WITH GRANT OPTION;  
GRANT INSERT ON C##SCOTT.DEPT TO C##AYUDANTE WITH GRANT OPTION;  
GRANT DELETE ON C##SCOTT.DEPT TO C##AYUDANTE WITH GRANT OPTION;
```

```
SQL> GRANT UPDATE ON C##SCOTT.DEPT TO C##AYUDANTE WITH GRANT OPTION;  
Concesion terminada correctamente.
```

```
SQL> GRANT INSERT ON C##SCOTT.DEPT TO C##AYUDANTE WITH GRANT OPTION;  
Concesion terminada correctamente.
```

```
SQL> GRANT DELETE ON C##SCOTT.DEPT TO C##AYUDANTE WITH GRANT OPTION;  
Concesion terminada correctamente.
```

```
SQL> GRANT SELECT ON C##SCOTT.DEPT TO C##AYUDANTE WITH GRANT OPTION;  
Concesion terminada correctamente.
```

Probemos si ha funcionado:

```
SQL> SHOW USER
USER es "C##AYUDANTE"
SQL> INSERT INTO C##SCOTT.DEPT VALUES (50,'RH','New York');

1 fila creada.
```

```
SQL> SHOW USER
USER es "C##AYUDANTE"
SQL> UPDATE C##SCOTT.DEPT SET DNAME='RRRHH' WHERE DEPTNO=50;

1 fila actualizada.

SQL> DELETE FROM C##SCOTT.DEPT WHERE DEPTNO=50;

1 fila suprimida.
```

Ahora comprobemos si puede transmitir estos privilegios a otros usuarios:

```
SQL> SHOW USER;
USER es "C##AYUDANTE"
SQL> GRANT SELECT ON C##SCOTT.DEPT TO C##BECARIO;

Concesion terminada correctamente.

SQL> GRANT UPDATE ON C##SCOTT.DEPT TO C##BECARIO;

Concesion terminada correctamente.

SQL> GRANT INSERT ON C##SCOTT.DEPT TO C##BECARIO;

Concesion terminada correctamente.

SQL> GRANT DELETE ON C##SCOTT.DEPT TO C##BECARIO;

Concesion terminada correctamente.
```

## Realizar pruebas de todos los procedimientos existentes en la base de datos.

Para ello debemos darle privilegios para poder ejecutar todos los procedimientos de la base de datos:

```
GRANT EXECUTE ANY PROCEDURE TO C##AYUDANTE;
```

```
SQL> GRANT EXECUTE ANY PROCEDURE TO C##AYUDANTE;

Concesion terminada correctamente.
```

Probemos si puede ejecutar un procedimiento de otro usuario:

```
SQL> SHOW USER
USER es "C##AYUDANTE"
SQL> EXEC C##DPARRALES1.P_INFORME1(TO_DATE('02/04/17','DD/MM/YY'),'T-123');
Aerogenerador T-123 34M140 EBC SENV10N
D??a: 02/04/17

Listas de desconexiones:
Hora: 13:00 Producci??n: 2900
Energia Total Generada: 2900

Procedimiento PL/SQL terminado correctamente.

SQL> █
```

Como vemos, puede ejecutar cualquier procedimiento de la base de datos, siempre y cuando no sean del usuario SYS. Esto está hecho como medida de seguridad, y la única forma de pasar esta medida es dar los privilegios sobre los procedimientos de SYS uno por uno.

## Poner un tablespace fuera de línea

Para ello otorgamos el siguiente privilegio:

```
GRANT MANAGE TABLESPACE TO C##AYUDANTE;
```

```
SQL> GRANT MANAGE TABLESPACE TO C##AYUDANTE;

Concesion terminada correctamente.
```

Ahora probaremos si podemos poner fuera de línea el tablespace que creamos en el ejercicio 1 de la primera parte de la práctica grupal:

```
ALTER TABLESPACE BIGTBS_01 OFFLINE;
```

```
SQL> SHOW USER
USER es "C##AYUDANTE"
SQL> ALTER TABLESPACE BIGTBS_01 OFFLINE;

Tablespace modificado.
```

Ahora comprobemos si el usuario C##BECARIO puede seguir creando objetos en dicho tablespace:

```
SQL> INSERT INTO PRUEBA2 VALUES (1,'Prueba');
INSERT INTO PRUEBA2 VALUES (1,'Prueba')
*
ERROR en línea 1:
ORA-01542: tablespace 'BIGTBS_01' offline, no se puede asignar espacio en el

SQL> █
```

Como vemos, nos indica que el tablespace está fuera de línea y no puede añadir información a la tabla.

**2. (ORACLE) Muestra el texto de la última sentencia SQL que se ejecutó en el servidor, junto con el número de veces que se ha ejecutado desde que se cargó en el Shared Pool y el tiempo de CPU empleado en su ejecución.**

```
select distinct sql_text, executions, CPU_TIME
from v$sqlarea
order by first_load_TIME desc
fetch first 1 rows only;
```

```
SQL> select distinct sql_text, executions, CPU_TIME
from v$sqlarea
order by first_load_TIME desc
fetch first 1 rows only; 2      3      4
```

```
SQL_TEXT
```

```
-----
EXECUTIONS      CPU_TIME
-----
```

```
SELECT * FROM C##SCOTT.EMP
1          20706
```

**3. (ORACLE, Postgres) Realiza un procedimiento que reciba dos nombres de usuario y genere un script que asigne al primero los privilegios de inserción y modificación sobre todas las tablas del segundo, así como el de ejecución de cualquier procedimiento que tenga el segundo usuario.**

**Oracle**

```

create or replace procedure OtorgarPrivilegios(p_user1 varchar2, p_user2
varchar2)
is
    v_validacion1 number:=0;
    v_validacion2 number:=0;
begin
    v_validacion1:=ComprobarUsuario(p_user1);
    v_validacion2:=ComprobarUsuario(p_user2);
    if v_validacion1=0 and v_validacion2=0 then
        PrivilegiosSobreTablas(p_user1, p_user2);
        PrivilegiosSobreProcedimientos(p_user1, p_user2);
    end if;
end OtorgarPrivilegios;
/

```

```

create or replace function ComprobarUsuario(p_user varchar2)
return number
is
    v_resultado varchar2(30);
begin
    select USERNAME into v_resultado
    from DBA_USERS
    where USERNAME=p_user;
    return 0;
exception
    when NO_DATA_FOUND then
        dbms_output.put_line('No existe el usuario '||p_user);
        return 1;
end ComprobarUsuario;
/

```

```

create or replace procedure PrivilegiosSobreTablas(p_user1 varchar2, p_user2
varchar2)
is
    cursor c_tablas
    is
        select OBJECT_NAME
        from DBA_OBJECTS
        where OBJECT_TYPE='TABLE'
        and OWNER=p_user2;
    v_cursor c_tablas%ROWTYPE;
begin
    for v_cursor in c_tablas loop
        dbms_output.put_line('GRANT INSERT ON
'||p_user2||'.'||v_cursor.OBJECT_NAME||' TO '||p_user1||';');
        dbms_output.put_line('GRANT UPDATE ON
'||p_user2||'.'||v_cursor.OBJECT_NAME||' TO '||p_user1||';');
    end loop;
end PrivilegiosSobreTablas;
/

```

```

create or replace procedure PrivilegiosSobreProcedimientos (p_user1 varchar2,

```

```

p_user2 varchar2)
is
    cursor c_procedimientos
    is
        select OBJECT_NAME
        from DBA_OBJECTS
        where OBJECT_TYPE='PROCEDURE'
        and OWNER=p_user2;
    v_cursor c_procedimientos%ROWTYPE;
begin
    for v_cursor in c_procedimientos loop
        dbms_output.put_line('GRANT EXECUTE ON
'||p_user2||'.'||v_cursor.OBJECT_NAME||' TO '||p_user1||';');
    end loop;
end PrivilegiosSobreProcedimientos;
/

```

```

SQL> exec OtorgarPrivilegios('USRPRACTICA1','SCOTT');
GRANT INSERT ON SCOTT.DEPT TO USRPRACTICA1;
GRANT UPDATE ON SCOTT.DEPT TO USRPRACTICA1;
GRANT INSERT ON SCOTT.EMP TO USRPRACTICA1;
GRANT UPDATE ON SCOTT.EMP TO USRPRACTICA1;

Procedimiento PL/SQL terminado correctamente.

```

```

SQL> exec OtorgarPrivilegios('USRPRACTICA1','SCOTT');
GRANT INSERT ON SCOTT.DEPT TO USRPRACTICA1;
GRANT UPDATE ON SCOTT.DEPT TO USRPRACTICA1;
GRANT INSERT ON SCOTT.EMP TO USRPRACTICA1;
GRANT UPDATE ON SCOTT.EMP TO USRPRACTICA1;
GRANT EXECUTE ON SCOTT.PRUEBA TO USRPRACTICA1;

Procedimiento PL/SQL terminado correctamente.

```

```

SQL> create or replace procedure Prueba
2 is
3 begin
4 dbms_output.put_line('Hola');
5 end Prueba;
6 /

```

Procedimiento creado.

```
SQL> select user from dual;
```

USER

SCOTT

SQL>

## Postgres

```

create or replace function ComprobarUsuario(p_nombreusuario varchar)
returns integer
as
$$
DECLARE
    v_resultado integer:=0;
BEGIN
    SELECT COUNT(*) INTO v_resultado
    FROM pg_roles
    WHERE rolname=p_nombreusuario;
    RETURN v_resultado;
END;
$$ language plpgsql;

```

```

create or replace procedure PrivilegiosSobreTablas(p_user1 varchar, p_user2
varchar)
as $$
DECLARE
    v_cursor CURSOR IS
    SELECT tablename
    FROM pg_tables
    WHERE tableowner=lower(p_user2);

```

```

        v_tabla varchar;
BEGIN
    FOR v_tabla IN v_cursor LOOP
        raise notice 'GRANT INSERT, UPDATE, DELETE ON % to %', v_tabla,
p_user1;
    END LOOP;
END;
$$ language plpgsql;

create or replace function GetOID (p_nombreusuario varchar)
returns integer
as $$
DECLARE
    v_resultado integer:=0;
BEGIN
    SELECT oid INTO v_resultado
    FROM pg_roles
    WHERE rolname=p_nombreusuario;
    RETURN v_resultado;
END;
$$ language plpgsql;

create or replace procedure PrivilegiosSobreProcedimientos(p_user1 varchar,
p_user2 varchar)
as $$
DECLARE
    v_cursor CURSOR IS
    SELECT proname
    FROM pg_proc
    WHERE proowner=GetOID(p_user2);
BEGIN
    FOR v_procedimiento IN v_cursor LOOP
        raise notice 'GRANT EXECUTE ON % to %', v_procedimiento, p_user1;
    END LOOP;
END;
$$ language plpgsql;

create or replace procedure OtorgarPrivilegios(p_user1 varchar (30), p_user2
varchar (30))
as $$
DECLARE
    v_validacion1 integer:=0;
    v_validacion2 integer:=0;
BEGIN
    v_validacion1 := ComprobarUsuario(p_user1);
    v_validacion2 := ComprobarUsuario(p_user2);
    if v_validacion1 = 0 then
        raise notice 'El usuario % no existe', p_user1;
    elsif v_validacion2 = 0 then
        raise notice 'El usuario % no existe', p_user2;
    else
        raise notice 'Asignando privilegios sobre tablas';
        call PrivilegiosSobreTablas(p_user1, p_user2);
        raise notice 'Asignando privilegios sobre procedimientos';
    end if;
END;

```

```

        call PrivilegiosSobreProcedimientos(p_user1, p_user2);
    end if;
END;
$$ language plpgsql;

create user prueba with password 'prueba';
create user prueba1 with password 'prueba';
create database prueba;
alter database ejercicio3 owner to ejercicio3;
grant connect
grant select on aerogeneradores to prueba1;
grant update on desconexiones to prueba1;
create database prueba1;
grant update on prueba1 to prueba1;

select 'ALTER TABLE ' || table_name || ' OWNER TO ejercicio3;' from
information_schema.tables where table_schema = 'public';

create or replace function Prueba()
returns integer
as $$
DECLARE
begin
    return 1;
end;
$$ language plpgsql;

```

```

ejercicio3=# call OtorgarPrivilegios('prueba','ejercicio3');
NOTICE:  Asignando privilegios sobre tablas
NOTICE:  GRANT INSERT, UPDATE, DELETE ON (producciones_aerogeneradores) to prueba
NOTICE:  GRANT INSERT, UPDATE, DELETE ON (aerogeneradores) to prueba
NOTICE:  GRANT INSERT, UPDATE, DELETE ON (centrales) to prueba
NOTICE:  GRANT INSERT, UPDATE, DELETE ON (desconexiones) to prueba
NOTICE:  GRANT INSERT, UPDATE, DELETE ON (empresas) to prueba
NOTICE:  GRANT INSERT, UPDATE, DELETE ON (provincias) to prueba
NOTICE:  GRANT INSERT, UPDATE, DELETE ON (eolicas) to prueba
NOTICE:  GRANT INSERT, UPDATE, DELETE ON (predicciones_de_viento) to prueba
NOTICE:  GRANT INSERT, UPDATE, DELETE ON (modelo_aerogeneradores) to prueba
NOTICE:  Asignando privilegios sobre procedimientos
NOTICE:  GRANT EXECUTE ON (prueba) to prueba
CALL

```

**4. (ORACLE) Realiza un procedimiento que genere un script que cree un rol conteniendo todos los permisos que tenga el usuario cuyo nombre reciba como**



**parámetro, le hayan sido asignados a aquél directamente o a través de roles. El nuevo rol deberá llamarse BackupPrivsNombreUsuario.**

```
create or replace procedure PrivilegiosDirectos(p_nombreusuario
DBA_SYS_PRIVS.GRANTEE%TYPE)
is
    v_privilegios varchar2(30);
    cursor c_privilegios is
    select PRIVILEGE
    from DBA_SYS_PRIVS
    where GRANTEE=p_nombreusuario;
    v_cursor c_privilegios%ROWTYPE;
begin
    for v_cursor in c_privilegios loop
        dbms_output.put_line('GRANT '||v_cursor.PRIVILEGE||' TO
BackupPrivs'||p_nombreusuario||';');
    end loop;
end PrivilegiosDirectos;
/

create or replace procedure PrivilegiosObjetosDirectos(p_nombreusuario
DBA_ROLE_PRIVS.GRANTEE%TYPE)
is
    v_privilegios varchar2(30);
    cursor c_privilegios is
    select *
    from dba_tab_privs
    where grantee=p_nombreusuario;
    v_cursor c_privilegios%ROWTYPE;
begin
    for v_cursor in c_privilegios loop
        dbms_output.put_line('GRANT '||v_cursor.privilege||' ON
'||v_cursor.table_name||' TO BackupPrivs'||p_nombreusuario||';');
    end loop;
end PrivilegiosObjetosDirectos;
/

create or replace procedure PrivilegiosSistema(p_nombreusuario
DBA_ROLE_PRIVS.GRANTEE%TYPE, p_nombreusuariooriginal
DBA_ROLE_PRIVS.GRANTEE%TYPE)
is
    v_privilegios varchar2(30);
    cursor c_privilegios is
    select *
    from dba_sys_privs
    where grantee=p_nombreusuario;
```

```

        v_cursor c_privilegios%ROWTYPE;
begin
    for v_cursor in c_privilegios loop
        dbms_output.put_line('GRANT '||v_cursor.PRIVILEGE||' TO
BackupPrivs'||p_nombreusuariooriginal||';');
    end loop;
end PrivilegiosSistema;
/

create or replace procedure PrivilegiosObjetos(p_nombreusuario
DBA_ROLE_PRIVS.GRANTEE%TYPE, p_nombreusuariooriginal
DBA_ROLE_PRIVS.GRANTEE%TYPE)
is
    v_privilegios varchar2(30);
    cursor c_privilegios is
    select *
    from dba_tab_privs
    where grantee=p_nombreusuario;
    v_cursor c_privilegios%ROWTYPE;
begin
    for v_cursor in c_privilegios loop
        dbms_output.put_line('GRANT '||v_cursor.privilege||' ON
'||v_cursor.table_name||' TO BackupPrivs'||p_nombreusuariooriginal||';');
    end loop;
end PrivilegiosObjetos;
/

create or replace procedure PrivilegiosPadres(p_rol
DBA_ROLE_PRIVS.GRANTEE%TYPE, p_nombreusuario DBA_ROLE_PRIVS.GRANTEE%TYPE)
is
    cursor c_privilegios is
    select *
    from dba_role_privs
    where grantee=p_rol;
begin
    for v_cursor in c_privilegios loop
        PrivilegiosSistema(v_cursor.GRANTED_ROLE, p_nombreusuario);
        PrivilegiosObjetos(v_cursor.GRANTED_ROLE, p_nombreusuario);
        PrivilegiosPadres(v_cursor.GRANTED_ROLE, p_nombreusuario);
    end loop;
end PrivilegiosPadres;
/

create or replace procedure DarPrivilegios(p_nombreusuario
DBA_ROLE_PRIVS.GRANTEE%TYPE)
is
begin
    dbms_output.put_line('CREATE ROLE BackupPrivs'||p_nombreusuario||';');
    PrivilegiosDirectos(p_nombreusuario);
    PrivilegiosObjetosDirectos(p_nombreusuario);
    PrivilegiosSistema(p_nombreusuario, p_nombreusuario);
    PrivilegiosObjetos(p_nombreusuario, p_nombreusuario);
    PrivilegiosPadres(p_nombreusuario, p_nombreusuario);
end DarPrivilegios;
/

```

```
SQL> exec DarPrivilegios('USRPRACTICA2');  
CREATE ROLE BackupPrivsUSRPRACTICA2;  
GRANT CREATE SESSION TO BackupPrivsUSRPRACTICA2;  
GRANT CREATE SESSION TO BackupPrivsUSRPRACTICA2;  
GRANT INSERT ON EMP TO BackupPrivsUSRPRACTICA2;  
GRANT INSERT ON DEPT TO BackupPrivsUSRPRACTICA2;  
GRANT CREATE VIEW TO BackupPrivsUSRPRACTICA2;
```

Procedimiento PL/SQL terminado correctamente.