

# Principios

Para la gestión de la seguridad, Oracle permite :

- definir los usuarios que se pueden conectar a la base de datos (con una identificación por el sistema operativo o por la base de datos);
- definir en qué tablespace(s) puede crear los objetos un usuario (eventualmente ninguno);
- limitar el uso de los recursos de sistema;
- imponer una política de gestión de contraseñas (expiración periódica, no reutilizar antes de un periodo de tiempo, etc.);
- definir los permisos de cada usuario dentro de la base de datos.

En una base de datos Oracle los permisos de los usuarios se administran con la noción de **permiso**.

Un permiso es el derecho para:

- ejecutar una sentencia SQL en general (por ejemplo, crear una tabla): es la noción de **permiso de sistema**;
- acceder a un objeto de otro usuario (por ejemplo, actualizar los datos de la tabla CLIENT): es la noción de **permiso de objeto**.

Los permisos se pueden asignar directamente a los usuarios o por medio de roles. Un **rol** es una agrupación de permisos (sistemas y objetos) con un nombre, que se puede asignar como tal a un usuario; de esta manera, este usuario recibe automáticamente los permisos contenidos en el rol. Los roles facilitan la gestión de los permisos.

Además, Oracle ofrece una funcionalidad de auditoría que permite trazar la actividad de los usuarios en la base de datos. Para saber más, puede consultar la documentación Oracle® Database Security Guide.

# Crear y modificar usuarios

## 1. Modo de identificación del usuario

Oracle puede identificar un usuario, o bien dejar que sea el sistema operativo quien lo haga. Los dos modos de identificación se pueden utilizar al mismo tiempo, en la misma base de datos.

### a. Identificación por Oracle

El usuario se conecta a la base de datos introduciendo un nombre y una contraseña. Oracle comprueba el nombre y la contraseña del usuario.

```
SQL> CONNECT oheu/rx239$  
Conectado.
```

Es posible utilizar las funcionalidades de gestión de contraseñas propuestas por Oracle.

### b. Identificación por el sistema operativo

El usuario se conecta a la base de datos sin introducir nombre ni contraseña.

```
SQL> CONNECT /  
Conectado.
```

Oracle no comprueba la contraseña, sino que simplemente controla que el nombre del usuario, a nivel del sistema operativo, corresponda con un nombre de usuario de la base de datos. La identificación inicial se ha realizado por el sistema operativo.

Las funcionalidades de gestión de contraseñas propuestas por Oracle no se pueden usar (no es Oracle quien gestiona la contraseña).

Para establecer el vínculo entre el nombre del usuario en el sistema operativo y el nombre del usuario en la base de datos, Oracle utiliza un prefijo definido por el argumento `OS_AUTHENT_PREFIX` (por defecto igual a `OPS$`).

Por ejemplo, el usuario con nombre `vdep` a nivel del sistema operativo se podrá conectar a la base por un `CONNECT /` solo si existe una cuenta Oracle `ops$vdep`.

En plataformas Windows el nombre de dominio o, de manera predeterminada, el nombre de la máquina, forma parte del nombre del usuario: `SRVWINORA\VDEP`, por ejemplo. Es el nombre completo el que se debe prefijar para formar el nombre de la cuenta Oracle (todo en mayúsculas): `OPS$SRVWINORA\VDEP`, por ejemplo.

El prefijo puede ser igual a una cadena vacía (`OS_AUTHENT_PREFIX = ""`); en este caso, el nombre del usuario a nivel del sistema operativo y el nombre del usuario en Oracle son idénticos.

Además, el argumento `REMOTE_OS_AUTHENT` se puede poner a `TRUE` para indicar si los usuarios remotos se identifican con este método (`FALSE`, para prohibirlo; es el valor por defecto). Poner el argumento `REMOTE_OS_AUTHENT` a `TRUE` puede ser peligroso si la red no es segura. Este argumento se dejó de usar desde la versión 11.

## 2. Creación de un usuario

La sentencia SQL CREATE USER permite crear un nuevo usuario.

### Sintaxis simplificada

```
CREATE USER nombre IDENTIFIED { BY la_contraseña | EXTERNALLY }  
[ DEFAULT TABLESPACE nombre_tablespace ]  
[ TEMPORARY TABLESPACE nombre_tablespace ]  
[ QUOTA { valor [K|M] | UNLIMITED } ON nombre_tablespace [,...] ]  
[ PROFILE nombre_perfil ]  
[ PASSWORD EXPIRE ]  
[ ACCOUNT { LOCK | UNLOCK } ];
```

### Ejemplo:

- Usuario identificado por el OS, solo con las cláusulas obligatorias:

```
CREATE USER "OPS$SRVWINORA\VDEP" IDENTIFIED EXTERNALLY;
```

- Usuario identificado por Oracle, con cláusulas adicionales:

```
CREATE USER oheu IDENTIFIED BY tempo  
    DEFAULT TABLESPACE data  
    QUOTA UNLIMITED ON data  
    PASSWORD EXPIRE;
```

Observe la sintaxis particular para especificar el nombre del usuario OPS\$SRVWINORA\ VDEP: las comillas son necesarias, porque el nombre contiene caracteres no permitidos (barra invertida). En adelante, siempre será necesario utilizar la misma sintaxis para gestionar este usuario.

Para que un nuevo usuario pueda conectarse de manera efectiva, además hay que darle permisos para hacerlo asignándole el permiso de sistema CREATE SESSION (consulte la sección Gestionar los permisos).

Por lo tanto, es posible tener cuentas para los usuarios sin que estos últimos tengan permiso para conectarse. Esta funcionalidad era interesante en la versión 7, porque permitía preparar las cuentas de usuario sin activarlas inmediatamente, o prohibir temporalmente a un usuario conectarse sin eliminar su cuenta.

Esta funcionalidad siempre se puede utilizar, pero es más sencillo bloquear/desbloquear explícitamente la cuenta (ACCOUNT LOCK | UNLOCK).

Las opciones de la sentencia SQL CREATE USER son:

### **nombre**

Nombre del usuario. El nombre del usuario debe respetar las reglas de nomenclatura de Oracle presentadas en la sección La base de datos del capítulo Las bases de la arquitectura Oracle. Si no es el caso, hay que poner el nombre entre comillas.

### **IDENTIFIED**

Esta cláusula indica si el usuario se identifica por el sistema operativo (EXTERNALLY) o por Oracle (BY la\_contraseña).

En caso de una identificación por Oracle, se especifica la contraseña inicial del usuario.

Para la contraseña, se deben respetar las reglas de nomenclatura de Oracle, salvo si se ha implementado la funcionalidad de control de la complejidad de las contraseñas (novedad de la versión 8 - se verá más adelante).

Desde la versión 11, por defecto las contraseñas son sensibles a mayúsculas/minúsculas (argumento `SEC_CASE_SENSITIVE_LOGON` igual a `TRUE`, por defecto). Si desea tener contraseñas no sensibles a mayúsculas/minúsculas basta con asignar el valor `FALSE` al argumento `SEC_CASE_SENSITIVE_LOGON` (pero no es aconsejable para la seguridad). Desde la versión 12, este argumento ya no se utiliza y Oracle recomienda dejarlo a `TRUE`.

## **DEFAULT TABLESPACE**

Esta cláusula indica en qué tablespace se crean, por defecto, los segmentos del usuario (es decir, si no hay ninguna cláusula `TABLESPACE` durante la creación del segmento).

Si la cláusula se omite, se asigna al usuario el tablespace permanente predeterminado de la base de datos (consulte la sección `Tablespace permanente` del capítulo `Tablespaces y archivos de datos`).

La noción de tablespace por defecto, no impide al usuario crear objetos en otro tablespace (si tiene una cuota en el tablespace en cuestión); permite simplemente especificar un tablespace por defecto si el usuario omite la cláusula `TABLESPACE` durante la creación de un segmento. Esta cláusula presenta interés, principalmente, para aquellos usuarios que pueden crear segmentos: desarrolladores, ingenieros de calidad y, de manera menos habitual, usuarios finales.

## **TEMPORARY TABLESPACE**

Esta cláusula indica en qué tablespace se crean los segmentos temporales del usuario (durante una operación de ordenación, por ejemplo). Puede indicar el nombre de un tablespace temporal o el nombre de un grupo de tablespaces temporales.

Si la cláusula se omite, el tablespace temporal predeterminado de la base de datos se asigna al usuario (consulte la sección `Tablespace temporal` del capítulo `Tablespaces y archivos de datos`).

## **QUOTA**

Esta cláusula indica en qué tablespace(s) puede crear objetos el usuario, y hasta qué límite.

La noción de `QUOTA` permite limitar el espacio que un usuario puede emplear en un tablespace con los segmentos que crea.

Esta funcionalidad solo afecta a los usuarios que pueden crear segmentos, y no a los usuarios finales de una aplicación, que se conforman con emplear los objetos ya existentes y que pertenecen generalmente a una cuenta distinta (de alguna manera, el "propietario" de la aplicación).

Por defecto, los usuarios no tienen ninguna cuota en ningún tablespace, salvo los `DBA`, que tienen una cuota ilimitada en todos los tablespaces.

Si un usuario tiene permiso para crear segmentos, hay que darle explícitamente una cuota en, al menos, un tablespace. El hecho de que se haya utilizado una cláusula `DEFAULT TABLESPACE` no asigna ninguna cuota en el tablespace en cuestión; son dos mecanismos diferentes.

En la práctica, solo hay que dar cuotas a los usuarios que lo necesitan (los desarrolladores, la cuenta "propietario")

de la aplicación) y solo en los tablespaces estrictamente necesarios y suficientes. En realidad, es mejor evitar dar cuotas a estos usuarios en el tablespace SYSTEM o el tablespace SYSAUX.

La noción de cuota queda sin objeto para el tablespace temporal y el tablespace de anulación.

Si un usuario quiere crear un segmento en un tablespace en el que no tiene cuota o supera la cuota asignado a este usuario, se obtiene un error:

- Ninguna cuota en el tablespace:

ORA-01950: no hay permisos en el tablespace 'DATA'

- Superar la cuota en el tablespace:

ORA-01536: se ha sobrepasado la cuota de espacio asignada al tablespace 'DATA'

## **PROFILE**

Esta cláusula indica el perfil asignado al usuario. La noción de perfil se presenta en la sección Utilizar perfiles.

## **PASSWORD EXPIRE**

Esta cláusula permite forzar una modificación de la contraseña durante la primera conexión (la contraseña del usuario ha expirado). Esta cláusula no tiene efecto si el usuario se identifica por el sistema operativo.

Si la cuenta se crea con la opción `PASSWORD EXPIRE`, se le pedirá al usuario durante su primera conexión que cambie la contraseña que se le ha asignado inicialmente.

## **ACCOUNT**

`LOCK`: la cuenta está bloqueada y la conexión prohibida (error ORA-28000: cuenta bloqueada).

`UNLOCK`: la cuenta no está bloqueada y la conexión está permitida (valor por defecto).

Si la cuenta se crea con la opción `LOCK`, la cuenta existe pero el usuario no se puede conectar. La sentencia SQL `ALTER USER` se podrá utilizar más tarde para desbloquear la cuenta del usuario (consulte la sección Crear y modificar usuarios).

## **3. Modificación de un usuario**

La sentencia SQL `ALTER USER` permite modificar un usuario.

### **Sintaxis simplificada**

```
ALTER USER nombre
[ IDENTIFIED { BY la_contraseña | EXTERNALLY } ]
[ DEFAULT TABLESPACE nombre_tablespace ]
[ TEMPORARY TABLESPACE nombre_tablespace ]
[ QUOTA { valor [K|M] | UNLIMITED } ON nombre_tablespace [,...] ]
[ PROFILE nombre_perfil ]
[ PASSWORD EXPIRE ]
```

```
[ ACCOUNT { LOCK | UNLOCK } ];
```

Las cláusulas son las mismas que para la creación.

#### Ejemplos:

- Modificación de la contraseña de un usuario:

```
ALTER USER oheu  
  IDENTIFIED BY tempo  
  PASSWORD EXPIRE;
```

- Modificación del tablespace por defecto y asignación de cuotas:

```
ALTER USER oheu  
  DEFAULT TABLESPACE test  
  QUOTA UNLIMITED ON test  
  QUOTA 10M ON data;
```

- Bloquear una cuenta:

```
ALTER USER oheu ACCOUNT LOCK;
```

- Desbloquear una cuenta:

```
ALTER USER oheu ACCOUNT UNLOCK;
```

El primer ejemplo permite modificar la contraseña de un usuario, forzándole a cambiarla durante su primera conexión; esta técnica se puede emplear si el usuario ha perdido su contraseña (el DBA no tiene ningún medio de conocer la contraseña de los usuarios).

El segundo ejemplo permite modificar el tablespace por defecto del usuario y asignarle cuotas en dos tablespaces.

El tercer ejemplo se puede utilizar para prohibir temporalmente la conexión a un usuario. Si actualmente está conectado, no se le desconecta.

El cuarto ejemplo se puede utilizar para autorizar de nuevo a un usuario a conectarse.

Disminuir una cuota o ponerla a 0 no elimina los objetos ya creados por el usuario.



Modificar la contraseña de SYS modifica la contraseña de SYSDBA, almacenada en el archivo de contraseñas (si se utiliza un archivo de contraseñas).

## 4. Eliminación de un usuario

La sentencia SQL DROP USER permite eliminar un usuario.

#### Sintaxis


```
DROP USER nombre [ CASCADE ];
```

### Ejemplo:

```
DROP USER "OPS$SRVWINORA\VDEP" CASCADE;
```

Si el usuario tiene objetos, la opción CASCADE debe estar presente para forzar la eliminación inicial de los objetos. Si el usuario tiene objetos y la opción CASCADE no está, se obtiene el error ORA-01922:

```
ORA-01922: CASCADE necesaria para eliminar 'OPS$SRVWINORA\VDEP'
```

 Es una sentencia DDL no hay ROLLBACK posible.

Un usuario actualmente conectado no se puede eliminar:

```
ORA-01940: imposible eliminar un usuario que está conectado
```

Antes de eliminar un usuario es posible exportar los objetos que le pertenecen; estos objetos se podrán volver a importar en otro esquema.

## 5. Encontrar información de los usuarios

Varias vistas del diccionario de datos permiten obtener información de los usuarios:

- DBA\_USERS: información de los usuarios;
- DBA\_TS\_QUOTAS: información de las cuotas de los usuarios.

Las columnas interesantes de las diferentes vistas se presentan a continuación.

### **DBA\_USERS**

USERNAME	Nombre del usuario.
USER_ID	Identificador del usuario.
PASSWORD	Contraseña (cifrada) del usuario.
ACCOUNT_STATUS	Estado de la cuenta (OPEN, LOCKED, UNLOCKED, EXPIRED, etc.).
LOCK_DATE	Fecha de bloqueo (si la cuenta está bloqueada).
EXPIRY_DATE	Fecha de expiración de la contraseña.
DEFAULT_TABLESPACE	Tablespace por defecto del usuario.
TEMPORARY_TABLESPACE	Tablespace temporal del usuario.
CREATED	Fecha de creación del usuario.
PROFILE	Perfil.
AUTHENTICATION_TYPE	Mecanismo utilizado para la autenticación.
LAST_LOGIN	Fecha y hora de la última conexión del usuario.
ORACLE_MAINTAINED	Y si la cuenta es una cuenta interna predefinida por Oracle (consulte la observación más adelante). N en caso contrario.

## **DBA\_TS\_QUOTAS**

TABLESPACE_NAME	Nombre del tablespace.
USERNAME	Nombre del usuario que tiene una cuota en el tablespace.
BYTES	Espacio, en bytes, utilizado actualmente por el usuario.
MAX_BYTES	Cuota, en bytes, del usuario en el tablespace (1).
BLOCKS	Espacio, en bloques, utilizado actualmente por el usuario.
MAX_BLOCKS	Cuota, en bloques, del usuario en el tablespace (1).

(1) -1 si la cuota es cuota UNLIMITED



Una base de datos Oracle contiene varias cuentas internas predefinidas utilizadas por algunas funcionalidades o algunos componentes; inicialmente están bloqueadas y tienen una contraseña expirada (ACCOUNT\_STATUS vale EXPIRED& LOCKED). Estas cuentas se enumeran y describen en la documentación Oracle Database 2 Day + Security Guide.



# Utilizar perfiles

## 1. Presentación

Un perfil es un conjunto de limitaciones de recursos, con un nombre que se puede asignar a un usuario.

Se pueden limitar los siguientes recursos:

- tiempo de CPU por llamada y/o por sesión;
- número de lecturas lógicas por llamada y/o por sesión;
- número de sesiones abiertas al mismo tiempo por un usuario;
- tiempo de inactividad por sesión;
- duración total de la sesión;
- cantidad de memoria privada en la SGA (solo en configuración de servidores compartidos).

Una lectura lógica corresponde a una lectura de un bloque durante una consulta, tanto si el bloque ya está presente en memoria (en la *Database Buffer Cache*) como si se lee de disco (en este caso, la lectura lógica también corresponde a una lectura física).

Desde la versión 8, los perfiles también se pueden utilizar para implementar una política de gestión de contraseñas.

Se pueden poner en marcha las siguientes funcionalidades:

- bloqueo de cuenta (y duración de bloqueo) cuando se supera un número determinado de intentos fallidos de conexión;
- tiempo de vida de las contraseñas (eventualmente con un periodo de cortesía);
- no reutilizar una contraseña antes de un plazo determinado de tiempo o antes de determinado número de cambios;
- complejidad de la contraseña.

El perfil llamado `DEFAULT` se crea automáticamente durante la creación de la base de datos. Este perfil se asigna por defecto a los usuarios. Por defecto, este perfil `DEFAULT` no impone ningún límite para los recursos; por el contrario, desde la versión 11, este perfil tiene límites para las contraseñas (como se verá más adelante).

La limitación de los recursos, con ayuda de los perfiles, no ofrece muchas posibilidades. Si desea controlar de manera más precisa la asignación de recursos (CPU, espacio de anulación, duración de inactividad) a los usuarios o grupos de usuarios, puede utilizar la *Database Resource Manager*. La implementación de esta funcionalidad se realiza gracias al paquete `DBMS_RESOURCE_MANAGER`. Para saber más, consulte la documentación Oracle® Database Administrator's Guide.

## 2. Creación de un perfil

La sentencia SQL `CREATE PROFILE` permite crear un nuevo perfil.

### Sintaxis

```
CREATE PROFILE nombre LIMIT
[ SESSIONS_PER_USER { valor | UNLIMITED | DEFAULT } ]
[ CPU_PER_SESSION { valor | UNLIMITED | DEFAULT } ]
```

```
[ CPU_PER_CALL { valor | UNLIMITED | DEFAULT } ]
[ CONNECT_TIME { valor | UNLIMITED | DEFAULT } ]
[ IDLE_TIME { valor | UNLIMITED | DEFAULT } ]
[ LOGICAL_READS_PER_SESSION { valor | UNLIMITED | DEFAULT } ]
[ LOGICAL_READS_PER_CALL { valor | UNLIMITED | DEFAULT } ]
[ COMPOSITE_LIMIT { valor | UNLIMITED | DEFAULT } ]
[ PRIVATE_SGA { valor [K|M] | UNLIMITED | DEFAULT } ]
[ FAILED_LOGIN_ATTEMPTS { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_LIFE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_REUSE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_REUSE_MAX { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_LOCK_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_GRACE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_VERIFY_FUNCTION { nombre_función | NULL | DEFAULT } ];
```

### Ejemplo:

```
CREATE PROFILE exploitation LIMIT
  SESSIONS_PER_USER 3
  IDLE_TIME 30
  FAILED_LOGIN_ATTEMPTS 3
  PASSWORD_LIFE_TIME 30
  PASSWORD_REUSE_TIME 180
  PASSWORD_LOCK_TIME UNLIMITED
  PASSWORD_GRACE_TIME 3
  PASSWORD_VERIFY_FUNCTION verif_mdp_exploitation;
```

Las limitaciones de recursos son las siguientes:

SESSIONS_PER_USER	Número de sesiones simultáneas.
CPU_PER_SESSION	CPU total por sesión (1/100 s).
CPU_PER_CALL	CPU total por llamada (1/100 s).
CONNECT_TIME	Duración total de conexión (minutos).
IDLE_TIME	Duración de la inactividad (minutos).
LOGICAL_READS_PER_SESSION	Número de lecturas lógicas por sesión.
LOGICAL_READS_PER_CALL	Número de lecturas lógicas por llamada.
PRIVATE_SGA	Cantidad de memoria privada en la SGA.
COMPOSITE_LIMIT	Suma ponderada de CPU_PER_SESSION, CONNECT_TIME, LOGICAL_READS_PER_SESSION y PRIVATE_SGA.

Para el límite COMPOSITE\_LIMIT, la vista RESOURCE\_COST permite consultar las ponderaciones utilizadas y la sentencia SQL ALTER RESOURCE COST modificarlas.

Las limitaciones relativas a las contraseñas son las siguientes:

FAILED_LOGIN_ATTEMPTS	Número de intentos erróneos de conexión permitidos antes del bloqueo de la cuenta, 10 en el perfil DEFAULT.
PASSWORD_LOCK_TIME	Duración del bloqueo (en días), 1 en el perfil DEFAULT.

PASSWORD_LIFE_TIME	Duración de la contraseña (en días), 180 en el perfil DEFAULT.
PASSWORD_GRACE_TIME	Periodo de cortesía después de la expiración de la contraseña (en días), 7 en el perfil DEFAULT.
PASSWORD_REUSE_TIME	Número de días durante el que no se puede reutilizar una contraseña.
PASSWORD_REUSE_MAX	Número de cambios de contraseña antes de que se pueda reutilizar una contraseña.
PASSWORD_VERIFY_FUNCTION	Función de comprobación de la complejidad de la contraseña.

Se pueden utilizar palabras clave para especificar el valor de un límite:

- UNLIMITED: ninguna limitación.
- DEFAULT: el argumento hereda el valor del perfil DEFAULT.

No se especifica ningún límite en un perfil que toma el valor DEFAULT.

Los límites PASSWORD\_REUSE\_TIME y PASSWORD\_REUSE\_MAX se utilizan de manera conjunta para controlar la reutilización de una contraseña antigua:

- Si especifica un valor para estos dos límites, entonces el usuario no puede reutilizar una contraseña mientras que la contraseña no se haya cambiado el número de veces indicado por el límite PASSWORD\_REUSE\_MAX, durante el número de días definido por el límite PASSWORD\_REUSE\_TIME. Por ejemplo, si PASSWORD\_REUSE\_TIME vale 60 y PASSWORD\_REUSE\_MAX vale 5, entonces el usuario puede reutilizar una contraseña al cabo de 60 días, a condición de que haya cambiado la contraseña, al menos, 5 veces.
- Si especifica un valor para alguno de los dos límites y UNLIMITED para el otro, entonces el usuario nunca puede reutilizar una contraseña.
- Si especifica UNLIMITED para los dos límites, entonces se ignoran y el usuario puede reutilizar una contraseña sin ninguna restricción.

No olvide que si no se define un límite o si se define con el valor DEFAULT, entonces se utiliza el valor especificado en el perfil DEFAULT.

Para los diferente límites especificados en días, es posible utilizar números decimales, representando fracciones de día (por ejemplo 1/24 = una hora).

El límite PASSWORD\_VERIFY\_FUNCTION permite especificar una función PL/SQL que se utilizará para comprobar si la contraseña introducida por el usuario respeta algunas reglas. Un valor NULL permite no utilizar la función de comprobación. Esta función debe aceptar tres argumentos de entrada (el nombre del usuario, su nueva contraseña y su contraseña antigua) y devuelve un valor booleano.

El script utlpwdmg.sql (repositorio %ORACLE\_HOME%\rdbms\admin o bien \$ORACLE\_HOME/rdbms/admin) contiene un ejemplo de función de comprobación, que se asigna al perfil DEFAULT si se ejecuta el script.

### 3. Modificación de un perfil

La sentencia SQL ALTER PROFILE permite modificar un perfil.

#### Sintaxis

```
ALTER PROFILE nombre LIMIT
[ SESSIONS_PER_USER { valor | UNLIMITED | DEFAULT } ]
[ CPU_PER_SESSION { valor | UNLIMITED | DEFAULT } ]
[ CPU_PER_CALL { valor | UNLIMITED | DEFAULT } ]
[ CONNECT_TIME { valor | UNLIMITED | DEFAULT } ]
[ IDLE_TIME { valor | UNLIMITED | DEFAULT } ]
[ LOGICAL_READS_PER_SESSION { valor | UNLIMITED | DEFAULT } ]
[ LOGICAL_READS_PER_CALL { valor | UNLIMITED | DEFAULT } ]
[ COMPOSITE_LIMIT { valor | UNLIMITED | DEFAULT } ]
[ PRIVATE_SGA { valor [K|M] | UNLIMITED | DEFAULT } ]
[ FAILED_LOGIN_ATTEMPTS { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_LIFE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_REUSE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_REUSE_MAX { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_LOCK_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_GRACE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_VERIFY_FUNCTION { nombre_función | NULL | DEFAULT } ];
```

#### Ejemplo:

- Modificación del perfil DEFAULT:

```
ALTER PROFILE default LIMIT
SESSIONS_PER_USER 3
IDLE_TIME 30
FAILED_LOGIN_ATTEMPTS 5;
-- los demás argumentos conservan el valor por defecto (UNLIMITED)
```

- Modificación de otro perfil:

```
ALTER PROFILE exploitation LIMIT
SESSIONS_PER_USER 5 -- pasa de 3 a 5
IDLE_TIME UNLIMITED -- eliminación del límite
FAILED_LOGIN_ATTEMPTS DEFAULT; -- toma el valor por defecto (5)
-- permanece inalterado
```

Las opciones son las mismas que para la sentencia SQL CREATE PROFILE.

La modificación de un perfil se asigna a los usuarios durante su próxima conexión; no se tiene en cuenta inmediatamente para los usuarios ya conectados.

Modificar el perfil DEFAULT también asigna los perfiles que tienen límites especificados a DEFAULT.

## 4. Asignación de un perfil a un usuario

Un perfil se puede asignar a un usuario:

- durante la creación del usuario (CREATE USER);
- durante una modificación del usuario (ALTER USER).

#### Ejemplos:

Durante la creación del usuario:

```
CREATE USER xgeo IDENTIFIED BY tempo
    TEMPORARY TABLESPACE temp
    PROFILE exploitation
    PASSWORD EXPIRE;
```

Durante una modificación del usuario:

- Asignación de un perfil:

```
ALTER USER oheu PROFILE exploitation;
```

- Reasignación del perfil DEFAULT:

```
ALTER USER oheu PROFILE DEFAULT;
```

La asignación de un nuevo perfil a los usuarios tiene efecto durante su próxima conexión.

Por defecto, un usuario se crea con el perfil DEFAULT.

## 5. Activación de la limitación de los recursos

Por defecto, el control de la limitación de los recursos no está activada. Crear perfiles y asignarlos a los usuarios no tiene ningún efecto.

Para activar el control de la limitación de los recursos hay que pasar el argumento RESOURCE\_LIMIT a TRUE (FALSE, por defecto):

```
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE [ cláusula_SCOPE ];
```

No olvide utilizar la cláusula SCOPE = BOTH para hacer que la modificación sea persistente en caso de reinicio de la base de datos.

Las funcionalidades de gestión de las contraseñas funcionan incluso si el argumento RESOURCE\_LIMIT es FALSE.

## 6. Eliminación de un perfil

La sentencia SQL DROP PROFILE permite eliminar un perfil.

### Sintaxis

```
DROP PROFILE nombre [ CASCADE ];
```

### Ejemplo:

```
DROP PROFILE exploitation CASCADE;
```

Si el perfil está asignado a los usuarios, la opción CASCADE debe estar presente. Si el perfil está asignado a los usuarios y la opción CASCADE no está, se obtiene el error ORA-02382:

```
ORA-02382: El perfil EXPLOITATION tiene usuarios, imposible realizar la eliminación sin CASCADE
```

El perfil DEFAULT se asigna reubicando a los usuarios afectados. La eliminación de un perfil asigna a los usuarios durante su próxima conexión. El perfil DEFAULT no se puede eliminar.

## 7. Encontrar la información de los perfiles

Varias vistas del diccionario de datos permiten obtener información de los perfiles:

- DBA\_USERS: información de los usuarios, incluyendo el perfil asignado (columna PROFILE);
- DBA\_PROFILES: información de los perfiles.

Las columnas interesantes de la vista DBA\_PROFILES se presentan a continuación.

### **DBA\_PROFILES**

PROFILE	Nombre del perfil.
RESOURCE_NAME	Nombre del recurso controlado.
RESOURCE_TYPE	Tipo del recurso controlado (KERNEL o PASSWORD).
LIMIT	Límite del recurso.

# Gestionar los permisos

## 1. Permiso de sistema

### a. Definición

Un permiso de sistema consiste en poder ejecutar una sentencia SQL en general, por ejemplo, crear una tabla.

Generalmente, cada sentencia SQL tiene, al menos, un permiso de sistema asociado, que tiene el mismo nombre que la sentencia SQL. Por ejemplo, la sentencia SQL `CREATE TABLE` tiene un permiso de sistema asociado `CREATE TABLE` (da permiso para crear una tabla en su propio esquema).

Algunos permisos de sistema toman el nombre de la sentencia SQL con la palabra clave `ANY`. En este caso, el permiso de sistema permite ejecutar la sentencia en cualquier esquema de la base de datos. Por ejemplo, el permiso de sistema `CREATE ANY TABLE` da el permiso para crear una tabla en cualquier esquema de la base de datos.

Cuando la sentencia SQL implicada no es relativa a los objetos de un esquema, no hay permiso `ANY` (`ANY` quiere decir en cualquier esquema): por ejemplo, el permiso para crear un `tablespace` es `CREATE TABLESPACE`; no hay `CREATE ANY TABLESPACE` (un `tablespace` no pertenece a un esquema).

Los permisos de sistema son fuente de poder y origen de peligro, sobre todo en lo que respecta a la gestión de usuarios y permisos (`CREATE USER`, `ALTER USER`, `DROP USER`, `GRANT ANY PRIVILEGE`, `GRANT ANY ROLE`) y la eliminación de objetos (`DROP ANY TABLE`, `DROP TABLESPACE`, etc.); por tanto, los permisos de sistema se deben asignar con cuidado (especialmente los permisos `ANY`).



Piense que si da el permiso `ALTER USER` a un usuario, éste podrá modificar las cuentas de usuario (cambiar las contraseñas, por ejemplo), incluida la de usted.

Algunos permisos de sistema particulares:

<code>CREATE SESSION</code>	Da permiso al usuario para conectarse.
<code>SELECT ANY DICTIONARY</code>	Da permiso al usuario para consultar cualquier objeto del diccionario de datos en el esquema <code>SYS</code> .

Si un usuario no tiene el permiso `CREATE SESSION`, se obtiene el error `ORA-01045` durante un intento de conexión:

```
ORA-01045: el usuario OHEU no tiene el permiso CREATE SESSION; conexión rechazada
```

El permiso `SELECT ANY DICTIONARY` es interesante porque permite dar a un usuario permiso para leer las vistas `DBA`, sin tener que ser `DBA`. En la versión 8 u 8i, el rol `SELECT_CATALOG_ROLE` se puede utilizar para conseguir el mismo objetivo; este rol sigue existiendo por razones de compatibilidad hacia atrás. En la versión 7 era necesario dar a los usuarios el permiso `SELECT ANY TABLE`, lo que podía presentar problemas de seguridad, porque el usuario podía leer cualquier esquema.

Los permisos de sistema se usan principalmente para controlar el uso de las sentencias DDL y, por tanto, están destinados a los administradores, desarrolladores, a la cuenta de propietario de una aplicación y, con menor frecuencia, al usuario final de una aplicación.

Si un usuario no tiene el permiso necesario para realizar una acción, se obtiene el error `ORA-01031`:

ORA-01031: permisos insuficientes

La vista `SYSTEM_PRIVILEGE_MAP` devuelve la lista de todos los permisos de sistema.

## **b. Asignación de un permiso de sistema a un usuario**

La sentencia SQL `GRANT` permite asignar un permiso de sistema.

### Sintaxis

```
GRANT nombre_permiso [,...]
TO { nombre_usuario | PUBLIC } [,...]
[ WITH ADMIN OPTION ];
```

### Ejemplo:

```
GRANT CREATE SESSION, CREATE TABLE TO oheu;
```

El permiso se puede asignar a un usuario o a todos los usuarios (`PUBLIC`).

La cláusula `WITH ADMIN OPTION` asigna al receptor el permiso de transmitir el permiso de sistema.

El permiso asignado se activa inmediatamente.

Para asignar un permiso de sistema, hay que haber recibido:

- el permiso en cuestión, con la cláusula `WITH ADMIN OPTION`;
- o el permiso de sistema `GRANT ANY PRIVILEGE`.

Se pueden asignar varios permisos a varios usuarios, en un único orden. Todos los permisos de sistema se pueden asignar de una vez con la palabra clave `ALL PRIVILEGES` (`GRANT ALL PRIVILEGES TO ...`). Esta posibilidad se debe manipular con mucha precaución.

## **c. Revocación de un permiso de sistema a un usuario**

La sentencia SQL `REVOKE` permite revocar un permiso de sistema.

### Sintaxis

```
REVOKE nombre_permiso [,...]
FROM { nombre_usuario | PUBLIC } [,...];
```

### Ejemplo:

```
REVOKE CREATE TABLE FROM oheu;
```

El permiso se revoca inmediatamente y nunca más se puede ejercer.

Para revocar un permiso de sistema, hay que haber recibido:



- el permiso en cuestión, con la cláusula `WITH ADMIN OPTION`;
- o el permiso de sistema `GRANT ANY PRIVILEGE`.

No se propaga en cascada la revocación de un permiso de sistema que se ha transmitido mediante la cláusula `WITH ADMIN OPTION`. Si un permiso se ha asignado a Ángel con la opción `WITH ADMIN OPTION` y Ángel lo ha transmitido a Iván, revocar el permiso de Ángel no tiene efecto alguno en el permiso transmitido por Ángel a Iván. La cláusula `WITH ADMIN OPTION`, por lo tanto, es doblemente peligrosa.

La sentencia `REVOKE` permite revocar solo los permisos que un usuario ha recibido de manera directa (no los permisos que tiene implícitamente a través `PUBLIC`). Es igual para `PUBLIC`: no puede revocar a `PUBLIC` un permiso no asignado a `PUBLIC`, pensando en eliminarlo de esta manera a todo el mundo.

Si un permiso se ha asignado a un usuario y a `PUBLIC`, la revocación del usuario no tiene efecto sobre la posibilidad del usuario para continuar ejerciendo el permiso: siempre lo tiene a través de `PUBLIC`.

Todos los permisos de sistema se pueden revocar al mismo tiempo, con la palabra clave `ALL PRIVILEGES` (`REVOKE ALL PRIVILEGES FROM ...`).

Si ha asignado un permiso con la opción `WITH ADMIN OPTION` y desea eliminar esta posibilidad de transmisión, hay que revocar el permiso y asignarlo de nuevo sin la opción.

#### **d. Los permisos de sistema `SYSDBA` y `SYSOPER`**

Ya hemos visto que los permisos `SYSDBA` y `SYSOPER` eran necesarios para realizar algunas operaciones de administración (inicio/parada/creación de la base de datos).

Estos permisos se pueden controlar por su pertenencia a un grupo particular del sistema operativo, o bien por un archivo de contraseñas.

En caso de usar un archivo de contraseñas, por defecto solo `SYS` ha recibido los permisos `SYSDBA` y `SYSOPER`.

Si el archivo de contraseñas se asocia exclusivamente a la base de datos (argumento `REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE`), es posible asignar alguno de estos permisos a otros usuarios.

La asignación y revocación se realizan con las sentencias SQL `GRANT` y `REVOKE`.

En la práctica, es muy raro asignar permisos `SYSOPER` y sobre todo `SYSDBA` (que da un control total sobre la base de datos) a otros usuarios; la cuenta `SYSDBA/SYSOPER` utilizada habitualmente es la cuenta `SYS` (está destinada a esto).

Para asignar el permiso `SYSDBA`, hay que estar conectado `AS SYSDBA`; para asignar el permiso `SYSOPER`, hay que estar conectado `AS SYSDBA` o `AS SYSOPER`.

La vista `V$PWFIL_USERS` permite enumerar los usuarios que han recibido los permisos `SYSDBA` o `SYSOPER`; esta vista siempre se vacía si `REMOTE_LOGIN_PASSWORDFILE=NONE`.



Existen otros permisos administrativos de este tipo, además de `SYSBACKUP` que apareció en la versión 12, que permiten realizar operaciones de copia de seguridad y restauración con la ayuda del Recovery Manager (RMAN) o de SQL\*Plus (consulte el capítulo Copia de seguridad y restauración).

## 2. Permiso de objeto

### a. Definición

Un permiso de objeto es el permiso para acceder a un objeto de otro usuario: por ejemplo, actualizar los datos de la tabla `CLIENT`.

Por defecto, solo el propietario de un objeto tiene el permiso de acceso. Para que otro usuario pueda acceder al objeto, el propietario del objeto le debe dar un permiso de objeto.

Los principales permisos de objeto son los siguientes:

Permiso	Tabla	Vista	Secuencia	Programa
SELECT	x	x	x	
INSERT	x	x		
UPDATE	x	x		
DELETE	x	x		
EXECUTE				x

En la tabla anterior, la columna "Programa" hace referencia a los procedimientos y funciones almacenados, y a los paquetes.

Éstos dan los siguientes permisos:

SELECT	Permiso de lectura de los datos (ejecución de la sentencia SQL <code>SELECT</code> ).
INSERT	Permiso de creación de los datos (ejecución de la sentencia SQL <code>INSERT</code> ).
UPDATE	Permiso de actualización de los datos (ejecución de la sentencia SQL <code>UPDATE</code> ).
DELETE	Permiso de eliminación de los datos (ejecución de la sentencia SQL <code>DELETE</code> ).
EXECUTE	Permiso de ejecución del programa (invocar al procedimiento, función o paquete desde otro programa).

Tener permiso sobre un objeto no elimina la obligatoriedad de cualificar al objeto con el nombre del propietario para acceder a él (en otro caso, Oracle piensa que usted intenta acceder a un objeto en su esquema). Para facilitar la escritura de consultas y hacer que el esquema sea propietario de los objetos transparentes, hay que utilizar sinónimos en la ocurrencia, en lugar de sinónimos públicos.

De manera recíproca, la existencia de un sinónimo, incluso público, no da ningún permiso sobre el objeto subyacente.

Los permisos de objeto están destinados a controlar el acceso a los objetos identificados. Por ejemplo: el permiso para crear un comando (es decir, el permiso para hacer un `INSERT` en la tabla `PEDIDO`) o el permiso para eliminar una ficha cliente (es decir, el permiso para hacer un `DELETE` en la tabla `CLIENTE`).

Principalmente se usan para permitir a los usuarios finales de una aplicación acceder, directamente o a través de una interfaz de usuario, a los objetos de la aplicación creados en una cuenta "propietaria" de la aplicación (ya que, por defecto, solo el propietario de un objeto tiene permiso de acceso).

El mensaje de error que devuelve Oracle cuando un usuario no tiene el permiso necesario para realizar una acción concreta sobre algún objeto se determina de manera diferente si el usuario tiene o no, al menos, un permiso sobre el objeto:

- Si el usuario no tiene ningún permiso sobre el objeto, Oracle devuelve el error ORA-00942: Tabla o vista no existe.
- Si el usuario tiene, al menos, un permiso sobre el objeto, Oracle devuelve el error ORA-01031: permisos insuficientes.

En el primer caso, Oracle considera que el usuario normalmente no tiene ningún medio para saber que el objeto al que accede existe: por tanto, indica que el objeto no existe. En el segundo caso, Oracle considera que el usuario puede saber que el objeto existe: por tanto, indica que el permiso es insuficiente.

## **b. Asignación de un permiso de objeto a un usuario**

La sentencia SQL GRANT permite asignar un permiso de objeto.

### Sintaxis

```
GRANT { nombre_permiso[(lista_columnas)] [,...] | ALL [PRIVILEGES] }
ON [nombre_esquema.]nombre_objeto
TO { nombre_usuario | PUBLIC } [,...]
[ WITH GRANT OPTION ];
```

### Ejemplo:

```
GRANT SELECT, INSERT, UPDATE(nombre,apellidos) ON miembro TO oheu;
```

Para los permisos INSERT y UPDATE es posible especificar una lista de columnas con el objetivo de limitar el permiso a las columnas indicadas.

La cláusula WITH GRANT OPTION da al receptor la posibilidad de transmitir el permiso de objeto.

La palabra clave ALL permite asignar todos los permisos. En caso de que el usuario que asigna todos los permisos no sea el propietario del objeto, sino un usuario que ha recibido algunos permisos sobre el objeto con la posibilidad de transmitirlos (cláusula WITH GRANT OPTION), la palabra clave ALL hace referencia solo a los permisos que el usuario ha recibido.

El permiso se puede asignar a un usuario o a todos los usuarios (PUBLIC).

Es posible asignar varios permisos a varios usuarios en un único orden; por el contrario, la asignación de los permisos de objeto se realiza objeto por objeto.

El permiso asignado se activa inmediatamente.

Para asignar un permiso de objeto hay que:

- ser propietario del objeto;
- haber recibido el permiso en cuestión con la cláusula WITH GRANT OPTION;
- o haber recibido el permiso de sistema ANY OBJECT PRIVILEGE.

Un usuario que transmite un permiso que ha recibido con la opción WITH GRANT OPTION debe cualificar el nombre del objeto con el nombre del propietario (ya que el objeto no le pertenece), salvo si existe algún sinónimo sobre el objeto.

Algunos permisos de sistema dan permisos de objeto implícitamente a todos los objetos. Por ejemplo: `SELECT ANY TABLE`.

El DBA tiene los permisos de sistema `ANY` indicados anteriormente; es la razón por la que puede acceder a cualquier objeto sin permiso de objeto. Desde la versión 9, también tiene permiso de sistema `ANY OBJECT PRIVILEGE`, que le permite asignar un permiso de objeto a cualquier objeto sin haber recibido el permiso en cuestión `WITH GRANT OPTION`. Antes de la versión 9, esto no era posible.

### c. Revocación de un permiso de objeto a un usuario

La sentencia SQL `REVOKE` permite revocar un permiso de objeto.

#### Sintaxis

```
REVOKE { nombre_permiso [,...] | ALL [PRIVILEGES] }  
ON [nombre_esquema.]nombre_objeto  
FROM { nombre_usuario | PUBLIC } [,...];
```

#### Ejemplo:

```
REVOKE INSERT, UPDATE ON cliente FROM oheu;
```

La sentencia `REVOKE` permite revocar a un usuario solamente aquellos permisos que ha recibido de manera directa (no los permisos que tiene implícitamente a través `PUBLIC`). Es igual para `PUBLIC`: no puede revocar a `PUBLIC` un permiso no asignado a `PUBLIC`, pensando que de esta manera se lo quita a todo el mundo.

Si un permiso se ha asignado a un usuario y a `PUBLIC`, el revocar el permiso al usuario no tiene ningún efecto; el usuario continúa ejerciendo el permiso: siempre lo tiene a través de `PUBLIC`. Todos los permisos de objeto se pueden revocar de una vez mediante la palabra clave `ALL` (`REVOKE ALL ON ... FROM ...`). Si ha asignado un permiso con la opción `WITH GRANT OPTION` y desea eliminar esta posibilidad de transmisión, hay que revocar el permiso y asignarlo de nuevo sin la opción. El permiso se elimina inmediatamente y no se puede ejercer más.

Para eliminar un permiso de objeto hay que:

- ser propietario del objeto;
- haber recibido el permiso en cuestión con la cláusula `WITH GRANT OPTION`;
- o haber recibido el permiso de sistema `ANY OBJECT PRIVILEGE`.

Existe propagación en cascada en la revocación de un permiso de objeto que ha sido transmitido gracias a la cláusula `WITH GRANT OPTION`. Si un permiso se ha asignado a Ángel con la opción `WITH GRANT OPTION` y Ángel lo ha transmitido a Iván, revocar el permiso de Ángel revoca también el de Iván. El funcionamiento no es el mismo que para los permisos de sistema.

### d. Permisos de las vistas y programas almacenados

Un usuario que tiene un permiso sobre una vista, no necesita tener permisos sobre los objetos manipulados por la vista.

Por defecto, es igual para los programas almacenados: el programa almacenado se ejecuta con los permisos del propietario (*definer rights*). Si es necesario, el programa almacenado se puede diseñar para ejecutarse con los

permisos de quien lo invoca (*invoker rights*).

El comportamiento deseado se define durante la creación del programa almacenado, gracias a la cláusula AUTHID.

#### Sintaxis

```
AUTHID { CURRENT_USER | DEFINER }
```

El modo de funcionamiento por defecto (permisos del propietario) es muy interesante porque permite utilizar las vistas y los programas almacenados como capa intermedia para el acceso a los objetos de la base de datos. Es posible construir una aplicación en la que la parte de cliente no acceda nunca directamente a las tablas de la base de datos, sino que pase por las vistas y/o los programas almacenados. Este tipo de enfoque permite, principalmente:

- ocultar la estructura real de las tablas y hacerla evolucionar con el mínimo impacto en la parte cliente;
- implementar reglas de gestión (controles, cálculos, seguridad, etc.) en el lado servidor.

### **e. Llamar a un objeto de otro esquema**

Incluso si un usuario tiene permisos sobre un objeto de un otro esquema, debe utilizar el nombre de su propietario como prefijo en el nombre del objeto, para poder acceder:

```
SELECT * FROM diane.miembro;
```

Se pueden definir sinónimos públicos para simplificar la escritura de las consultas y hacerlas independientes del nombre del propietario:

```
CREATE PUBLIC SYNONYM miembro FOR diane.miembro;
```

Cuando la consulta `SELECT * FROM miembro` se ejecuta, Oracle mira en primer lugar en el esquema actual si existe un objeto llamado `miembro`. Si es el caso, lo usa. Si no es el caso, busca un sinónimo público con este nombre. Si lo encuentra, reemplaza el sinónimo por su definición.

Desde la versión 8i, también es posible "ubicar" su sesión en el esquema de otro usuario. Cuando se hace referencia a un objeto en una sentencia SQL, Oracle mira si existe un objeto como este en el otro esquema antes de buscar un posible sinónimo público.

#### Ejemplo:

```
ALTER SESSION SET CURRENT_SCHEMA = diane;
```



Observe que las dos técnicas no dan el permiso en sí; son técnicas de resolución de nombres. Una vez que el nombre está resuelto, Oracle mira si el usuario tiene los permisos necesarios para acceder al objeto.

### **f. Ir más allá en la gestión de los permisos**

Oracle ofrece las funcionalidades *Virtual Private Database* (VPD) y *Fine-Grained Access* (FGA), que permiten utilizar mecanismos de filtrado de registros en las tablas. Estas funcionalidades se describen en la documentación Oracle® Database Security Guide; se basan en el uso del paquete DBMS\_RLS.

## 3. Rol

### a. Definición

Un rol es una agrupación de permisos (sistema y objeto) con un nombre, que se puede asignar a un usuario. Todos los permisos agrupados en el rol se asignan al mismo tiempo al usuario. Los roles permiten simplificar la gestión de los permisos.

Las principales características de los roles son las siguientes:

- Un rol se puede asignar a otro rol.
- Un usuario puede tener varios roles.
- Un rol no pertenece a nadie.

La implementación se realiza en tres etapas:

- 1 - creación del rol;
- 2 - asignación de los permisos (sistema y objeto) al rol;
- 3 - asignación del rol a los usuarios (o bien a otros roles).

Veremos también que, desde la versión 12, un rol se puede asignar a un programa almacenado y, de esta manera, los permisos asignados al rol en cuestión solo se podrán utilizar durante la ejecución del programa implicado. Esta funcionalidad se llama "seguridad basada en código" (*Code-Based Security*) o "control de acceso basado en código" (*Code-Based Access Control*).

### b. Creación de un rol

La sentencia SQL `CREATE ROLE` permite crear un rol.

#### Sintaxis

```
CREATE ROLE nombre  
[ IDENTIFIED { BY la_contraseña | EXTERNALLY | USING nombre_paquete }  
| NOT IDENTIFIED ];
```

#### Ejemplo:

```
CREATE ROLE mailing;
```

Las opciones son:

<code>IDENTIFIED BY la_contraseña</code>	Indica que se necesita una contraseña para activar el rol.
<code>IDENTIFIED EXTERNALLY</code>	Indica que se necesita una identificación externa para activar el rol.
<code>IDENTIFIED USING nombre_paquete</code>	Indica que solo el paquete puede activar el rol.
<code>NOT IDENTIFIED</code>	Indica que no es necesaria ninguna identificación para activar el rol. Es el valor por defecto.

Para crear un rol hay que tener el permiso de sistema `CREATE ROLE`.

Durante la creación de un rol es posible concretar el mecanismo por el que se podrá activar el rol: una contraseña, una identificación externa (sistema operativo) o un paquete. El mecanismo de activación se presentará en este capítulo, en la sección Rol - Activación o desactivación de un rol.

La sentencia SQL `ALTER ROLE` permite modificar un rol y el modo de identificación para poder activarlo.

#### Sintaxis

```
ALTER ROLE nombre
[ IDENTIFIED { BY la_contraseña | EXTERNALLY | USING nombre_paquete }
| NOT IDENTIFIED ];
```

### **c. Asignación de un permiso a un rol**

La sentencia SQL `GRANT` permite asignar permisos de sistema o de objeto a un rol.

#### Sintaxis para los permisos de sistema

```
GRANT nombre_permiso [... ]
TO nombre_rol [... ]
[ WITH ADMIN OPTION ];
```

#### Sintaxis para los permisos de objeto

```
GRANT { nombre_permiso[(lista_columnas)] [... ] | ALL [PRIVILEGES] }
ON [nombre_esquema.]nombre_objeto
TO nombre_rol [... ];
```

#### Ejemplo:

```
GRANT CREATE SESSION, CREATE TABLE TO mailing;
GRANT SELECT, INSERT ON miembro TO mailing;
```

La sintaxis es la misma que para la asignación directa a un usuario, con excepción de la cláusula `WITH GRANT OPTION`, que no está permitida para la asignación de un permiso de objeto a un rol.

Los permisos asignados se activan inmediatamente para los usuarios conectados que tienen el rol activo. La noción de rol activo se presenta un poco más adelante en este capítulo, Gestionar los permisos, sección Rol - Activación o desactivación de un rol.

Cualquier usuario puede asignar un permiso a un rol, siempre que tenga la capacidad de asignar el permiso; éste es el motivo por el que el rol no pertenece a nadie. Para asignar un permiso de sistema a un rol hay que haber recibido el permiso correspondiente mediante la cláusula `WITH ADMIN OPTION` o tener el permiso de sistema `GRANT ANY PRIVILEGE`. Para asignar un permiso de objeto a un rol hay que ser propietario del objeto, haber recibido el permiso correspondiente mediante la cláusula `WITH GRANT OPTION` o tener el permiso de sistema `ANY OBJECT PRIVILEGE`.

### **d. Revocación de un permiso a un rol**

La sentencia SQL REVOKE permite revocar permisos de sistema o de objeto a un rol.

#### Sintaxis para los permisos de sistema

```
REVOKE nombre_permiso [ ,...]  
FROM nombre_rol [ ,...];
```

#### Sintaxis para los permisos de objeto

```
REVOKE { nombre_permiso [ ,... ] | ALL [PRIVILEGES] }  
ON [nombre_esquema.]nombre_objeto  
FROM nombre_rol [ ,...];
```

#### Ejemplo:

```
REVOKE CREATE TABLE FROM mailing;  
REVOKE UPDATE ON miembro FROM mailing;
```

La sintaxis es la misma que para la revocación directa para un usuario. De manera general, la sentencia SQL REVOKE solo permite eliminar a un "destino" (usuario, rol o PUBLIC) que haya sido asignado explícitamente a este "destino". Por ejemplo, si se ha asignado un permiso a un rol, no es posible eliminarlo directamente para un usuario que ha recibido el rol.

Preste atención a la mezcla de asignación directa y asignación a un rol: si un permiso se asigna a un rol y el rol al usuario y además, en paralelo, el permiso se asigna directamente al usuario, revocar el permiso al usuario le permitirá seguir ejerciéndolo (a través del rol).

Los permisos se revocan inmediatamente y ya no se pueden ejercer más por parte de los usuarios conectados que tienen el rol activo. La noción de rol activo se presenta un poco más adelante en este capítulo, sección Rol - Activación o desactivación de un rol.

Cualquier usuario puede revocar un permiso de un rol, siempre que pueda revocar el permiso (el rol no pertenece a nadie). Para revocar un permiso de sistema de un rol hay que haber recibido el permiso correspondiente mediante la cláusula WITH ADMIN OPTION o bien tener el permiso de sistema GRANT ANY PRIVILEGE. Para revocar un permiso de objeto de un rol hay que ser el propietario del objeto, haber recibido el permiso en cuestión mediante la cláusula WITH GRANT OPTION o tener el permiso de sistema ANY OBJECT PRIVILEGE.

### **e. Asignación de un rol a un usuario o a otro rol**

La sentencia SQL GRANT permite asignar un rol a un usuario o a otro rol.

#### Sintaxis

```
GRANT nombre_rol [ ,...]  
TO { nombre_usuario | PUBLIC | nombre_rol } [ ,...]  
[ WITH ADMIN OPTION ];
```

#### Ejemplo:

```
GRANT mailing TO oheu;
```

La sintaxis es la misma que para la asignación de un permiso de sistema.



Para poder asignar un rol hay que haber recibido el rol en cuestión mediante la cláusula `WITH ADMIN OPTION` o tener el permiso de sistema `GRANT ANY ROLE`. El creador de un rol no es propietario del rol, sino que se le asigna mediante la opción `WITH ADMIN OPTION`; por tanto, puede asignar el rol que ha creado.

El rol asignado no se activa inmediatamente si el usuario ya está conectado.

La cláusula `WITH ADMIN OPTION` también da el permiso para modificar (sentencia `SQL ALTER ROLE`) y para eliminar el rol (sentencia `SQL DROP ROLE`).

Un usuario puede tener varios roles; en este caso, los permisos se acumulan (no hay permiso "negativo").

#### **f. Revocación de un rol a un usuario o a otro rol**

La sentencia `SQL REVOKE` permite revocar un rol.

##### Sintaxis

```
REVOKE nombre_rol [,...]
FROM { nombre_usuario | PUBLIC | nombre_rol } [,...];
```

##### Ejemplo:

```
REVOKE mailing FROM oheu;
```

La sintaxis es la misma que para la revocación de un permiso de sistema. Para poder eliminar un rol hay que haber recibido el rol en cuestión mediante la cláusula `WITH ADMIN OPTION` o tener el permiso de sistema `GRANT ANY ROLE`. El creador del rol que ha recibido este último con la cláusula `WITH ADMIN OPTION` puede eliminar el rol.

Cuando un rol se revoca, los usuarios conectados con el rol activo pueden continuar ejerciendo los permisos asociados hasta finalizar la sesión o hasta desactivar el rol.

#### **g. Eliminación de un rol**

La sentencia `SQL DROP ROLE` permite eliminar un rol.

##### Sintaxis

```
DROP ROLE nombre_rol;
```

##### Ejemplo:

```
DROP ROLE mailing;
```

Para eliminar un rol hay que haber recibido el rol en cuestión con la cláusula `WITH ADMIN OPTION` o tener el permiso de sistema `DROP ANY ROLE`.

El rol se elimina inmediatamente a los usuarios; los permisos asociados ya no se pueden ejercer más.

#### **h. Activación o desactivación de un rol**

Un rol asignado a un usuario (directamente o a través un otro rol), se activa automáticamente por defecto durante la conexión del usuario.

Si el usuario se conecta durante la asignación, la activación inmediata no es automática. El usuario puede activar el rol gracias a la sentencia SQL SET ROLE.

Adicionalmente, entre los roles asignados al usuario, es posible definir los que se activan automáticamente durante la conexión del usuario. Estos son los roles por defecto, definidos mediante una sentencia SQL ALTER USER. A continuación, el usuario puede activar los demás gracias a la sentencia SQL SET ROLE.

Utilizar varios roles sin que estén todos activos al mismo tiempo es interesante porque:

- existe un argumento, MAX\_ENABLED\_ROLES, que limita el número de roles activos al mismo tiempo para un usuario. Si un usuario es susceptible de emplear un número de roles superior a este límite, es posible desactivar algunos para activar otros. El argumento MAX\_ENABLED\_ROLES ya no se usa y solo se conserva por razones de compatibilidad hacia atrás. Vale 150 por defecto, y este valor está codificado en duro en Oracle; por lo tanto, modificar el valor de este argumento no tiene ningún efecto. Cuando este argumento se elimine definitivamente, la limitación en el número de roles activos al mismo tiempo dejará de existir.
- se puede asignar roles, protegidos por contraseñas, a los usuarios, pero por defecto inactivos y sin dar la contraseña al usuario; se trata de aplicaciones que activan los roles que necesitan, proporcionando la contraseña. De esta manera, fuera de la aplicación en cuestión, el usuario no tiene ningún medio para activar y utilizar el rol (y eventualmente, cometer errores).

La sentencia SQL ALTER USER permite definir los roles por defecto de un usuario.

#### Sintaxis

```
ALTER USER nombre_usuario DEFAULT ROLE
{ nombre_rol [,...] | ALL [ EXCEPT nombre_rol [,...] ] | NONE };
```

#### Ejemplo:

```
ALTER USER oheu DEFAULT ROLE mailing;
ALTER USER vdep DEFAULT ROLE ALL EXCEPT mailing;
```

Las opciones son:

ALL	Todos los roles asignados al usuario están activados por defecto. La cláusula EXCEPT permite eliminar algunos.
NONE	Ninguno de los roles asignados al usuario está activo por defecto.

Esta sentencia SQL anula y sustituye la situación actual de los roles por defecto: no añade o elimina los roles a la lista actual. Los roles asignados a un usuario que ya tiene roles por defecto no se definen por defecto.

La sentencia SQL SET ROLE permite activar o desactivar un rol.

#### Sintaxis

```
SET ROLE
{ nombre_rol [ IDENTIFIED BY la_contraseña ] [,...]
| ALL [ EXCEPT nombre_rol [,...] ] | NONE };
```

#### Ejemplo:

---

```
-- El usuario VDEP activa el rol MAILING
SET ROLE mailing;
```

Las opciones son:

IDENTIFIED BY	Indica la contraseña que permite activar el rol.
ALL	Todos los roles asignados al usuario están activados. La cláusula EXCEPT permite eliminar algunos.
NONE	Ninguno de los roles asignados al usuario está activado (por tanto, todos los roles desactivados).

Los roles se han debido asignar inicialmente a un usuario; por tanto, no es posible auto asignarse un rol activándolo (afortunadamente).

Esta sentencia SQL anula y sustituye los roles actualmente activos (no añade).

La opción ALL no se puede utilizar para roles protegidos con contraseña.

Los roles definidos con la opción IDENTIFIED USING nombre\_paquete solo se pueden activar a partir del paquete especificado.

El procedimiento SET\_ROLE del paquete DBMS\_SESSION permite hacer lo mismo (consulte la documentación "PL/SQL Packages and Types Reference").

## i. Limitación de los roles

Para desarrollar un objeto (una vista o un programa almacenado) que utiliza objetos de otro esquema hay que haber recibido los permisos de los objetos de manera directa y no a través de un rol.

Por otra parte, durante la ejecución de un programa almacenado, los roles activos del usuario que lo invoca solo se tienen en cuenta si el programa almacenado está diseñado para ejecutarse con los permisos del usuario que lo invoca (cláusula AUTHID CURRENT\_USER).

## j. Roles predefinidos

De manera estándar, Oracle ofrece un gran número de roles predefinidos, entre los que encontramos:

CONNECT	Autoriza la conexión (contiene solo el permiso de sistema CREATE SESSION).
RESOURCE	Permite crear los principales objetos de un esquema (tablas, triggers, secuencias, programas almacenados, pero no las vistas).
DBA	Da todos los permisos de sistema, con la opción WITH ADMIN OPTION.
EM_EXPRESS_BASIC	Permite conectarse a EM Express y ver las páginas en modo solo lectura.
EM_EXPRESS_ALL	Permite conectarse a EM Express y utilizar todas las funcionalidades (consulta y modificación).

Las vistas DBA\_SYS\_PRIVS y DBA\_TAB\_PRIVS permiten conocer los permisos agrupados en estos roles predefinidos. Oracle aconseja no utilizar los roles predefinidos CONNECT, RESOURCE y DBA, sino crear sus propios roles. Desde la versión 10.2 (10 g Release 2), el rol CONNECT solo contiene el permiso de sistema CREATE SESSION. Antes de esta versión, este rol contenía otros permisos que permitían crear los principales

objetos de un esquema (tabla, vista, etc.) o modificar la sesión (permiso de sistema ALTER SESSION). Desde la versión 10.2, si necesita asignar estos permisos a un usuario, el rol CONNECT no es suficiente; por el contrario, puede asignar estos permisos directamente al usuario o a través de un rol específico que se cree.

De la misma manera, a lo largo de las versiones, el rol RESOURCE cada vez es más restrictivo. Desde la versión 12 este rol ya no contiene el permiso de sistema UNLIMITED TABLESPACE.

## k. Seguridad basada en código

Desde la versión 12 se puede asignar un rol a un programa almacenado, de manera que los permisos asignados al rol en cuestión solo se puedan utilizar durante la ejecución del programa implicado. Esta funcionalidad se llama "seguridad basada en código" (*Code-Based Security*) o "control de acceso basado en código" (*Code Based Access Control*).

Veremos que esta funcionalidad puede ser interesante, pero de manera diferente, tanto para los programas almacenados que se ejecutan con los permisos del propietario (*definer rights*) como para los que se ejecutan con los permisos del usuario que lo invoca (*invoker rights*).

### Sintaxis para asignar un rol a un programa almacenado

```
GRANT nombre_rol [,...]
TO {PROCEDURE | FUNCTION | PACKAGE} nombre_programa_almacenado [,...];
```

### Sintaxis para retirar un rol de un programa almacenado

```
REVOKE ALL | nombre_rol [,...]
FROM {PROCEDURE | FUNCTION | PACKAGE} nombre_programa_almacenado [,...];
```

Un rol se puede asignar a un programa almacenado por el usuario SYS o por el propietario del programa; si no es el caso, se obtiene el error ORA-28702: La unidad de programa xxx no pertenece al usuario que concede el permiso. Por otra parte, de manera inicial, se debe haber asignado el rol directamente al propietario del programa almacenado; si no es el caso, se obtiene el error ORA-01924: el rol 'xxx' no está permitido o no existe.

En caso de un programa almacenado que se ejecuta con los permisos del propietario (*definer rights*), que es el caso por defecto, esta funcionalidad es interesante solo para el código que se ejecute en SQL dinámico. De hecho, para que un programa almacenado se pueda compilar sin error, todos los permisos necesarios para el código SQL estático se deben asignar directamente al propietario y no por medio de un rol (consulte la sección Limitación de los roles de este capítulo). En caso de SQL dinámico, los permisos se comprueban durante la ejecución y Oracle tiene en cuenta los permisos asignados directamente al usuario y los permisos concedidos a los roles asignados al programa almacenado. Esto permite salvar la limitación de los roles que se ha mencionado anteriormente (los permisos asignados al rol ahora se tienen en cuenta durante la ejecución del programa almacenado), pero solo para algunos programas, no para todos, lo que puede permitir controlar mejor los problemas de seguridad relacionados con la inyección de SQL, por ejemplo.

En el caso de un programa almacenado que se ejecuta con los permisos del usuario que lo invoca (*invoker rights*), esta funcionalidad es interesante porque permite limitar los permisos asignados al usuario que ejecuta el programa almacenado. El rol se asigna solo al propietario del programa almacenado y al programa almacenado. Solo está activo durante la ejecución del programa almacenado, sin que sea necesario asignarlo al usuario que ejecuta el programa almacenado: por tanto, este último no puede ejercer directamente los permisos asignados al rol.

## 4. Encontrar información de los permisos

### a. Permisos de sistema

Varias vistas del diccionario de datos permiten obtener información de los permisos de sistema:

- `DBA_SYS_PRIVS`: permisos de sistema asignados a los usuarios (o a los roles);
- `SESSION_PRIVS`: permisos de sistema activos actualmente en la sesión (obtenidos directamente o a través de un rol);
- `SYSTEM_PRIVILEGE_MAP`: lista de todos los permisos de sistema.

Las columnas interesantes de las diferentes vistas se presentan a continuación.

#### **DBA\_SYS\_PRIVS**

GRANTEE	Nombre del usuario o del rol que ha recibido el permiso de sistema.
PRIVILEGE	Permiso de sistema recibido.
ADMIN_OPTION	Permiso recibido con la cláusula <code>WITH ADMIN OPTION</code> (YES o NO).

#### **SESSION\_PRIVS**

PRIVILEGE	Nombre del permiso.
-----------	---------------------

#### **SYSTEM\_PRIVILEGE\_MAP**

NAME	Nombre del permiso.
------	---------------------

### b. Permisos de objeto

Varias vistas del diccionario de datos permiten obtener información de los permisos de objeto:

- `DBA_TAB_PRIVS`: permisos de objeto asignados a los usuarios (o a los roles) sobre la totalidad del objeto;
- `DBA_COL_PRIVS`: permisos de objeto asignados a los usuarios (o a los roles) únicamente sobre algunas columnas del objeto;
- `TABLE_PRIVILEGE_MAP`: lista de todos los permisos de objeto.

Las columnas interesantes de las diferentes vistas se presentan a continuación.

#### **DBA\_TAB\_PRIVS**

GRANTEE	Nombre del usuario o del rol que ha recibido el permiso de objeto.
OWNER	Nombre del usuario propietario del objeto.
TABLE_NAME	Nombre del objeto (no es forzosamente una tabla, a pesar del nombre).
GRANTOR	Nombre del usuario que ha asignado el permiso.

PRIVILEGE	Permiso del objeto recibido.
GRANTABLE	Permiso recibido con la cláusula WITH GRANT OPTION (YES o NO).

### **DBA\_COL\_PRIVS**

GRANTEE	Nombre del usuario o del rol que ha recibido el permiso de objeto.
OWNER	Nombre del usuario propietario del objeto.
TABLE_NAME	Nombre del objeto (tabla o vista).
COLUMN_NAME	Nombre de la columna.
GRANTOR	Nombre del usuario que ha asignado el permiso.
PRIVILEGE	Permiso del objeto recibido.
GRANTABLE	Permiso recibido con la cláusula WITH GRANT OPTION (YES o NO).

### **TABLE\_PRIVILEGE\_MAP**

NAME	Nombre del permiso.
------	---------------------

## **c. Roles**

Varias vistas del diccionario de datos permiten obtener información de los roles:

- DBA\_ROLES: lista de los roles existentes en la base de datos;
- DBA\_APPLICATION\_ROLES: descripción de los roles activados por un paquete;
- DBA\_CODE\_ROLE\_PRIVS: lista de los roles asignados a programas almacenados.
- DBA\_SYS\_PRIVS: permisos de sistema asignados a los roles (o a los usuarios), ver más atrás;
- DBA\_TAB\_PRIVS: permisos de objeto asignados a los roles (o a los usuarios), sobre la totalidad del objeto, ver más atrás;
- DBA\_COL\_PRIVS: permisos de objeto asignados a los roles (o a los usuarios), únicamente sobre algunas columnas del objeto, ver más atrás;
- DBA\_ROLE\_PRIVS: roles asignados a los usuarios o a los roles;
- ROLE\_SYS\_PRIVS: permisos de sistema asignados a los roles (solo para los roles a los que el usuario tiene acceso);
- ROLE\_TAB\_PRIVS: permisos de objeto asignados a los roles (solo para los roles a los que el usuario tiene acceso);
- ROLE\_ROLE\_PRIVS: roles asignados a otros roles (solo para los roles a los que el usuario tiene acceso);
- SESSION\_ROLES: roles activos actualmente en la sesión.

Las columnas interesantes de las diferentes vistas se presentan a continuación.

### **DBA\_ROLES**

ROLE	Nombre del rol.
PASSWORD_REQUIRED	Indica si es necesaria una contraseña para activar el rol (YES, NO o

GLOBAL).

### **DBA\_APPLICATION\_ROLES**

ROLE	Nombre del rol.
SCHEMA	Esquema del paquete utilizado para la activación.
PACKAGE	Nombre del paquete utilizado para la activación.

### **DBA\_CODE\_ROLE\_PRIVS**

OWNER	Propietario del programa almacenado.
OBJECT_NAME	Nombre del programa almacenado.
OBJECT_TYPE	Tipo (PROCEDURE, FUNCTION, PACKAGE) del programa almacenado.
ROLE	Rol asignado al programa almacenado.

### **DBA\_ROLE\_PRIVS**

GRANTEE	Nombre del usuario o del rol que ha recibido el rol.
GRANTED_ROLE	Nombre del rol recibido.
ADMIN_OPTION	Rol recibido con la cláusula WITH ADMIN OPTION (YES o NO).

### **ROLE\_SYS\_PRIVS**

ROLE	Nombre del rol.
PRIVILEGE	Nombre del permiso de sistema recibido a través del rol.
ADMIN_OPTION	Permiso recibido con la cláusula WITH ADMIN OPTION (YES o NO)

### **ROLE\_TAB\_PRIVS**

ROLE	Nombre del rol.
OWNER	Nombre del usuario propietario del objeto.
TABLE_NAME	Nombre del objeto (no necesariamente una tabla, a pesar del nombre).
COLUMN_NAME	Nombre de la columna (si aplica).
PRIVILEGE	Permiso del objeto recibido.

### **ROLE\_ROLE\_PRIVS**

ROLE	Nombre del rol.
GRANTED_ROLE	Nombre del rol asignado al rol.
ADMIN_OPTION	Rol recibido con la cláusula WITH ADMIN OPTION (YES o NO).

**SESSION\_ROLES**

ROLE	Nombre del rol.
------	-----------------



# Resumen

## 1. Las diferentes tipos de cuentas

En general, una base Oracle contiene cuatro tipos de cuentas.

### Administración

Una cuenta como esta tiene todos los permisos de sistema necesarios, fundamentalmente para la gestión de las estructuras de almacenamiento y la gestión de los usuarios. Además, las cuentas de administración tienen un acceso completo al diccionario de datos.

Estos permisos se pueden obtener por medio del rol DBA o de un rol equivalente.

### Desarrollo/alojamiento del esquema de aplicaciones

Una cuenta como esta tiene los permisos de sistema necesarios para la creación de los diferentes tipos de objetos (tablas, vistas, procedimientos...) y tiene una cuota sobre, al menos, un tablespace. Los permisos de sistema necesarios se pueden obtener por medio de los roles CONNECT y RESOURCE o a través de un rol equivalente que usted haya creado (aconsejable).

Para las cuentas de desarrollo, puede ser aconsejable prever un tablespace dedicado y definir una cuota solo sobre este tablespace. En la situación ideal, es preferible utilizar una base de datos a parte para el desarrollo.

La cuenta "propietaria" de una aplicación generalmente tiene cuotas ilimitadas en los tablespaces (de tablas e índices) dedicados a la aplicación.

### Usuario final

Una cuenta como esta necesita pocos permisos de sistema: CREATE SESSION (obligatorio), ALTER SESSION (algunas veces necesaria) y generalmente es todo. Por el contrario, tiene los permisos de objeto para los objetos del esquema de aplicaciones, generalmente por medio de un rol.



El permiso de sistema ALTER SESSION no es necesario para ejecutar la sentencia SQL ALTER SESSION SET CURRENT\_SCHEMA o las sentencias SQL que modifican los argumentos NLS de la sesión (como ALTER SESSION SET nls\_date\_format = ... por ejemplo).

Las cuentas de los usuarios finales no necesitan ninguna cuota en los tablespaces. Acceden a los objetos del esquema de la aplicación gracias a los permisos de objeto y la escritura de las consultas se simplifica usando sinónimos públicos o ejecutando la sentencia SQL ALTER SESSION SET CURRENT\_SCHEMA.

Los perfiles también se pueden utilizar para controlar el uso de algunos recursos o implementar una política de gestión de contraseñas.

## 2. Algunos consejos para dotar de seguridad a su base de datos

A continuación se mencionan algunos consejos sencillos (habitualmente de sentido común), que permiten dotar de seguridad a su base de datos:

- Limitar los accesos al servidor (fundamentalmente al archivo de contraseñas, al archivo de argumentos y a los archivos de traza o de alerta de cada instancia Oracle).
- Prohibir la autenticación por el sistema operativo a través de la red (el argumento REMOTE\_OS\_AUTHENT debe ser igual a FALSE).
- Bloquee y deje caducada la contraseña de las cuentas por defecto que no se utilicen (por defecto, es el caso cuando crea una base de datos con el asistente gráfico Configuración de base de datos). Las únicas cuentas por defecto que obligatoriamente deben estar abiertas son SYS y SYSTEM.
- Modifique las contraseñas por defecto de las cuentas por defecto que utilice (en primer lugar SYS y SYSTEM). Puede consultar la vista DBA\_USERS\_WITH\_DEFPWD para obtener la lista de usuarios que todavía tienen su contraseña por defecto.
- Utilice contraseñas complejas (10 caracteres al menos, con letras mayúsculas, minúsculas, cifras y caracteres especiales).
- Utilice una política de gestión de contraseñas, con un número limitado de intentos erróneos de conexión permitidos y reglas de complejidad de las contraseñas.
- No almacene las contraseñas sin cifrar en tablas o scripts.
- Asigne la menor cantidad de permisos posible a los usuarios (fundamentalmente permisos de sistema ANY).
- Utilice roles para gestionar los permisos. Si el rol está activado por una aplicación, protéjalo con una contraseña.
- Defina sus propios roles y no utilice los roles predefinidos por Oracle.
- Solo asigne un rol a un usuario si realmente necesita todos los permisos contenidos en el rol. Si no es el caso, cree otro rol más restrictivo.
- No asigne ningún permiso de sistema a PUBLIC (no tiene ninguno por defecto).
- No asigne ningún permiso de objeto a PUBLIC (diferente a los asignados por defecto).
- Revoque los permisos EXECUTE que varios paquetes asignan por defecto a PUBLIC, potencialmente peligrosos para la seguridad: DBMS\_LOB, DBMS\_JOB, UTL\_FILE, UTL\_HTTP, UTL\_TCP, UTL\_SMTP. Este punto es complejo porque estos paquetes se utilizan por numerosas cuentas Oracle (SYS principalmente). Si elimina el permiso EXECUTE asignado a PUBLIC en estos paquetes tendrá que volver a asignar los permisos necesarios directamente a los usuarios afectados (puede consultar la vista DBA\_DEPENDENCIES para conocer las cuentas que utilizan estos paquetes).

# Supervisar los usuarios conectados

La vista V\$SESSION permite identificar los usuarios conectados actualmente:

SQL> SELECT sid,serial#,username,osuser,status FROM v\$session;				
SID	SERIAL#	USERNAME	OSUSER	STATUS
-----				
1	3		SYSTEM	ACTIVE
...				
10	10494	VDEP	vdep	INACTIVE
14	450	SYSTEM	Administrator	ACTIVE

 Las sesiones sin valor en la columna USERNAME corresponden a procesos batch.

Las columnas interesantes de la vista V\$SESSION son las siguientes:

SID	Identificador de la sesión.
SERIAL#	Número de serie de la sesión.
USERNAME	Nombre del usuario (cuenta Oracle).
SCHEMANAME	Nombre del esquema del usuario (puede ser diferente de USERNAME si la sesión ha ejecutado la sentencia SQL ALTER SESSION SET CURRENT_SCHEMA).
STATUS	Estado de la sesión (ACTIVE, INACTIVE o KILLED).
LOGON_TIME	Fecha y hora de conexión.
OSUSER	Nombre del usuario, a nivel del sistema operativo.
MACHINE	Nombre de la máquina del usuario, a nivel del sistema operativo.
TERMINAL	Nombre del terminal del usuario, a nivel del sistema operativo.
PROGRAM	Nombre del programa utilizado por el usuario para conectarse a la base de datos.
SERVER	Tipo de proceso de servidor (DEDICATED o SHARED).
SQL_ID	Identificador de la consulta SQL que se está ejecutando (un join con V\$SQL o V\$SQLAREA sobre la misma columna permite ver la sentencia SQL en cuestión).
SERVICE_NAME	Nombre de servicio de la sesión (servicio al que está conectado el usuario).

Si es necesario, también puede consultar la vista V\$SESSION\_LONGOPS para obtener información relativa a las operaciones largas (en ejecución desde hace más de 6 segundos). Para desconectar a un usuario puede utilizar la sentencia SQL ALTER SYSTEM.

### Sintaxis

```
ALTER SYSTEM KILL SESSION 'sid,serial#';
```

o

```
ALTER SYSTEM DISCONNECT SESSION 'sid,serial#'  
{ IMMEDIATE | POST_TRANSACTION};
```

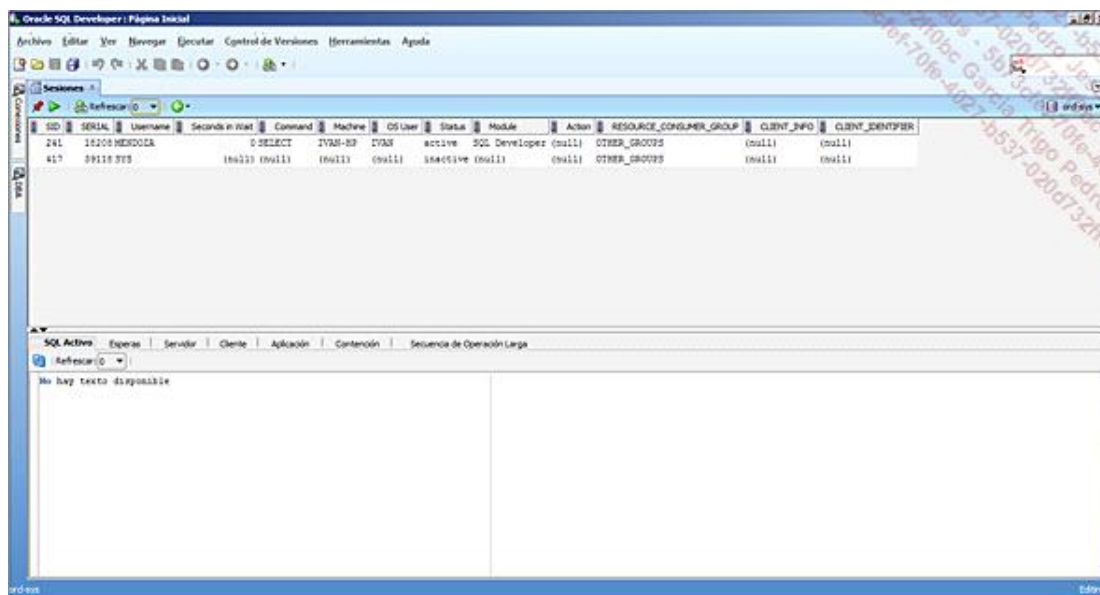
### Ejemplo:

```
ALTER SYSTEM KILL SESSION '10,10494';
```

Las sentencias SQL `ALTER SYSTEM KILL SESSION` y `ALTER SYSTEM DISCONNECT SESSION ... IMMEDIATE` son equivalentes: cierran la sesión inmediatamente, sin esperar al final de una eventual transacción en ejecución (esta última se anula). Por el contrario, la sentencia SQL `ALTER SYSTEM DISCONNECT SESSION ... POST_TRANSACTION`, espera a que la transacción en ejecución termine.

Un usuario que está ejecutando una consulta tiene un estado `ACTIVE` (`INACTIVE`, en caso contrario). Si un usuario se desconecta mientras está activo, su consulta se interrumpe y un mensaje de error le indica que ha sido desconectado (`ORA-00028: su sesión se ha cerrado`) y la sesión desaparece de `V$SESSION`. Si está inactivo, la conexión se cierra, pero la sesión sigue estando visible en `V$SESSION` con el estado `KILLED`, hasta que se notifique al usuario la desconexión durante su próxima acción (con el mismo error `ORA-00028`).

En Oracle SQL Developer, puede seleccionar el elemento **Controlar sesiones...** del menú **Herramientas** con el objetivo de mostrar la lista de sesiones diferentes a las que se corresponden con procesos batch:



En la parte inferior del panel **Sesiones**, varias pestañas permiten obtener información acerca de la actividad de la sesión.

Haciendo clic con el botón derecho en un registro de la lista se abre un menú contextual que fundamentalmente permite cerrar la sesión (menú **Matar sesión**).

# Utilizar EM Express

## 1. Usuarios

En EM Express, seleccione el elemento **Usuarios** del menú **Seguridad** para acceder a la página de gestión de usuarios:

Nombre	Estado de la Cuenta	Fecha de Caducidad	Tablespace por Defecto	Tablespace Temporal	Perfil	Creación
ANONYMOUS	🔒	jue 11 de sep de 2014 11:15:19	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:54:42
APEX_040200	🔒	jue 11 de sep de 2014 10:33:25	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 10:29:19
APEX_PUBLIC_USER	🔒	jue 11 de sep de 2014 10:29:19	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 10:29:19
APPQOSSYS	🔒	jue 11 de sep de 2014 8:54:34	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:54:34
AUDSYS	🔒	jue 11 de sep de 2014 8:40:57	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 8:40:57
BI	🔒	dom 30 de nov de 2014 19:33:46	USERS	TEMP	DEFAULT	dom 30 de nov de 2014 19:33:46
CTXSYS	🔒	dom 30 de nov de 2014 19:33:46	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:40:42
DESKMP	🔒	jue 11 de sep de 2014 8:54:32	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:54:32
DIP	🔒	jue 11 de sep de 2014 8:44:33	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 8:44:33
DVF	🔒	jue 11 de sep de 2014 11:15:19	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 11:15:19
DVSYS	🔒	jue 11 de sep de 2014 11:15:19	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 11:15:19
FLOWSEEDS	🔒	jue 11 de sep de 2014 10:29:19	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 10:29:19
GPMACHTN_INTERNAL	🔒	jue 11 de sep de 2014 8:44:25	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:44:25
GPMACHTN_USER	🔒	jue 11 de sep de 2014 8:58:54	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 8:58:54
GPMACHTN_USER	🔒	jue 11 de sep de 2014 8:44:25	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 8:44:25
HR	🔒	dom 30 de nov de 2014 19:33:46	USERS	TEMP	DEFAULT	dom 30 de nov de 2014 19:33:46
DI	🔒	dom 30 de nov de 2014 19:33:46	USERS	TEMP	DEFAULT	dom 30 de nov de 2014 19:33:46
UACSYS	🔒	jue 11 de sep de 2014 11:15:19	SYSTEM	TEMP	DEFAULT	jue 11 de sep de 2014 11:15:19
MDSYS	🔒	jue 11 de sep de 2014 11:15:19	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 11:15:19
MDSYS	🔒	jue 11 de sep de 2014 8:40:50	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:40:50
OE	🔒	dom 30 de nov de 2014 19:33:46	EXAMPLE	TEMP	DEFAULT	dom 30 de nov de 2014 19:33:46

Desde esta página puede realizar diversas acciones relacionadas con la gestión de usuarios:

Acciones	Estado de la Cuenta	Fecha de Caducidad	Tablespace por Defecto
🔒	🔒	jue 11 de sep de 2014 11:15:19	SYSAUX
🔒	🔒	jue 11 de sep de 2014 10:33:25	SYSAUX
🔒	🔒	jue 11 de sep de 2014 10:29:19	USERS
🔒	🔒	jue 11 de sep de 2014 8:54:34	SYSAUX
🔒	🔒	jue 11 de sep de 2014 8:40:57	USERS
🔒	🔒	dom 30 de nov de 2014 19:33:46	USERS
🔒	🔒	dom 30 de nov de 2014 19:33:46	SYSAUX

- crear un nuevo usuario (menú o botón **Crear por**);
- crear un nuevo usuario idéntico a otro usuario existente (menú o botón **Crear como**);
- eliminar un usuario (menú o botón **Borrar Usuario**);
- mostrar los detalles de un usuario (menú **Ver Detalles** o clic en el nombre del usuario en la lista);
- modificar las propiedades básicas (autenticación, contraseña o perfil) de un usuario (menú **Modificar Cuenta**);
- modificar los tablespaces por defecto de un usuario (menú **Modificar Tablespaces**);
- modificar los permisos de sistema y los roles asignados a un usuario (menú **Modificar Privilegios y Roles**);
- modificar los permisos de objeto asignados a un usuario (menú **Otorgar Privilegios de Objeto**).

Existen diferentes cuadros de diálogo para realizar estas acciones:

- Crear un usuario:

**Creado por**

☒ **Cuenta de Usuario**
☐ Tablespaces
 ☐ Privilegio

Nombre \*

Autenticación \* ☒ Contraseña ☐ Externo ☐ Global

Contraseña \*

Confirmar Contraseña \*

Perfil

Contraseña Caducada ☐

Cuenta Bloqueada ☐

- Modificar la cuenta:

**Modificar Cuenta**

Nombre \*

Autenticación ☒ Contraseña ☐ Externo ☐ Global

Contraseña

Confirmar Contraseña

Perfil

Contraseña Caducada ☐

Cuenta Bloqueada ☐

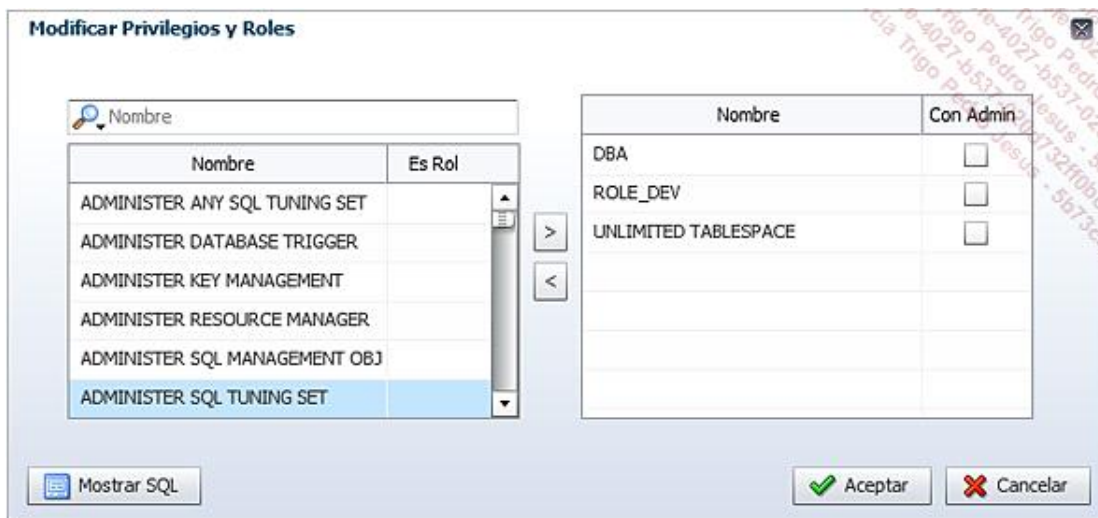
- Modificar los tablespaces:

**Modificar Tablespaces**

Tablespace por Defecto

Tablespace Temporal

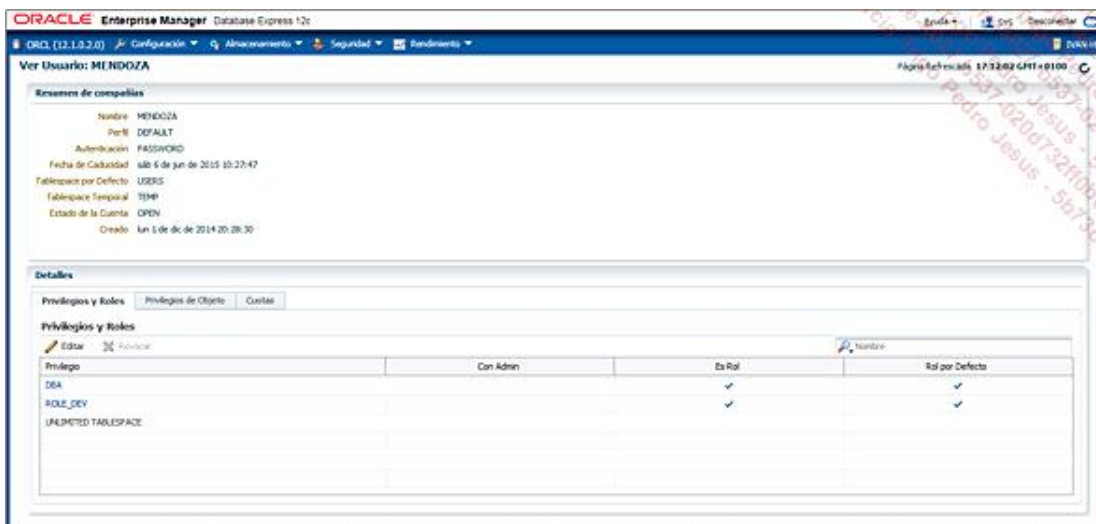
- Modificar los privilegios y los roles:



- Asignar privilegios de objeto:



Haciendo clic en el enlace del nombre del usuario o seleccionando el elemento **Ver detalles**, del menú **Acciones**, se accede a una página que permite consultar las características de un usuario:

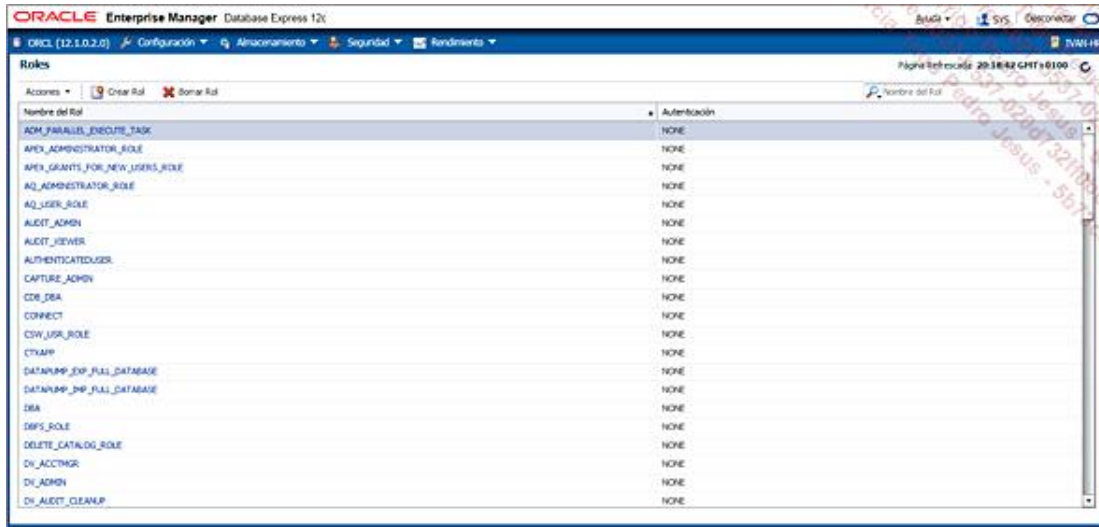


Las pestañas del panel **Detalles** permiten consultar y modificar los permisos del usuario.

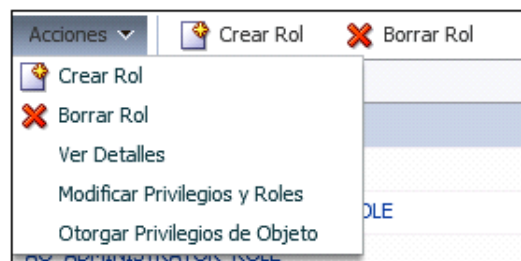


## 2. Roles

En EM Express, seleccione el elemento **Roles** del menú **Seguridad** para acceder a la página de gestión de roles:



Desde esta página puede realizar diversas acciones relacionadas con la gestión de roles:



- crear un nuevo rol (menú o botón **Crear Rol**);
- eliminar un rol (menú o botón **Borrar Rol**);
- mostrar los detalles de un rol (menú **Ver Detalles** o clic en el nombre del rol en la lista);
- modificar los permisos de sistema y los roles asignados a un rol (menú **Modificar Privilegios y Roles**);
- modificar los permisos de objeto asignados a un rol (menú **Otorgar Privilegios de Objeto**).

Hay diferentes cuadros de diálogo para realizar estas acciones:

- Crear un rol:





- Modificar privilegios y roles:

**Modificar Privilegios y Roles**

Nombre

Nombre	Es Rol
ADMINISTER ANY SQL TUNING SET	
ADMINISTER DATABASE TRIGGER	
ADMINISTER KEY MANAGEMENT	
ADMINISTER RESOURCE MANAGER	
ADMINISTER SQL MANAGEMENT OB.	
ADMINISTER SQL TUNING SET	

> <

Nombre	Con Admin
CREATE SESSION	<input type="checkbox"/>
CREATE TABLE	<input type="checkbox"/>
CREATE VIEW	<input type="checkbox"/>
CREATE PROCEDURE	<input type="checkbox"/>
CREATE SEQUENCE	<input type="checkbox"/>
CREATE TRIGGER	<input type="checkbox"/>

Mostrar SQL

Aceptar Cancelar

- Asignar privilegios de objeto:

**Otorgar Privilegios de Objeto**

Seleccionar Esquema y Tipo de Objeto    Seleccionar Objetos    Otorgar Privilegios de Objeto

Esquema \* APPQOSSYS

Tipo de Objeto \* TABLE

Nombre del Objeto

Mostrar SQL

Aceptar Cancelar

Haciendo clic en el enlace del nombre del rol o seleccionando el elemento **Ver Detalles** del menú **Acciones**, se accede a una página que permite consultar las características de un rol:

**ORACLE Enterprise Manager Database Express 12c**

Ver Rol: ROLE\_DEV

Resumen de propiedades

Nombre del Rol: ROLE\_DEV

Autenticación: NONE

**Detalles**

Privilegios y Roles

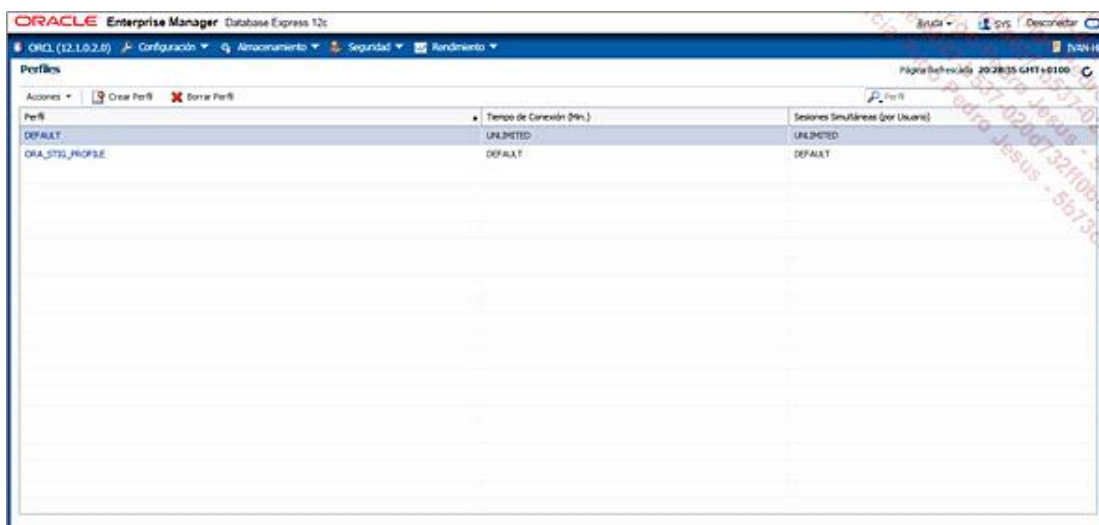
Privilegios y Roles

Privilegio	Con Admin	Es Rol	Rol por Defecto
CREATE PROCEDURE			
CREATE SEQUENCE			
CREATE SESSION			
CREATE TABLE			
CREATE TRIGGER			
CREATE VIEW			

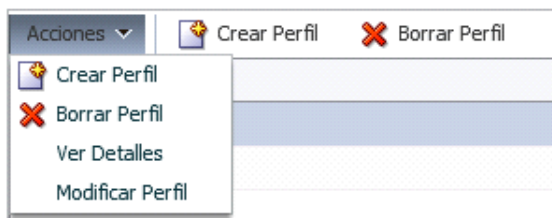
Las pestañas del panel **Detalles** permiten consultar y modificar los permisos del rol.

### 3. Perfiles

En EM Express, seleccione el elemento **Perfiles** del menú **Seguridad** para acceder a la página de gestión de perfiles:



Desde esta página puede realizar diversas acciones relacionadas con la gestión de perfiles:



- crear un nuevo perfil (menú o botón **Crear Perfil**);
- eliminar un perfil (menú o botón **Borrar Perfil**);
- mostrar los detalles de un perfil (menú **Ver Detalles** o clic en el nombre del perfil en la lista);
- modificar un perfil (menú **Modificar Perfil**).

Hay diferentes cuadros de diálogo para realizar estas acciones:

- Crear un perfil:



- Modificar el perfil:



**Modificar Perfil**

General Contraseña

CPU/Sesión (Seg./100) \* Ilimitado ▼

CPU/Llamada (Seg./100) \* Ilimitado ▼

Tiempo de Conexión (Min.) \* Ilimitado ▼

Tiempo de Inactividad (Min.) \* Ilimitado ▼

Sesiones Simultáneas (por Usuario) \* Ilimitado ▼

Lecturas/Sesión (Bloques) \* Ilimitado ▼

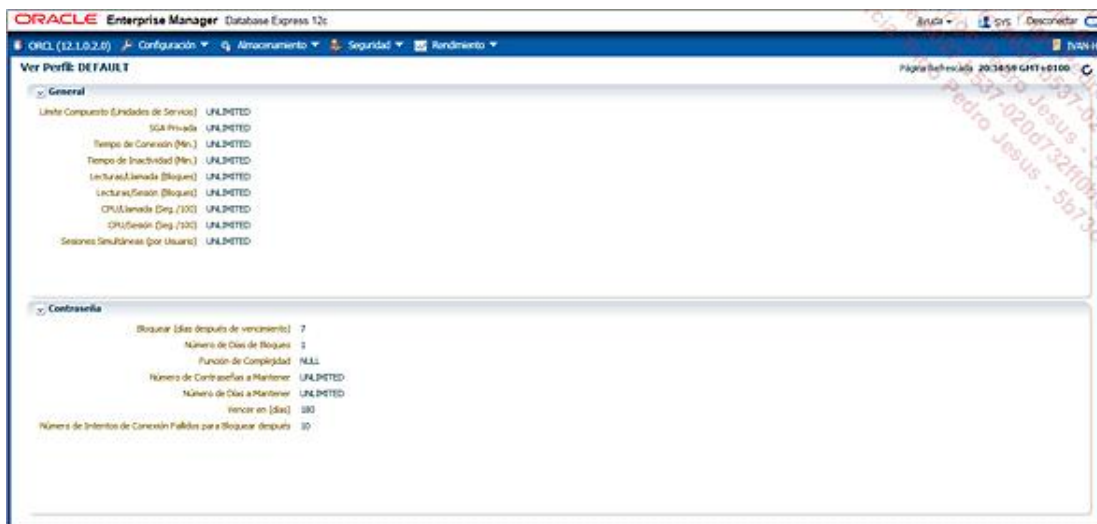
Lecturas/Llamada (Bloques) \* Ilimitado ▼

SGA Privada \* Ilimitado ▼ ⓘ

Límite Compuesto (Unidades de Servicio) \* Ilimitado ▼

Mostrar SQL Aceptar Cancelar

Haciendo clic en el enlace del nombre del perfil o seleccionando el elemento **Ver Detalles** del menú **Acciones**, se accede a una página que permite consultar las características de un perfil:



ORACLE Enterprise Manager Database Express 12c

Ver Perfil: DEFAULT

General

Límite Compuesto (Unidades de Servicio) UNLIMITED

SGA Privada UNLIMITED

Tiempo de Conexión (Min.) UNLIMITED

Tiempo de Inactividad (Min.) UNLIMITED

Lecturas/Llamada (Bloques) UNLIMITED

Lecturas/Sesión (Bloques) UNLIMITED

CPU/Llamada (Seg./100) UNLIMITED

CPU/Sesión (Seg./100) UNLIMITED

Sesiones Simultáneas (por Usuario) UNLIMITED

Contraseña

Bloquear (días después de vencimiento) 7

Número de Días de Bloqueo 3

Función de Complejidad NULL

Número de Contraseñas a Mantener UNLIMITED

Número de Días a Mantener UNLIMITED

Intentar en (días) 30

Número de Intentos de Conexión Fallidos para Bloquear después 30

# Principios

Para la gestión de la seguridad, Oracle permite:

- definir los usuarios que se pueden conectar a la base de datos (con una identificación por el sistema operativo o por la base de datos);
- definir en qué tablespace(s) puede crear los objetos un usuario (eventualmente ninguno);
- limitar el uso de los recursos de sistema;
- imponer una política de gestión de contraseñas (expiración periódica, no reutilizar antes de un periodo de tiempo, etc.);
- definir los permisos de cada usuario dentro de la base de datos.

En una base de datos Oracle los permisos de los usuarios se administran con la noción de **permiso**.

Un permiso es el derecho para:

- ejecutar una sentencia SQL en general (por ejemplo, crear una tabla): es la noción de **permiso de sistema**;
- acceder a un objeto de otro usuario (por ejemplo, actualizar los datos de la tabla CLIENT): es la noción de **permiso de objeto**.

Los permisos se pueden asignar directamente a los usuarios o por medio de roles. Un **rol** es una agrupación de permisos (sistemas y objetos) con un nombre, que se puede asignar como tal a un usuario; de esta manera, este usuario recibe automáticamente los permisos contenidos en el rol. Los roles facilitan la gestión de los permisos.

Además, Oracle ofrece una funcionalidad de auditoría que permite trazar la actividad de los usuarios en la base de datos. Para saber más, puede consultar la documentación Oracle® Database Security Guide.

# Crear y modificar usuarios

## 1. Modo de identificación del usuario

Oracle puede identificar un usuario, o bien dejar que sea el sistema operativo quien lo haga. Los dos modos de identificación se pueden utilizar al mismo tiempo, en la misma base de datos.

### a. Identificación por Oracle

El usuario se conecta a la base de datos introduciendo un nombre y una contraseña. Oracle comprueba el nombre y la contraseña del usuario.

```
SQL> CONNECT oheu/rx239$  
Conectado.
```

Es posible utilizar las funcionalidades de gestión de contraseñas propuestas por Oracle.

### b. Identificación por el sistema operativo

El usuario se conecta a la base de datos sin introducir nombre ni contraseña.

```
SQL> CONNECT /  
Conectado.
```

Oracle no comprueba la contraseña, sino que simplemente controla que el nombre del usuario, a nivel del sistema operativo, corresponda con un nombre de usuario de la base de datos. La identificación inicial se ha realizado por el sistema operativo.

Las funcionalidades de gestión de contraseñas propuestas por Oracle no se pueden usar (no es Oracle quien gestiona la contraseña).

Para establecer el vínculo entre el nombre del usuario en el sistema operativo y el nombre del usuario en la base de datos, Oracle utiliza un prefijo definido por el argumento `OS_AUTHENT_PREFIX` (por defecto igual a `OPS$`).

Por ejemplo, el usuario con nombre `vdep` a nivel del sistema operativo se podrá conectar a la base por un `CONNECT /` solo si existe una cuenta Oracle `ops$vdep`.

En plataformas Windows el nombre de dominio o, de manera predeterminada, el nombre de la máquina, forma parte del nombre del usuario: `SRVWINORA\VDEP`, por ejemplo. Es el nombre completo el que se debe prefijar para formar el nombre de la cuenta Oracle (todo en mayúsculas): `OPS$SRVWINORA\VDEP`, por ejemplo.

El prefijo puede ser igual a una cadena vacía (`OS_AUTHENT_PREFIX = ""`); en este caso, el nombre del usuario a nivel del sistema operativo y el nombre del usuario en Oracle son idénticos.

Además, el argumento `REMOTE_OS_AUTHENT` se puede poner a `TRUE` para indicar si los usuarios remotos se identifican con este método (`FALSE`, para prohibirlo; es el valor por defecto). Poner el argumento `REMOTE_OS_AUTHENT` a `TRUE` puede ser peligroso si la red no es segura. Este argumento se dejó de usar desde la versión 11.

## 2. Creación de un usuario

La sentencia SQL CREATE USER permite crear un nuevo usuario.

### Sintaxis simplificada

```
CREATE USER nombre IDENTIFIED { BY la_contraseña | EXTERNALLY }  
[ DEFAULT TABLESPACE nombre_tablespace ]  
[ TEMPORARY TABLESPACE nombre_tablespace ]  
[ QUOTA { valor [K|M] | UNLIMITED } ON nombre_tablespace [,...] ]  
[ PROFILE nombre_perfil ]  
[ PASSWORD EXPIRE ]  
[ ACCOUNT { LOCK | UNLOCK } ];
```

### Ejemplo:

- Usuario identificado por el OS, solo con las cláusulas obligatorias:

```
CREATE USER "OPS$SRVWINORA\VDEP" IDENTIFIED EXTERNALLY;
```

- Usuario identificado por Oracle, con cláusulas adicionales:

```
CREATE USER oheu IDENTIFIED BY tempo  
DEFAULT TABLESPACE data  
QUOTA UNLIMITED ON data  
PASSWORD EXPIRE;
```

Observe la sintaxis particular para especificar el nombre del usuario OPS\$SRVWINORA\ VDEP: las comillas son necesarias, porque el nombre contiene caracteres no permitidos (barra invertida). En adelante, siempre será necesario utilizar la misma sintaxis para gestionar este usuario.

Para que un nuevo usuario pueda conectarse de manera efectiva, además hay que darle permisos para hacerlo asignándole el permiso de sistema CREATE SESSION (consulte la sección Gestionar los permisos).

Por lo tanto, es posible tener cuentas para los usuarios sin que estos últimos tengan permiso para conectarse. Esta funcionalidad era interesante en la versión 7, porque permitía preparar las cuentas de usuario sin activarlas inmediatamente, o prohibir temporalmente a un usuario conectarse sin eliminar su cuenta.

Esta funcionalidad siempre se puede utilizar, pero es más sencillo bloquear/desbloquear explícitamente la cuenta (ACCOUNT LOCK | UNLOCK).

Las opciones de la sentencia SQL CREATE USER son:

### **nombre**

Nombre del usuario. El nombre del usuario debe respetar las reglas de nomenclatura de Oracle presentadas en la sección La base de datos del capítulo Las bases de la arquitectura Oracle. Si no es el caso, hay que poner el nombre entre comillas.

### **IDENTIFIED**

Esta cláusula indica si el usuario se identifica por el sistema operativo (EXTERNALLY) o por Oracle (BY la\_contraseña).

En caso de una identificación por Oracle, se especifica la contraseña inicial del usuario.

Para la contraseña, se deben respetar las reglas de nomenclatura de Oracle, salvo si se ha implementado la funcionalidad de control de la complejidad de las contraseñas (novedad de la versión 8 - se verá más adelante).

Desde la versión 11, por defecto las contraseñas son sensibles a mayúsculas/minúsculas (argumento `SEC_CASE_SENSITIVE_LOGON` igual a `TRUE`, por defecto). Si desea tener contraseñas no sensibles a mayúsculas/minúsculas basta con asignar el valor `FALSE` al argumento `SEC_CASE_SENSITIVE_LOGON` (pero no es aconsejable para la seguridad). Desde la versión 12, este argumento ya no se utiliza y Oracle recomienda dejarlo a `TRUE`.

## **DEFAULT TABLESPACE**

Esta cláusula indica en qué tablespace se crean, por defecto, los segmentos del usuario (es decir, si no hay ninguna cláusula `TABLESPACE` durante la creación del segmento).

Si la cláusula se omite, se asigna al usuario el tablespace permanente predeterminado de la base de datos (consulte la sección `Tablespace permanente` del capítulo `Tablespaces y archivos de datos`).

La noción de tablespace por defecto, no impide al usuario crear objetos en otro tablespace (si tiene una cuota en el tablespace en cuestión); permite simplemente especificar un tablespace por defecto si el usuario omite la cláusula `TABLESPACE` durante la creación de un segmento. Esta cláusula presenta interés, principalmente, para aquellos usuarios que pueden crear segmentos: desarrolladores, ingenieros de calidad y, de manera menos habitual, usuarios finales.

## **TEMPORARY TABLESPACE**

Esta cláusula indica en qué tablespace se crean los segmentos temporales del usuario (durante una operación de ordenación, por ejemplo). Puede indicar el nombre de un tablespace temporal o el nombre de un grupo de tablespaces temporales.

Si la cláusula se omite, el tablespace temporal predeterminado de la base de datos se asigna al usuario (consulte la sección `Tablespace temporal` del capítulo `Tablespaces y archivos de datos`).

## **QUOTA**

Esta cláusula indica en qué tablespace(s) puede crear objetos el usuario, y hasta qué límite.

La noción de `QUOTA` permite limitar el espacio que un usuario puede emplear en un tablespace con los segmentos que crea.

Esta funcionalidad solo afecta a los usuarios que pueden crear segmentos, y no a los usuarios finales de una aplicación, que se conforman con emplear los objetos ya existentes y que pertenecen generalmente a una cuenta distinta (de alguna manera, el "propietario" de la aplicación).

Por defecto, los usuarios no tienen ninguna cuota en ningún tablespace, salvo los `DBA`, que tienen una cuota ilimitada en todos los tablespaces.

Si un usuario tiene permiso para crear segmentos, hay que darle explícitamente una cuota en, al menos, un tablespace. El hecho de que se haya utilizado una cláusula `DEFAULT TABLESPACE` no asigna ninguna cuota en el tablespace en cuestión; son dos mecanismos diferentes.

En la práctica, solo hay que dar cuotas a los usuarios que lo necesitan (los desarrolladores, la cuenta "propietario")

de la aplicación) y solo en los tablespaces estrictamente necesarios y suficientes. En realidad, es mejor evitar dar cuotas a estos usuarios en el tablespace SYSTEM o el tablespace SYSAUX.

La noción de cuota queda sin objeto para el tablespace temporal y el tablespace de anulación.

Si un usuario quiere crear un segmento en un tablespace en el que no tiene cuota o supera la cuota asignado a este usuario, se obtiene un error:

- Ninguna cuota en el tablespace:

ORA-01950: no hay permisos en el tablespace 'DATA'

- Superar la cuota en el tablespace:

ORA-01536: se ha sobrepasado la cuota de espacio asignada al tablespace 'DATA'

## **PROFILE**

Esta cláusula indica el perfil asignado al usuario. La noción de perfil se presenta en la sección Utilizar perfiles.

## **PASSWORD EXPIRE**

Esta cláusula permite forzar una modificación de la contraseña durante la primera conexión (la contraseña del usuario ha expirado). Esta cláusula no tiene efecto si el usuario se identifica por el sistema operativo.

Si la cuenta se crea con la opción `PASSWORD EXPIRE`, se le pedirá al usuario durante su primera conexión que cambie la contraseña que se le ha asignado inicialmente.

## **ACCOUNT**

`LOCK`: la cuenta está bloqueada y la conexión prohibida (error ORA-28000: cuenta bloqueada).

`UNLOCK`: la cuenta no está bloqueada y la conexión está permitida (valor por defecto).

Si la cuenta se crea con la opción `LOCK`, la cuenta existe pero el usuario no se puede conectar. La sentencia SQL `ALTER USER` se podrá utilizar más tarde para desbloquear la cuenta del usuario (consulte la sección Crear y modificar usuarios).

## **3. Modificación de un usuario**

La sentencia SQL `ALTER USER` permite modificar un usuario.

### **Sintaxis simplificada**

```
ALTER USER nombre
[ IDENTIFIED { BY la_contraseña | EXTERNALLY } ]
[ DEFAULT TABLESPACE nombre_tablespace ]
[ TEMPORARY TABLESPACE nombre_tablespace ]
[ QUOTA { valor [K|M] | UNLIMITED } ON nombre_tablespace [,...] ]
[ PROFILE nombre_perfil ]
[ PASSWORD EXPIRE ]
```



```
[ ACCOUNT { LOCK | UNLOCK } ];
```

Las cláusulas son las mismas que para la creación.

#### Ejemplos:

- Modificación de la contraseña de un usuario:

```
ALTER USER oheu  
  IDENTIFIED BY tempo  
  PASSWORD EXPIRE;
```

- Modificación del tablespace por defecto y asignación de cuotas:

```
ALTER USER oheu  
  DEFAULT TABLESPACE test  
  QUOTA UNLIMITED ON test  
  QUOTA 10M ON data;
```

- Bloquear una cuenta:

```
ALTER USER oheu ACCOUNT LOCK;
```

- Desbloquear una cuenta:

```
ALTER USER oheu ACCOUNT UNLOCK;
```

El primer ejemplo permite modificar la contraseña de un usuario, forzándole a cambiarla durante su primera conexión; esta técnica se puede emplear si el usuario ha perdido su contraseña (el DBA no tiene ningún medio de conocer la contraseña de los usuarios).

El segundo ejemplo permite modificar el tablespace por defecto del usuario y asignarle cuotas en dos tablespaces.

El tercer ejemplo se puede utilizar para prohibir temporalmente la conexión a un usuario. Si actualmente está conectado, no se le desconecta.

El cuarto ejemplo se puede utilizar para autorizar de nuevo a un usuario a conectarse.

Disminuir una cuota o ponerla a 0 no elimina los objetos ya creados por el usuario.



Modificar la contraseña de SYS modifica la contraseña de SYSDBA, almacenada en el archivo de contraseñas (si se utiliza un archivo de contraseñas).

## 4. Eliminación de un usuario

La sentencia SQL DROP USER permite eliminar un usuario.

#### Sintaxis


```
DROP USER nombre [ CASCADE ];
```

### Ejemplo:

```
DROP USER "OPS$SRVWINORA\VDEP" CASCADE;
```

Si el usuario tiene objetos, la opción CASCADE debe estar presente para forzar la eliminación inicial de los objetos. Si el usuario tiene objetos y la opción CASCADE no está, se obtiene el error ORA-01922:

```
ORA-01922: CASCADE necesaria para eliminar 'OPS$SRVWINORA\VDEP'
```

 Es una sentencia DDL no hay ROLLBACK posible.

Un usuario actualmente conectado no se puede eliminar:

```
ORA-01940: imposible eliminar un usuario que está conectado
```

Antes de eliminar un usuario es posible exportar los objetos que le pertenecen; estos objetos se podrán volver a importar en otro esquema.

## 5. Encontrar información de los usuarios

Varias vistas del diccionario de datos permiten obtener información de los usuarios:

- DBA\_USERS: información de los usuarios;
- DBA\_TS\_QUOTAS: información de las cuotas de los usuarios.

Las columnas interesantes de las diferentes vistas se presentan a continuación.

### **DBA\_USERS**

USERNAME	Nombre del usuario.
USER_ID	Identificador del usuario.
PASSWORD	Contraseña (cifrada) del usuario.
ACCOUNT_STATUS	Estado de la cuenta (OPEN, LOCKED, UNLOCKED, EXPIRED, etc.).
LOCK_DATE	Fecha de bloqueo (si la cuenta está bloqueada).
EXPIRY_DATE	Fecha de expiración de la contraseña.
DEFAULT_TABLESPACE	Tablespace por defecto del usuario.
TEMPORARY_TABLESPACE	Tablespace temporal del usuario.
CREATED	Fecha de creación del usuario.
PROFILE	Perfil.
AUTHENTICATION_TYPE	Mecanismo utilizado para la autenticación.
LAST_LOGIN	Fecha y hora de la última conexión del usuario.
ORACLE_MAINTAINED	Y si la cuenta es una cuenta interna predefinida por Oracle (consulte la observación más adelante). N en caso contrario.

## **DBA\_TS\_QUOTAS**

TABLESPACE_NAME	Nombre del tablespace.
USERNAME	Nombre del usuario que tiene una cuota en el tablespace.
BYTES	Espacio, en bytes, utilizado actualmente por el usuario.
MAX_BYTES	Cuota, en bytes, del usuario en el tablespace (1).
BLOCKS	Espacio, en bloques, utilizado actualmente por el usuario.
MAX_BLOCKS	Cuota, en bloques, del usuario en el tablespace (1).

(1) -1 si la cuota es cuota UNLIMITED



Una base de datos Oracle contiene varias cuentas internas predefinidas utilizadas por algunas funcionalidades o algunos componentes; inicialmente están bloqueadas y tienen una contraseña expirada (ACCOUNT\_STATUS vale EXPIRED& LOCKED). Estas cuentas se enumeran y describen en la documentación Oracle Database 2 Day + Security Guide.

# Utilizar perfiles

## 1. Presentación

Un perfil es un conjunto de limitaciones de recursos, con un nombre que se puede asignar a un usuario.

Se pueden limitar los siguientes recursos:

- tiempo de CPU por llamada y/o por sesión;
- número de lecturas lógicas por llamada y/o por sesión;
- número de sesiones abiertas al mismo tiempo por un usuario;
- tiempo de inactividad por sesión;
- duración total de la sesión;
- cantidad de memoria privada en la SGA (solo en configuración de servidores compartidos).

Una lectura lógica corresponde a una lectura de un bloque durante una consulta, tanto si el bloque ya está presente en memoria (en la *Database Buffer Cache*) como si se lee de disco (en este caso, la lectura lógica también corresponde a una lectura física).

Desde la versión 8, los perfiles también se pueden utilizar para implementar una política de gestión de contraseñas.

Se pueden poner en marcha las siguientes funcionalidades:

- bloqueo de cuenta (y duración de bloqueo) cuando se supera un número determinado de intentos fallidos de conexión;
- tiempo de vida de las contraseñas (eventualmente con un periodo de cortesía);
- no reutilizar una contraseña antes de un plazo determinado de tiempo o antes de determinado número de cambios;
- complejidad de la contraseña.

El perfil llamado `DEFAULT` se crea automáticamente durante la creación de la base de datos. Este perfil se asigna por defecto a los usuarios. Por defecto, este perfil `DEFAULT` no impone ningún límite para los recursos; por el contrario, desde la versión 11, este perfil tiene límites para las contraseñas (como se verá más adelante).

La limitación de los recursos, con ayuda de los perfiles, no ofrece muchas posibilidades. Si desea controlar de manera más precisa la asignación de recursos (CPU, espacio de anulación, duración de inactividad) a los usuarios o grupos de usuarios, puede utilizar la *Database Resource Manager*. La implementación de esta funcionalidad se realiza gracias al paquete `DBMS_RESOURCE_MANAGER`. Para saber más, consulte la documentación Oracle® Database Administrator's Guide.

## 2. Creación de un perfil

La sentencia SQL `CREATE PROFILE` permite crear un nuevo perfil.

### Sintaxis

```
CREATE PROFILE nombre LIMIT
[ SESSIONS_PER_USER { valor | UNLIMITED | DEFAULT } ]
[ CPU_PER_SESSION { valor | UNLIMITED | DEFAULT } ]
```

```
[ CPU_PER_CALL { valor | UNLIMITED | DEFAULT } ]
[ CONNECT_TIME { valor | UNLIMITED | DEFAULT } ]
[ IDLE_TIME { valor | UNLIMITED | DEFAULT } ]
[ LOGICAL_READS_PER_SESSION { valor | UNLIMITED | DEFAULT } ]
[ LOGICAL_READS_PER_CALL { valor | UNLIMITED | DEFAULT } ]
[ COMPOSITE_LIMIT { valor | UNLIMITED | DEFAULT } ]
[ PRIVATE_SGA { valor [K|M] | UNLIMITED | DEFAULT } ]
[ FAILED_LOGIN_ATTEMPTS { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_LIFE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_REUSE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_REUSE_MAX { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_LOCK_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_GRACE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_VERIFY_FUNCTION { nombre_función | NULL | DEFAULT } ];
```

### Ejemplo:

```
CREATE PROFILE exploitation LIMIT
  SESSIONS_PER_USER 3
  IDLE_TIME 30
  FAILED_LOGIN_ATTEMPTS 3
  PASSWORD_LIFE_TIME 30
  PASSWORD_REUSE_TIME 180
  PASSWORD_LOCK_TIME UNLIMITED
  PASSWORD_GRACE_TIME 3
  PASSWORD_VERIFY_FUNCTION verif_mdp_exploitation;
```

Las limitaciones de recursos son las siguientes:

SESSIONS_PER_USER	Número de sesiones simultáneas.
CPU_PER_SESSION	CPU total por sesión (1/100 s).
CPU_PER_CALL	CPU total por llamada (1/100 s).
CONNECT_TIME	Duración total de conexión (minutos).
IDLE_TIME	Duración de la inactividad (minutos).
LOGICAL_READS_PER_SESSION	Número de lecturas lógicas por sesión.
LOGICAL_READS_PER_CALL	Número de lecturas lógicas por llamada.
PRIVATE_SGA	Cantidad de memoria privada en la SGA.
COMPOSITE_LIMIT	Suma ponderada de CPU_PER_SESSION, CONNECT_TIME, LOGICAL_READS_PER_SESSION y PRIVATE_SGA.

Para el límite COMPOSITE\_LIMIT, la vista RESOURCE\_COST permite consultar las ponderaciones utilizadas y la sentencia SQL ALTER RESOURCE COST modificarlas.

Las limitaciones relativas a las contraseñas son las siguientes:

FAILED_LOGIN_ATTEMPTS	Número de intentos erróneos de conexión permitidos antes del bloqueo de la cuenta, 10 en el perfil DEFAULT.
PASSWORD_LOCK_TIME	Duración del bloqueo (en días), 1 en el perfil DEFAULT.

PASSWORD_LIFE_TIME	Duración de la contraseña (en días), 180 en el perfil DEFAULT.
PASSWORD_GRACE_TIME	Periodo de cortesía después de la expiración de la contraseña (en días), 7 en el perfil DEFAULT.
PASSWORD_REUSE_TIME	Número de días durante el que no se puede reutilizar una contraseña.
PASSWORD_REUSE_MAX	Número de cambios de contraseña antes de que se pueda reutilizar una contraseña.
PASSWORD_VERIFY_FUNCTION	Función de comprobación de la complejidad de la contraseña.

Se pueden utilizar palabras clave para especificar el valor de un límite:

- UNLIMITED: ninguna limitación.
- DEFAULT: el argumento hereda el valor del perfil DEFAULT.

No se especifica ningún límite en un perfil que toma el valor DEFAULT.

Los límites PASSWORD\_REUSE\_TIME y PASSWORD\_REUSE\_MAX se utilizan de manera conjunta para controlar la reutilización de una contraseña antigua:

- Si especifica un valor para estos dos límites, entonces el usuario no puede reutilizar una contraseña mientras que la contraseña no se haya cambiado el número de veces indicado por el límite PASSWORD\_REUSE\_MAX, durante el número de días definido por el límite PASSWORD\_REUSE\_TIME. Por ejemplo, si PASSWORD\_REUSE\_TIME vale 60 y PASSWORD\_REUSE\_MAX vale 5, entonces el usuario puede reutilizar una contraseña al cabo de 60 días, a condición de que haya cambiado la contraseña, al menos, 5 veces.
- Si especifica un valor para alguno de los dos límites y UNLIMITED para el otro, entonces el usuario nunca puede reutilizar una contraseña.
- Si especifica UNLIMITED para los dos límites, entonces se ignoran y el usuario puede reutilizar una contraseña sin ninguna restricción.

No olvide que si no se define un límite o si se define con el valor DEFAULT, entonces se utiliza el valor especificado en el perfil DEFAULT.

Para los diferente límites especificados en días, es posible utilizar números decimales, representando fracciones de día (por ejemplo 1/24 = una hora).

El límite PASSWORD\_VERIFY\_FUNCTION permite especificar una función PL/SQL que se utilizará para comprobar si la contraseña introducida por el usuario respeta algunas reglas. Un valor NULL permite no utilizar la función de comprobación. Esta función debe aceptar tres argumentos de entrada (el nombre del usuario, su nueva contraseña y su contraseña antigua) y devuelve un valor booleano.

El script utlpwdmg.sql (repositorio %ORACLE\_HOME%\rdbms\admin o bien \$ORACLE\_HOME/rdbms/admin) contiene un ejemplo de función de comprobación, que se asigna al perfil DEFAULT si se ejecuta el script.

### 3. Modificación de un perfil

La sentencia SQL ALTER PROFILE permite modificar un perfil.

#### Sintaxis

```
ALTER PROFILE nombre LIMIT
[ SESSIONS_PER_USER { valor | UNLIMITED | DEFAULT } ]
[ CPU_PER_SESSION { valor | UNLIMITED | DEFAULT } ]
[ CPU_PER_CALL { valor | UNLIMITED | DEFAULT } ]
[ CONNECT_TIME { valor | UNLIMITED | DEFAULT } ]
[ IDLE_TIME { valor | UNLIMITED | DEFAULT } ]
[ LOGICAL_READS_PER_SESSION { valor | UNLIMITED | DEFAULT } ]
[ LOGICAL_READS_PER_CALL { valor | UNLIMITED | DEFAULT } ]
[ COMPOSITE_LIMIT { valor | UNLIMITED | DEFAULT } ]
[ PRIVATE_SGA { valor [K|M] | UNLIMITED | DEFAULT } ]
[ FAILED_LOGIN_ATTEMPTS { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_LIFE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_REUSE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_REUSE_MAX { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_LOCK_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_GRACE_TIME { valor | UNLIMITED | DEFAULT } ]
[ PASSWORD_VERIFY_FUNCTION { nombre_función | NULL | DEFAULT } ];
```

#### Ejemplo:

- Modificación del perfil DEFAULT:

```
ALTER PROFILE default LIMIT
SESSIONS_PER_USER 3
IDLE_TIME 30
FAILED_LOGIN_ATTEMPTS 5;
-- los demás argumentos conservan el valor por defecto (UNLIMITED)
```

- Modificación de otro perfil:

```
ALTER PROFILE exploitation LIMIT
SESSIONS_PER_USER 5 -- pasa de 3 a 5
IDLE_TIME UNLIMITED -- eliminación del límite
FAILED_LOGIN_ATTEMPTS DEFAULT; -- toma el valor por defecto (5)
-- permanece inalterado
```

Las opciones son las mismas que para la sentencia SQL CREATE PROFILE.

La modificación de un perfil se asigna a los usuarios durante su próxima conexión; no se tiene en cuenta inmediatamente para los usuarios ya conectados.

Modificar el perfil DEFAULT también asigna los perfiles que tienen límites especificados a DEFAULT.

## 4. Asignación de un perfil a un usuario

Un perfil se puede asignar a un usuario:

- durante la creación del usuario (CREATE USER);
- durante una modificación del usuario (ALTER USER).

#### Ejemplos:

Durante la creación del usuario:

```
CREATE USER xgeo IDENTIFIED BY tempo
    TEMPORARY TABLESPACE temp
    PROFILE exploitation
    PASSWORD EXPIRE;
```

Durante una modificación del usuario:

- Asignación de un perfil:

```
ALTER USER oheu PROFILE exploitation;
```

- Reasignación del perfil DEFAULT:

```
ALTER USER oheu PROFILE DEFAULT;
```

La asignación de un nuevo perfil a los usuarios tiene efecto durante su próxima conexión.

Por defecto, un usuario se crea con el perfil DEFAULT.

## 5. Activación de la limitación de los recursos

Por defecto, el control de la limitación de los recursos no está activada. Crear perfiles y asignarlos a los usuarios no tiene ningún efecto.

Para activar el control de la limitación de los recursos hay que pasar el argumento RESOURCE\_LIMIT a TRUE (FALSE, por defecto):

```
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE [ cláusula_SCOPE ];
```

No olvide utilizar la cláusula SCOPE = BOTH para hacer que la modificación sea persistente en caso de reinicio de la base de datos.

Las funcionalidades de gestión de las contraseñas funcionan incluso si el argumento RESOURCE\_LIMIT es FALSE.

## 6. Eliminación de un perfil

La sentencia SQL DROP PROFILE permite eliminar un perfil.

### Sintaxis

```
DROP PROFILE nombre [ CASCADE ];
```

### Ejemplo:

```
DROP PROFILE exploitation CASCADE;
```



Si el perfil está asignado a los usuarios, la opción CASCADE debe estar presente. Si el perfil está asignado a los usuarios y la opción CASCADE no está, se obtiene el error ORA-02382:

```
ORA-02382: El perfil EXPLOITATION tiene usuarios, imposible realizar la eliminación sin CASCADE
```

El perfil DEFAULT se asigna reubicando a los usuarios afectados. La eliminación de un perfil asigna a los usuarios durante su próxima conexión. El perfil DEFAULT no se puede eliminar.

## 7. Encontrar la información de los perfiles

Varias vistas del diccionario de datos permiten obtener información de los perfiles:

- DBA\_USERS: información de los usuarios, incluyendo el perfil asignado (columna PROFILE);
- DBA\_PROFILES: información de los perfiles.

Las columnas interesantes de la vista DBA\_PROFILES se presentan a continuación.

### **DBA\_PROFILES**

PROFILE	Nombre del perfil.
RESOURCE_NAME	Nombre del recurso controlado.
RESOURCE_TYPE	Tipo del recurso controlado (KERNEL o PASSWORD).
LIMIT	Límite del recurso.

# Gestionar los permisos

## 1. Permiso de sistema

### a. Definición

Un permiso de sistema consiste en poder ejecutar una sentencia SQL en general, por ejemplo, crear una tabla.

Generalmente, cada sentencia SQL tiene, al menos, un permiso de sistema asociado, que tiene el mismo nombre que la sentencia SQL. Por ejemplo, la sentencia SQL `CREATE TABLE` tiene un permiso de sistema asociado `CREATE TABLE` (da permiso para crear una tabla en su propio esquema).

Algunos permisos de sistema toman el nombre de la sentencia SQL con la palabra clave `ANY`. En este caso, el permiso de sistema permite ejecutar la sentencia en cualquier esquema de la base de datos. Por ejemplo, el permiso de sistema `CREATE ANY TABLE` da el permiso para crear una tabla en cualquier esquema de la base de datos.

Cuando la sentencia SQL implicada no es relativa a los objetos de un esquema, no hay permiso `ANY` (`ANY` quiere decir en cualquier esquema): por ejemplo, el permiso para crear un `tablespace` es `CREATE TABLESPACE`; no hay `CREATE ANY TABLESPACE` (un `tablespace` no pertenece a un esquema).

Los permisos de sistema son fuente de poder y origen de peligro, sobre todo en lo que respecta a la gestión de usuarios y permisos (`CREATE USER`, `ALTER USER`, `DROP USER`, `GRANT ANY PRIVILEGE`, `GRANT ANY ROLE`) y la eliminación de objetos (`DROP ANY TABLE`, `DROP TABLESPACE`, etc.); por tanto, los permisos de sistema se deben asignar con cuidado (especialmente los permisos `ANY`).



Piense que si da el permiso `ALTER USER` a un usuario, éste podrá modificar las cuentas de usuario (cambiar las contraseñas, por ejemplo), incluida la de usted.

Algunos permisos de sistema particulares:

<code>CREATE SESSION</code>	Da permiso al usuario para conectarse.
<code>SELECT ANY DICTIONARY</code>	Da permiso al usuario para consultar cualquier objeto del diccionario de datos en el esquema <code>SYS</code> .

Si un usuario no tiene el permiso `CREATE SESSION`, se obtiene el error `ORA-01045` durante un intento de conexión:

```
ORA-01045: el usuario OHEU no tiene el permiso CREATE SESSION; conexión rechazada
```

El permiso `SELECT ANY DICTIONARY` es interesante porque permite dar a un usuario permiso para leer las vistas `DBA`, sin tener que ser `DBA`. En la versión 8 u 8i, el rol `SELECT_CATALOG_ROLE` se puede utilizar para conseguir el mismo objetivo; este rol sigue existiendo por razones de compatibilidad hacia atrás. En la versión 7 era necesario dar a los usuarios el permiso `SELECT ANY TABLE`, lo que podía presentar problemas de seguridad, porque el usuario podía leer cualquier esquema.

Los permisos de sistema se usan principalmente para controlar el uso de las sentencias DDL y, por tanto, están destinados a los administradores, desarrolladores, a la cuenta de propietario de una aplicación y, con menor frecuencia, al usuario final de una aplicación.

Si un usuario no tiene el permiso necesario para realizar una acción, se obtiene el error `ORA-01031`:

ORA-01031: permisos insuficientes

La vista `SYSTEM_PRIVILEGE_MAP` devuelve la lista de todos los permisos de sistema.

## **b. Asignación de un permiso de sistema a un usuario**

La sentencia SQL `GRANT` permite asignar un permiso de sistema.

### Sintaxis

```
GRANT nombre_permiso [,...]
TO { nombre_usuario | PUBLIC } [,...]
[ WITH ADMIN OPTION ];
```

### Ejemplo:

```
GRANT CREATE SESSION, CREATE TABLE TO oheu;
```

El permiso se puede asignar a un usuario o a todos los usuarios (`PUBLIC`).

La cláusula `WITH ADMIN OPTION` asigna al receptor el permiso de transmitir el permiso de sistema.

El permiso asignado se activa inmediatamente.

Para asignar un permiso de sistema, hay que haber recibido:

- el permiso en cuestión, con la cláusula `WITH ADMIN OPTION`;
- o el permiso de sistema `GRANT ANY PRIVILEGE`.

Se pueden asignar varios permisos a varios usuarios, en un único orden. Todos los permisos de sistema se pueden asignar de una vez con la palabra clave `ALL PRIVILEGES` (`GRANT ALL PRIVILEGES TO ...`). Esta posibilidad se debe manipular con mucha precaución.

## **c. Revocación de un permiso de sistema a un usuario**

La sentencia SQL `REVOKE` permite revocar un permiso de sistema.

### Sintaxis

```
REVOKE nombre_permiso [,...]
FROM { nombre_usuario | PUBLIC } [,...];
```

### Ejemplo:

```
REVOKE CREATE TABLE FROM oheu;
```

El permiso se revoca inmediatamente y nunca más se puede ejercer.

Para revocar un permiso de sistema, hay que haber recibido:

- el permiso en cuestión, con la cláusula `WITH ADMIN OPTION`;
- o el permiso de sistema `GRANT ANY PRIVILEGE`.

No se propaga en cascada la revocación de un permiso de sistema que se ha transmitido mediante la cláusula `WITH ADMIN OPTION`. Si un permiso se ha asignado a Ángel con la opción `WITH ADMIN OPTION` y Ángel lo ha transmitido a Iván, revocar el permiso de Ángel no tiene efecto alguno en el permiso transmitido por Ángel a Iván. La cláusula `WITH ADMIN OPTION`, por lo tanto, es doblemente peligrosa.

La sentencia `REVOKE` permite revocar solo los permisos que un usuario ha recibido de manera directa (no los permisos que tiene implícitamente a través `PUBLIC`). Es igual para `PUBLIC`: no puede revocar a `PUBLIC` un permiso no asignado a `PUBLIC`, pensando en eliminarlo de esta manera a todo el mundo.

Si un permiso se ha asignado a un usuario y a `PUBLIC`, la revocación del usuario no tiene efecto sobre la posibilidad del usuario para continuar ejerciendo el permiso: siempre lo tiene a través de `PUBLIC`.

Todos los permisos de sistema se pueden revocar al mismo tiempo, con la palabra clave `ALL PRIVILEGES` (`REVOKE ALL PRIVILEGES FROM ...`).

Si ha asignado un permiso con la opción `WITH ADMIN OPTION` y desea eliminar esta posibilidad de transmisión, hay que revocar el permiso y asignarlo de nuevo sin la opción.

#### **d. Los permisos de sistema `SYSDBA` y `SYSOPER`**

Ya hemos visto que los permisos `SYSDBA` y `SYSOPER` eran necesarios para realizar algunas operaciones de administración (inicio/parada/creación de la base de datos).

Estos permisos se pueden controlar por su pertenencia a un grupo particular del sistema operativo, o bien por un archivo de contraseñas.

En caso de usar un archivo de contraseñas, por defecto solo `SYS` ha recibido los permisos `SYSDBA` y `SYSOPER`.

Si el archivo de contraseñas se asocia exclusivamente a la base de datos (argumento `REMOTE_LOGIN_PASSWORDFILE=EXCLUSIVE`), es posible asignar alguno de estos permisos a otros usuarios.

La asignación y revocación se realizan con las sentencias SQL `GRANT` y `REVOKE`.

En la práctica, es muy raro asignar permisos `SYSOPER` y sobre todo `SYSDBA` (que da un control total sobre la base de datos) a otros usuarios; la cuenta `SYSDBA/SYSOPER` utilizada habitualmente es la cuenta `SYS` (está destinada a esto).

Para asignar el permiso `SYSDBA`, hay que estar conectado `AS SYSDBA`; para asignar el permiso `SYSOPER`, hay que estar conectado `AS SYSDBA` o `AS SYSOPER`.

La vista `V$PWFILERS` permite enumerar los usuarios que han recibido los permisos `SYSDBA` o `SYSOPER`; esta vista siempre se vacía si `REMOTE_LOGIN_PASSWORDFILE=NONE`.



Existen otros permisos administrativos de este tipo, además de `SYSBACKUP` que apareció en la versión 12, que permiten realizar operaciones de copia de seguridad y restauración con la ayuda del Recovery Manager (RMAN) o de SQL\*Plus (consulte el capítulo Copia de seguridad y restauración).

## 2. Permiso de objeto

### a. Definición

Un permiso de objeto es el permiso para acceder a un objeto de otro usuario: por ejemplo, actualizar los datos de la tabla `CLIENT`.

Por defecto, solo el propietario de un objeto tiene el permiso de acceso. Para que otro usuario pueda acceder al objeto, el propietario del objeto le debe dar un permiso de objeto.

Los principales permisos de objeto son los siguientes:

Permiso	Tabla	Vista	Secuencia	Programa
SELECT	x	x	x	
INSERT	x	x		
UPDATE	x	x		
DELETE	x	x		
EXECUTE				x

En la tabla anterior, la columna "Programa" hace referencia a los procedimientos y funciones almacenados, y a los paquetes.

Éstos dan los siguientes permisos:

SELECT	Permiso de lectura de los datos (ejecución de la sentencia SQL <code>SELECT</code> ).
INSERT	Permiso de creación de los datos (ejecución de la sentencia SQL <code>INSERT</code> ).
UPDATE	Permiso de actualización de los datos (ejecución de la sentencia SQL <code>UPDATE</code> ).
DELETE	Permiso de eliminación de los datos (ejecución de la sentencia SQL <code>DELETE</code> ).
EXECUTE	Permiso de ejecución del programa (invocar al procedimiento, función o paquete desde otro programa).

Tener permiso sobre un objeto no elimina la obligatoriedad de cualificar al objeto con el nombre del propietario para acceder a él (en otro caso, Oracle piensa que usted intenta acceder a un objeto en su esquema). Para facilitar la escritura de consultas y hacer que el esquema sea propietario de los objetos transparentes, hay que utilizar sinónimos en la ocurrencia, en lugar de sinónimos públicos.

De manera recíproca, la existencia de un sinónimo, incluso público, no da ningún permiso sobre el objeto subyacente.

Los permisos de objeto están destinados a controlar el acceso a los objetos identificados. Por ejemplo: el permiso para crear un comando (es decir, el permiso para hacer un `INSERT` en la tabla `PEDIDO`) o el permiso para eliminar una ficha cliente (es decir, el permiso para hacer un `DELETE` en la tabla `CLIENTE`).

Principalmente se usan para permitir a los usuarios finales de una aplicación acceder, directamente o a través de una interfaz de usuario, a los objetos de la aplicación creados en una cuenta "propietaria" de la aplicación (ya que, por defecto, solo el propietario de un objeto tiene permiso de acceso).

El mensaje de error que devuelve Oracle cuando un usuario no tiene el permiso necesario para realizar una acción concreta sobre algún objeto se determina de manera diferente si el usuario tiene o no, al menos, un permiso sobre el objeto:

- Si el usuario no tiene ningún permiso sobre el objeto, Oracle devuelve el error ORA-00942: Tabla o vista no existe.
- Si el usuario tiene, al menos, un permiso sobre el objeto, Oracle devuelve el error ORA-01031: permisos insuficientes.

En el primer caso, Oracle considera que el usuario normalmente no tiene ningún medio para saber que el objeto al que accede existe: por tanto, indica que el objeto no existe. En el segundo caso, Oracle considera que el usuario puede saber que el objeto existe: por tanto, indica que el permiso es insuficiente.

## **b. Asignación de un permiso de objeto a un usuario**

La sentencia SQL GRANT permite asignar un permiso de objeto.

### Sintaxis

```
GRANT { nombre_permiso[(lista_columnas)] [,...] | ALL [PRIVILEGES] }
ON [nombre_esquema.]nombre_objeto
TO { nombre_usuario | PUBLIC } [,...]
[ WITH GRANT OPTION ];
```

### Ejemplo:

```
GRANT SELECT, INSERT, UPDATE(nombre,apellidos) ON miembro TO oheu;
```

Para los permisos INSERT y UPDATE es posible especificar una lista de columnas con el objetivo de limitar el permiso a las columnas indicadas.

La cláusula WITH GRANT OPTION da al receptor la posibilidad de transmitir el permiso de objeto.

La palabra clave ALL permite asignar todos los permisos. En caso de que el usuario que asigna todos los permisos no sea el propietario del objeto, sino un usuario que ha recibido algunos permisos sobre el objeto con la posibilidad de transmitirlos (cláusula WITH GRANT OPTION), la palabra clave ALL hace referencia solo a los permisos que el usuario ha recibido.

El permiso se puede asignar a un usuario o a todos los usuarios (PUBLIC).

Es posible asignar varios permisos a varios usuarios en un único orden; por el contrario, la asignación de los permisos de objeto se realiza objeto por objeto.

El permiso asignado se activa inmediatamente.

Para asignar un permiso de objeto hay que:

- ser propietario del objeto;
- haber recibido el permiso en cuestión con la cláusula WITH GRANT OPTION;
- o haber recibido el permiso de sistema ANY OBJECT PRIVILEGE.

Un usuario que transmite un permiso que ha recibido con la opción WITH GRANT OPTION debe cualificar el nombre del objeto con el nombre del propietario (ya que el objeto no le pertenece), salvo si existe algún sinónimo sobre el objeto.

Algunos permisos de sistema dan permisos de objeto implícitamente a todos los objetos. Por ejemplo: `SELECT ANY TABLE`.

El DBA tiene los permisos de sistema `ANY` indicados anteriormente; es la razón por la que puede acceder a cualquier objeto sin permiso de objeto. Desde la versión 9, también tiene permiso de sistema `ANY OBJECT PRIVILEGE`, que le permite asignar un permiso de objeto a cualquier objeto sin haber recibido el permiso en cuestión `WITH GRANT OPTION`. Antes de la versión 9, esto no era posible.

### c. Revocación de un permiso de objeto a un usuario

La sentencia SQL `REVOKE` permite revocar un permiso de objeto.

#### Sintaxis

```
REVOKE { nombre_permiso [,...] | ALL [PRIVILEGES] }  
ON [nombre_esquema.]nombre_objeto  
FROM { nombre_usuario | PUBLIC } [,...];
```

#### Ejemplo:

```
REVOKE INSERT, UPDATE ON cliente FROM oheu;
```

La sentencia `REVOKE` permite revocar a un usuario solamente aquellos permisos que ha recibido de manera directa (no los permisos que tiene implícitamente a través `PUBLIC`). Es igual para `PUBLIC`: no puede revocar a `PUBLIC` un permiso no asignado a `PUBLIC`, pensando que de esta manera se lo quita a todo el mundo.

Si un permiso se ha asignado a un usuario y a `PUBLIC`, el revocar el permiso al usuario no tiene ningún efecto; el usuario continúa ejerciendo el permiso: siempre lo tiene a través de `PUBLIC`. Todos los permisos de objeto se pueden revocar de una vez mediante la palabra clave `ALL` (`REVOKE ALL ON ... FROM ...`). Si ha asignado un permiso con la opción `WITH GRANT OPTION` y desea eliminar esta posibilidad de transmisión, hay que revocar el permiso y asignarlo de nuevo sin la opción. El permiso se elimina inmediatamente y no se puede ejercer más.

Para eliminar un permiso de objeto hay que:

- ser propietario del objeto;
- haber recibido el permiso en cuestión con la cláusula `WITH GRANT OPTION`;
- o haber recibido el permiso de sistema `ANY OBJECT PRIVILEGE`.

Existe propagación en cascada en la revocación de un permiso de objeto que ha sido transmitido gracias a la cláusula `WITH GRANT OPTION`. Si un permiso se ha asignado a Ángel con la opción `WITH GRANT OPTION` y Ángel lo ha transmitido a Iván, revocar el permiso de Ángel revoca también el de Iván. El funcionamiento no es el mismo que para los permisos de sistema.

### d. Permisos de las vistas y programas almacenados

Un usuario que tiene un permiso sobre una vista, no necesita tener permisos sobre los objetos manipulados por la vista.

Por defecto, es igual para los programas almacenados: el programa almacenado se ejecuta con los permisos del propietario (*definer rights*). Si es necesario, el programa almacenado se puede diseñar para ejecutarse con los

permisos de quien lo invoca (*invoker rights*).

El comportamiento deseado se define durante la creación del programa almacenado, gracias a la cláusula AUTHID.

#### Sintaxis

```
AUTHID { CURRENT_USER | DEFINER }
```

El modo de funcionamiento por defecto (permisos del propietario) es muy interesante porque permite utilizar las vistas y los programas almacenados como capa intermedia para el acceso a los objetos de la base de datos. Es posible construir una aplicación en la que la parte de cliente no acceda nunca directamente a las tablas de la base de datos, sino que pase por las vistas y/o los programas almacenados. Este tipo de enfoque permite, principalmente:

- ocultar la estructura real de las tablas y hacerla evolucionar con el mínimo impacto en la parte cliente;
- implementar reglas de gestión (controles, cálculos, seguridad, etc.) en el lado servidor.

### **e. Llamar a un objeto de otro esquema**

Incluso si un usuario tiene permisos sobre un objeto de un otro esquema, debe utilizar el nombre de su propietario como prefijo en el nombre del objeto, para poder acceder:

```
SELECT * FROM diane.miembro;
```

Se pueden definir sinónimos públicos para simplificar la escritura de las consultas y hacerlas independientes del nombre del propietario:

```
CREATE PUBLIC SYNONYM miembro FOR diane.miembro;
```

Cuando la consulta `SELECT * FROM miembro` se ejecuta, Oracle mira en primer lugar en el esquema actual si existe un objeto llamado `miembro`. Si es el caso, lo usa. Si no es el caso, busca un sinónimo público con este nombre. Si lo encuentra, reemplaza el sinónimo por su definición.

Desde la versión 8i, también es posible "ubicar" su sesión en el esquema de otro usuario. Cuando se hace referencia a un objeto en una sentencia SQL, Oracle mira si existe un objeto como este en el otro esquema antes de buscar un posible sinónimo público.

#### Ejemplo:

```
ALTER SESSION SET CURRENT_SCHEMA = diane;
```



Observe que las dos técnicas no dan el permiso en sí; son técnicas de resolución de nombres. Una vez que el nombre está resuelto, Oracle mira si el usuario tiene los permisos necesarios para acceder al objeto.

### **f. Ir más allá en la gestión de los permisos**

Oracle ofrece las funcionalidades *Virtual Private Database* (VPD) y *Fine-Grained Access* (FGA), que permiten utilizar mecanismos de filtrado de registros en las tablas. Estas funcionalidades se describen en la documentación Oracle® Database Security Guide; se basan en el uso del paquete DBMS\_RLS.



### 3. Rol

#### a. Definición

Un rol es una agrupación de permisos (sistema y objeto) con un nombre, que se puede asignar a un usuario. Todos los permisos agrupados en el rol se asignan al mismo tiempo al usuario. Los roles permiten simplificar la gestión de los permisos.

Las principales características de los roles son las siguientes:

- Un rol se puede asignar a otro rol.
- Un usuario puede tener varios roles.
- Un rol no pertenece a nadie.

La implementación se realiza en tres etapas:

- 1 - creación del rol;
- 2 - asignación de los permisos (sistema y objeto) al rol;
- 3 - asignación del rol a los usuarios (o bien a otros roles).

Veremos también que, desde la versión 12, un rol se puede asignar a un programa almacenado y, de esta manera, los permisos asignados al rol en cuestión solo se podrán utilizar durante la ejecución del programa implicado. Esta funcionalidad se llama "seguridad basada en código" (*Code-Based Security*) o "control de acceso basado en código" (*Code-Based Access Control*).

#### b. Creación de un rol

La sentencia SQL `CREATE ROLE` permite crear un rol.

##### Sintaxis

```
CREATE ROLE nombre  
[ IDENTIFIED { BY la_contraseña | EXTERNALLY | USING nombre_paquete }  
| NOT IDENTIFIED ];
```

##### Ejemplo:

```
CREATE ROLE mailing;
```

Las opciones son:

<code>IDENTIFIED BY la_contraseña</code>	Indica que se necesita una contraseña para activar el rol.
<code>IDENTIFIED EXTERNALLY</code>	Indica que se necesita una identificación externa para activar el rol.
<code>IDENTIFIED USING nombre_paquete</code>	Indica que solo el paquete puede activar el rol.
<code>NOT IDENTIFIED</code>	Indica que no es necesaria ninguna identificación para activar el rol. Es el valor por defecto.

Para crear un rol hay que tener el permiso de sistema `CREATE ROLE`.

Durante la creación de un rol es posible concretar el mecanismo por el que se podrá activar el rol: una contraseña, una identificación externa (sistema operativo) o un paquete. El mecanismo de activación se presentará en este capítulo, en la sección Rol - Activación o desactivación de un rol.

La sentencia SQL `ALTER ROLE` permite modificar un rol y el modo de identificación para poder activarlo.

#### Sintaxis

```
ALTER ROLE nombre
[ IDENTIFIED { BY la_contraseña | EXTERNALLY | USING nombre_paquete }
| NOT IDENTIFIED ];
```

### **c. Asignación de un permiso a un rol**

La sentencia SQL `GRANT` permite asignar permisos de sistema o de objeto a un rol.

#### Sintaxis para los permisos de sistema

```
GRANT nombre_permiso [... ]
TO nombre_rol [... ]
[ WITH ADMIN OPTION ];
```

#### Sintaxis para los permisos de objeto

```
GRANT { nombre_permiso[(lista_columnas)] [... ] | ALL [PRIVILEGES] }
ON [nombre_esquema.]nombre_objeto
TO nombre_rol [... ];
```

#### Ejemplo:

```
GRANT CREATE SESSION, CREATE TABLE TO mailing;
GRANT SELECT, INSERT ON miembro TO mailing;
```

La sintaxis es la misma que para la asignación directa a un usuario, con excepción de la cláusula `WITH GRANT OPTION`, que no está permitida para la asignación de un permiso de objeto a un rol.

Los permisos asignados se activan inmediatamente para los usuarios conectados que tienen el rol activo. La noción de rol activo se presenta un poco más adelante en este capítulo, Gestionar los permisos, sección Rol - Activación o desactivación de un rol.

Cualquier usuario puede asignar un permiso a un rol, siempre que tenga la capacidad de asignar el permiso; éste es el motivo por el que el rol no pertenece a nadie. Para asignar un permiso de sistema a un rol hay que haber recibido el permiso correspondiente mediante la cláusula `WITH ADMIN OPTION` o tener el permiso de sistema `GRANT ANY PRIVILEGE`. Para asignar un permiso de objeto a un rol hay que ser propietario del objeto, haber recibido el permiso correspondiente mediante la cláusula `WITH GRANT OPTION` o tener el permiso de sistema `ANY OBJECT PRIVILEGE`.

### **d. Revocación de un permiso a un rol**

La sentencia SQL REVOKE permite revocar permisos de sistema o de objeto a un rol.

#### Sintaxis para los permisos de sistema

```
REVOKE nombre_permiso [ ,...]  
FROM nombre_rol [ ,...];
```

#### Sintaxis para los permisos de objeto

```
REVOKE { nombre_permiso [ ,... ] | ALL [PRIVILEGES] }  
ON [nombre_esquema.]nombre_objeto  
FROM nombre_rol [ ,...];
```

#### Ejemplo:

```
REVOKE CREATE TABLE FROM mailing;  
REVOKE UPDATE ON miembro FROM mailing;
```

La sintaxis es la misma que para la revocación directa para un usuario. De manera general, la sentencia SQL REVOKE solo permite eliminar a un "destino" (usuario, rol o PUBLIC) que haya sido asignado explícitamente a este "destino". Por ejemplo, si se ha asignado un permiso a un rol, no es posible eliminarlo directamente para un usuario que ha recibido el rol.

Preste atención a la mezcla de asignación directa y asignación a un rol: si un permiso se asigna a un rol y el rol al usuario y además, en paralelo, el permiso se asigna directamente al usuario, revocar el permiso al usuario le permitirá seguir ejerciéndolo (a través del rol).

Los permisos se revocan inmediatamente y ya no se pueden ejercer más por parte de los usuarios conectados que tienen el rol activo. La noción de rol activo se presenta un poco más adelante en este capítulo, sección Rol - Activación o desactivación de un rol.

Cualquier usuario puede revocar un permiso de un rol, siempre que pueda revocar el permiso (el rol no pertenece a nadie). Para revocar un permiso de sistema de un rol hay que haber recibido el permiso correspondiente mediante la cláusula WITH ADMIN OPTION o bien tener el permiso de sistema GRANT ANY PRIVILEGE. Para revocar un permiso de objeto de un rol hay que ser el propietario del objeto, haber recibido el permiso en cuestión mediante la cláusula WITH GRANT OPTION o tener el permiso de sistema ANY OBJECT PRIVILEGE.

### **e. Asignación de un rol a un usuario o a otro rol**

La sentencia SQL GRANT permite asignar un rol a un usuario o a otro rol.

#### Sintaxis

```
GRANT nombre_rol [ ,...]  
TO { nombre_usuario | PUBLIC | nombre_rol } [ ,...]  
[ WITH ADMIN OPTION ];
```

#### Ejemplo:

```
GRANT mailing TO oheu;
```

La sintaxis es la misma que para la asignación de un permiso de sistema.

Para poder asignar un rol hay que haber recibido el rol en cuestión mediante la cláusula `WITH ADMIN OPTION` o tener el permiso de sistema `GRANT ANY ROLE`. El creador de un rol no es propietario del rol, sino que se le asigna mediante la opción `WITH ADMIN OPTION`; por tanto, puede asignar el rol que ha creado.

El rol asignado no se activa inmediatamente si el usuario ya está conectado.

La cláusula `WITH ADMIN OPTION` también da el permiso para modificar (sentencia `SQL ALTER ROLE`) y para eliminar el rol (sentencia `SQL DROP ROLE`).

Un usuario puede tener varios roles; en este caso, los permisos se acumulan (no hay permiso "negativo").

## **f. Revocación de un rol a un usuario o a otro rol**

La sentencia `SQL REVOKE` permite revocar un rol.

### Sintaxis

```
REVOKE nombre_rol [,...]
FROM { nombre_usuario | PUBLIC | nombre_rol } [,...];
```

### Ejemplo:

```
REVOKE mailing FROM oheu;
```

La sintaxis es la misma que para la revocación de un permiso de sistema. Para poder eliminar un rol hay que haber recibido el rol en cuestión mediante la cláusula `WITH ADMIN OPTION` o tener el permiso de sistema `GRANT ANY ROLE`. El creador del rol que ha recibido este último con la cláusula `WITH ADMIN OPTION` puede eliminar el rol.

Cuando un rol se revoca, los usuarios conectados con el rol activo pueden continuar ejerciendo los permisos asociados hasta finalizar la sesión o hasta desactivar el rol.

## **g. Eliminación de un rol**

La sentencia `SQL DROP ROLE` permite eliminar un rol.

### Sintaxis

```
DROP ROLE nombre_rol;
```

### Ejemplo:

```
DROP ROLE mailing;
```

Para eliminar un rol hay que haber recibido el rol en cuestión con la cláusula `WITH ADMIN OPTION` o tener el permiso de sistema `DROP ANY ROLE`.

El rol se elimina inmediatamente a los usuarios; los permisos asociados ya no se pueden ejercer más.

## **h. Activación o desactivación de un rol**

Un rol asignado a un usuario (directamente o a través un otro rol), se activa automáticamente por defecto durante la conexión del usuario.

Si el usuario se conecta durante la asignación, la activación inmediata no es automática. El usuario puede activar el rol gracias a la sentencia SQL SET ROLE.

Adicionalmente, entre los roles asignados al usuario, es posible definir los que se activan automáticamente durante la conexión del usuario. Estos son los roles por defecto, definidos mediante una sentencia SQL ALTER USER. A continuación, el usuario puede activar los demás gracias a la sentencia SQL SET ROLE.

Utilizar varios roles sin que estén todos activos al mismo tiempo es interesante porque:

- existe un argumento, MAX\_ENABLED\_ROLES, que limita el número de roles activos al mismo tiempo para un usuario. Si un usuario es susceptible de emplear un número de roles superior a este límite, es posible desactivar algunos para activar otros. El argumento MAX\_ENABLED\_ROLES ya no se usa y solo se conserva por razones de compatibilidad hacia atrás. Vale 150 por defecto, y este valor está codificado en duro en Oracle; por lo tanto, modificar el valor de este argumento no tiene ningún efecto. Cuando este argumento se elimine definitivamente, la limitación en el número de roles activos al mismo tiempo dejará de existir.
- se puede asignar roles, protegidos por contraseñas, a los usuarios, pero por defecto inactivos y sin dar la contraseña al usuario; se trata de aplicaciones que activan los roles que necesitan, proporcionando la contraseña. De esta manera, fuera de la aplicación en cuestión, el usuario no tiene ningún medio para activar y utilizar el rol (y eventualmente, cometer errores).

La sentencia SQL ALTER USER permite definir los roles por defecto de un usuario.

#### Sintaxis

```
ALTER USER nombre_usuario DEFAULT ROLE
{ nombre_rol [,...] | ALL [ EXCEPT nombre_rol [,...] ] | NONE };
```

#### Ejemplo:

```
ALTER USER oheu DEFAULT ROLE mailing;
ALTER USER vdep DEFAULT ROLE ALL EXCEPT mailing;
```

Las opciones son:

ALL	Todos los roles asignados al usuario están activados por defecto. La cláusula EXCEPT permite eliminar algunos.
NONE	Ninguno de los roles asignados al usuario está activo por defecto.

Esta sentencia SQL anula y sustituye la situación actual de los roles por defecto: no añade o elimina los roles a la lista actual. Los roles asignados a un usuario que ya tiene roles por defecto no se definen por defecto.

La sentencia SQL SET ROLE permite activar o desactivar un rol.

#### Sintaxis

```
SET ROLE
{ nombre_rol [ IDENTIFIED BY la_contraseña ] [,...]
| ALL [ EXCEPT nombre_rol [,...] ] | NONE };
```

#### Ejemplo:

---

```
-- El usuario VDEP activa el rol MAILING
SET ROLE mailing;
```

Las opciones son:

IDENTIFIED BY	Indica la contraseña que permite activar el rol.
ALL	Todos los roles asignados al usuario están activados. La cláusula EXCEPT permite eliminar algunos.
NONE	Ninguno de los roles asignados al usuario está activado (por tanto, todos los roles desactivados).

Los roles se han debido asignar inicialmente a un usuario; por tanto, no es posible auto asignarse un rol activándolo (afortunadamente).

Esta sentencia SQL anula y sustituye los roles actualmente activos (no añade).

La opción ALL no se puede utilizar para roles protegidos con contraseña.

Los roles definidos con la opción IDENTIFIED USING nombre\_paquete solo se pueden activar a partir del paquete especificado.

El procedimiento SET\_ROLE del paquete DBMS\_SESSION permite hacer lo mismo (consulte la documentación "PL/SQL Packages and Types Reference").

## i. Limitación de los roles

Para desarrollar un objeto (una vista o un programa almacenado) que utiliza objetos de otro esquema hay que haber recibido los permisos de los objetos de manera directa y no a través de un rol.

Por otra parte, durante la ejecución de un programa almacenado, los roles activos del usuario que lo invoca solo se tienen en cuenta si el programa almacenado está diseñado para ejecutarse con los permisos del usuario que lo invoca (cláusula AUTHID CURRENT\_USER).

## j. Roles predefinidos

De manera estándar, Oracle ofrece un gran número de roles predefinidos, entre los que encontramos:

CONNECT	Autoriza la conexión (contiene solo el permiso de sistema CREATE SESSION).
RESOURCE	Permite crear los principales objetos de un esquema (tablas, triggers, secuencias, programas almacenados, pero no las vistas).
DBA	Da todos los permisos de sistema, con la opción WITH ADMIN OPTION.
EM_EXPRESS_BASIC	Permite conectarse a EM Express y ver las páginas en modo solo lectura.
EM_EXPRESS_ALL	Permite conectarse a EM Express y utilizar todas las funcionalidades (consulta y modificación).

Las vistas DBA\_SYS\_PRIVS y DBA\_TAB\_PRIVS permiten conocer los permisos agrupados en estos roles predefinidos. Oracle aconseja no utilizar los roles predefinidos CONNECT, RESOURCE y DBA, sino crear sus propios roles. Desde la versión 10.2 (10 g Release 2), el rol CONNECT solo contiene el permiso de sistema CREATE SESSION. Antes de esta versión, este rol contenía otros permisos que permitían crear los principales

objetos de un esquema (tabla, vista, etc.) o modificar la sesión (permiso de sistema ALTER SESSION). Desde la versión 10.2, si necesita asignar estos permisos a un usuario, el rol CONNECT no es suficiente; por el contrario, puede asignar estos permisos directamente al usuario o a través de un rol específico que se cree.

De la misma manera, a lo largo de las versiones, el rol RESOURCE cada vez es más restrictivo. Desde la versión 12 este rol ya no contiene el permiso de sistema UNLIMITED TABLESPACE.

## k. Seguridad basada en código

Desde la versión 12 se puede asignar un rol a un programa almacenado, de manera que los permisos asignados al rol en cuestión solo se puedan utilizar durante la ejecución del programa implicado. Esta funcionalidad se llama "seguridad basada en código" (*Code-Based Security*) o "control de acceso basado en código" (*Code Based Access Control*).

Veremos que esta funcionalidad puede ser interesante, pero de manera diferente, tanto para los programas almacenados que se ejecutan con los permisos del propietario (*definer rights*) como para los que se ejecutan con los permisos del usuario que lo invoca (*invoker rights*).

### Sintaxis para asignar un rol a un programa almacenado

```
GRANT nombre_rol [,...]
TO {PROCEDURE | FUNCTION | PACKAGE} nombre_programa_almacenado [,...];
```

### Sintaxis para retirar un rol de un programa almacenado

```
REVOKE ALL | nombre_rol [,...]
FROM {PROCEDURE | FUNCTION | PACKAGE} nombre_programa_almacenado [,...];
```

Un rol se puede asignar a un programa almacenado por el usuario SYS o por el propietario del programa; si no es el caso, se obtiene el error ORA-28702: La unidad de programa xxx no pertenece al usuario que concede el permiso. Por otra parte, de manera inicial, se debe haber asignado el rol directamente al propietario del programa almacenado; si no es el caso, se obtiene el error ORA-01924: el rol 'xxx' no está permitido o no existe.

En caso de un programa almacenado que se ejecuta con los permisos del propietario (*definer rights*), que es el caso por defecto, esta funcionalidad es interesante solo para el código que se ejecute en SQL dinámico. De hecho, para que un programa almacenado se pueda compilar sin error, todos los permisos necesarios para el código SQL estático se deben asignar directamente al propietario y no por medio de un rol (consulte la sección Limitación de los roles de este capítulo). En caso de SQL dinámico, los permisos se comprueban durante la ejecución y Oracle tiene en cuenta los permisos asignados directamente al usuario y los permisos concedidos a los roles asignados al programa almacenado. Esto permite salvar la limitación de los roles que se ha mencionado anteriormente (los permisos asignados al rol ahora se tienen en cuenta durante la ejecución del programa almacenado), pero solo para algunos programas, no para todos, lo que puede permitir controlar mejor los problemas de seguridad relacionados con la inyección de SQL, por ejemplo.

En el caso de un programa almacenado que se ejecuta con los permisos del usuario que lo invoca (*invoker rights*), esta funcionalidad es interesante porque permite limitar los permisos asignados al usuario que ejecuta el programa almacenado. El rol se asigna solo al propietario del programa almacenado y al programa almacenado. Solo está activo durante la ejecución del programa almacenado, sin que sea necesario asignarlo al usuario que ejecuta el programa almacenado: por tanto, este último no puede ejercer directamente los permisos asignados al rol.

## 4. Encontrar información de los permisos

### a. Permisos de sistema

Varias vistas del diccionario de datos permiten obtener información de los permisos de sistema:

- `DBA_SYS_PRIVS`: permisos de sistema asignados a los usuarios (o a los roles);
- `SESSION_PRIVS`: permisos de sistema activos actualmente en la sesión (obtenidos directamente o a través de un rol);
- `SYSTEM_PRIVILEGE_MAP`: lista de todos los permisos de sistema.

Las columnas interesantes de las diferentes vistas se presentan a continuación.

#### **DBA\_SYS\_PRIVS**

<code>GRANTEE</code>	Nombre del usuario o del rol que ha recibido el permiso de sistema.
<code>PRIVILEGE</code>	Permiso de sistema recibido.
<code>ADMIN_OPTION</code>	Permiso recibido con la cláusula <code>WITH ADMIN OPTION</code> (YES o NO).

#### **SESSION\_PRIVS**

<code>PRIVILEGE</code>	Nombre del permiso.
------------------------	---------------------

#### **SYSTEM\_PRIVILEGE\_MAP**

<code>NAME</code>	Nombre del permiso.
-------------------	---------------------

### b. Permisos de objeto

Varias vistas del diccionario de datos permiten obtener información de los permisos de objeto:

- `DBA_TAB_PRIVS`: permisos de objeto asignados a los usuarios (o a los roles) sobre la totalidad del objeto;
- `DBA_COL_PRIVS`: permisos de objeto asignados a los usuarios (o a los roles) únicamente sobre algunas columnas del objeto;
- `TABLE_PRIVILEGE_MAP`: lista de todos los permisos de objeto.

Las columnas interesantes de las diferentes vistas se presentan a continuación.

#### **DBA\_TAB\_PRIVS**

<code>GRANTEE</code>	Nombre del usuario o del rol que ha recibido el permiso de objeto.
<code>OWNER</code>	Nombre del usuario propietario del objeto.
<code>TABLE_NAME</code>	Nombre del objeto (no es forzosamente una tabla, a pesar del nombre).
<code>GRANTOR</code>	Nombre del usuario que ha asignado el permiso.



PRIVILEGE	Permiso del objeto recibido.
GRANTABLE	Permiso recibido con la cláusula WITH GRANT OPTION (YES o NO).

### **DBA\_COL\_PRIVS**

GRANTEE	Nombre del usuario o del rol que ha recibido el permiso de objeto.
OWNER	Nombre del usuario propietario del objeto.
TABLE_NAME	Nombre del objeto (tabla o vista).
COLUMN_NAME	Nombre de la columna.
GRANTOR	Nombre del usuario que ha asignado el permiso.
PRIVILEGE	Permiso del objeto recibido.
GRANTABLE	Permiso recibido con la cláusula WITH GRANT OPTION (YES o NO).

### **TABLE\_PRIVILEGE\_MAP**

NAME	Nombre del permiso.
------	---------------------

## **c. Roles**

Varias vistas del diccionario de datos permiten obtener información de los roles:

- DBA\_ROLES: lista de los roles existentes en la base de datos;
- DBA\_APPLICATION\_ROLES: descripción de los roles activados por un paquete;
- DBA\_CODE\_ROLE\_PRIVS: lista de los roles asignados a programas almacenados.
- DBA\_SYS\_PRIVS: permisos de sistema asignados a los roles (o a los usuarios), ver más atrás;
- DBA\_TAB\_PRIVS: permisos de objeto asignados a los roles (o a los usuarios), sobre la totalidad del objeto, ver más atrás;
- DBA\_COL\_PRIVS: permisos de objeto asignados a los roles (o a los usuarios), únicamente sobre algunas columnas del objeto, ver más atrás;
- DBA\_ROLE\_PRIVS: roles asignados a los usuarios o a los roles;
- ROLE\_SYS\_PRIVS: permisos de sistema asignados a los roles (solo para los roles a los que el usuario tiene acceso);
- ROLE\_TAB\_PRIVS: permisos de objeto asignados a los roles (solo para los roles a los que el usuario tiene acceso);
- ROLE\_ROLE\_PRIVS: roles asignados a otros roles (solo para los roles a los que el usuario tiene acceso);
- SESSION\_ROLES: roles activos actualmente en la sesión.

Las columnas interesantes de las diferentes vistas se presentan a continuación.

### **DBA\_ROLES**

ROLE	Nombre del rol.
PASSWORD_REQUIRED	Indica si es necesaria una contraseña para activar el rol (YES, NO o

GLOBAL).

### **DBA\_APPLICATION\_ROLES**

ROLE	Nombre del rol.
SCHEMA	Esquema del paquete utilizado para la activación.
PACKAGE	Nombre del paquete utilizado para la activación.

### **DBA\_CODE\_ROLE\_PRIVS**

OWNER	Propietario del programa almacenado.
OBJECT_NAME	Nombre del programa almacenado.
OBJECT_TYPE	Tipo (PROCEDURE, FUNCTION, PACKAGE) del programa almacenado.
ROLE	Rol asignado al programa almacenado.

### **DBA\_ROLE\_PRIVS**

GRANTEE	Nombre del usuario o del rol que ha recibido el rol.
GRANTED_ROLE	Nombre del rol recibido.
ADMIN_OPTION	Rol recibido con la cláusula WITH ADMIN OPTION (YES o NO).

### **ROLE\_SYS\_PRIVS**

ROLE	Nombre del rol.
PRIVILEGE	Nombre del permiso de sistema recibido a través del rol.
ADMIN_OPTION	Permiso recibido con la cláusula WITH ADMIN OPTION (YES o NO)

### **ROLE\_TAB\_PRIVS**

ROLE	Nombre del rol.
OWNER	Nombre del usuario propietario del objeto.
TABLE_NAME	Nombre del objeto (no necesariamente una tabla, a pesar del nombre).
COLUMN_NAME	Nombre de la columna (si aplica).
PRIVILEGE	Permiso del objeto recibido.

### **ROLE\_ROLE\_PRIVS**

ROLE	Nombre del rol.
GRANTED_ROLE	Nombre del rol asignado al rol.
ADMIN_OPTION	Rol recibido con la cláusula WITH ADMIN OPTION (YES o NO).

**SESSION\_ROLES**

ROLE	Nombre del rol.
------	-----------------

# Resumen

## 1. Las diferentes tipos de cuentas

En general, una base Oracle contiene cuatro tipos de cuentas.

### Administración

Una cuenta como esta tiene todos los permisos de sistema necesarios, fundamentalmente para la gestión de las estructuras de almacenamiento y la gestión de los usuarios. Además, las cuentas de administración tienen un acceso completo al diccionario de datos.

Estos permisos se pueden obtener por medio del rol DBA o de un rol equivalente.

### Desarrollo/alojamiento del esquema de aplicaciones

Una cuenta como esta tiene los permisos de sistema necesarios para la creación de los diferentes tipos de objetos (tablas, vistas, procedimientos...) y tiene una cuota sobre, al menos, un tablespace. Los permisos de sistema necesarios se pueden obtener por medio de los roles CONNECT y RESOURCE o a través de un rol equivalente que usted haya creado (aconsejable).

Para las cuentas de desarrollo, puede ser aconsejable prever un tablespace dedicado y definir una cuota solo sobre este tablespace. En la situación ideal, es preferible utilizar una base de datos a parte para el desarrollo.

La cuenta "propietaria" de una aplicación generalmente tiene cuotas ilimitadas en los tablespaces (de tablas e índices) dedicados a la aplicación.

### Usuario final

Una cuenta como esta necesita pocos permisos de sistema: CREATE SESSION (obligatorio), ALTER SESSION (algunas veces necesaria) y generalmente es todo. Por el contrario, tiene los permisos de objeto para los objetos del esquema de aplicaciones, generalmente por medio de un rol.



El permiso de sistema ALTER SESSION no es necesario para ejecutar la sentencia SQL ALTER SESSION SET CURRENT\_SCHEMA o las sentencias SQL que modifican los argumentos NLS de la sesión (como ALTER SESSION SET nls\_date\_format = ... por ejemplo).

Las cuentas de los usuarios finales no necesitan ninguna cuota en los tablespaces. Acceden a los objetos del esquema de la aplicación gracias a los permisos de objeto y la escritura de las consultas se simplifica usando sinónimos públicos o ejecutando la sentencia SQL ALTER SESSION SET CURRENT\_SCHEMA.

Los perfiles también se pueden utilizar para controlar el uso de algunos recursos o implementar una política de gestión de contraseñas.

## 2. Algunos consejos para dotar de seguridad a su base de datos

A continuación se mencionan algunos consejos sencillos (habitualmente de sentido común), que permiten dotar de seguridad a su base de datos:

- Limitar los accesos al servidor (fundamentalmente al archivo de contraseñas, al archivo de argumentos y a los archivos de traza o de alerta de cada instancia Oracle).
- Prohibir la autenticación por el sistema operativo a través de la red (el argumento REMOTE\_OS\_AUTHENT debe ser igual a FALSE).
- Bloquee y deje caducada la contraseña de las cuentas por defecto que no se utilicen (por defecto, es el caso cuando crea una base de datos con el asistente gráfico Configuración de base de datos). Las únicas cuentas por defecto que obligatoriamente deben estar abiertas son SYS y SYSTEM.
- Modifique las contraseñas por defecto de las cuentas por defecto que utilice (en primer lugar SYS y SYSTEM). Puede consultar la vista DBA\_USERS\_WITH\_DEFPWD para obtener la lista de usuarios que todavía tienen su contraseña por defecto.
- Utilice contraseñas complejas (10 caracteres al menos, con letras mayúsculas, minúsculas, cifras y caracteres especiales).
- Utilice una política de gestión de contraseñas, con un número limitado de intentos erróneos de conexión permitidos y reglas de complejidad de las contraseñas.
- No almacene las contraseñas sin cifrar en tablas o scripts.
- Asigne la menor cantidad de permisos posible a los usuarios (fundamentalmente permisos de sistema ANY).
- Utilice roles para gestionar los permisos. Si el rol está activado por una aplicación, protéjalo con una contraseña.
- Defina sus propios roles y no utilice los roles predefinidos por Oracle.
- Solo asigne un rol a un usuario si realmente necesita todos los permisos contenidos en el rol. Si no es el caso, cree otro rol más restrictivo.
- No asigne ningún permiso de sistema a PUBLIC (no tiene ninguno por defecto).
- No asigne ningún permiso de objeto a PUBLIC (diferente a los asignados por defecto).
- Revoque los permisos EXECUTE que varios paquetes asignan por defecto a PUBLIC, potencialmente peligrosos para la seguridad: DBMS\_LOB, DBMS\_JOB, UTL\_FILE, UTL\_HTTP, UTL\_TCP, UTL\_SMTP. Este punto es complejo porque estos paquetes se utilizan por numerosas cuentas Oracle (SYS principalmente). Si elimina el permiso EXECUTE asignado a PUBLIC en estos paquetes tendrá que volver a asignar los permisos necesarios directamente a los usuarios afectados (puede consultar la vista DBA\_DEPENDENCIES para conocer las cuentas que utilizan estos paquetes).

# Supervisar los usuarios conectados

La vista V\$SESSION permite identificar los usuarios conectados actualmente:

SQL> SELECT sid,serial#,username,osuser,status FROM v\$session;				
SID	SERIAL#	USERNAME	OSUSER	STATUS
-----				
1	3		SYSTEM	ACTIVE
...				
10	10494	VDEP	vdep	INACTIVE
14	450	SYSTEM	Administrator	ACTIVE

 Las sesiones sin valor en la columna USERNAME corresponden a procesos batch.

Las columnas interesantes de la vista V\$SESSION son las siguientes:

SID	Identificador de la sesión.
SERIAL#	Número de serie de la sesión.
USERNAME	Nombre del usuario (cuenta Oracle).
SCHEMANAME	Nombre del esquema del usuario (puede ser diferente de USERNAME si la sesión ha ejecutado la sentencia SQL ALTER SESSION SET CURRENT_SCHEMA).
STATUS	Estado de la sesión (ACTIVE, INACTIVE o KILLED).
LOGON_TIME	Fecha y hora de conexión.
OSUSER	Nombre del usuario, a nivel del sistema operativo.
MACHINE	Nombre de la máquina del usuario, a nivel del sistema operativo.
TERMINAL	Nombre del terminal del usuario, a nivel del sistema operativo.
PROGRAM	Nombre del programa utilizado por el usuario para conectarse a la base de datos.
SERVER	Tipo de proceso de servidor (DEDICATED o SHARED).
SQL_ID	Identificador de la consulta SQL que se está ejecutando (un join con V\$SQL o V\$SQLAREA sobre la misma columna permite ver la sentencia SQL en cuestión).
SERVICE_NAME	Nombre de servicio de la sesión (servicio al que está conectado el usuario).

Si es necesario, también puede consultar la vista V\$SESSION\_LONGOPS para obtener información relativa a las operaciones largas (en ejecución desde hace más de 6 segundos). Para desconectar a un usuario puede utilizar la sentencia SQL ALTER SYSTEM.

### Sintaxis

```
ALTER SYSTEM KILL SESSION 'sid,serial#';
```

O

```
ALTER SYSTEM DISCONNECT SESSION 'sid,serial#'  
{ IMMEDIATE | POST_TRANSACTION};
```

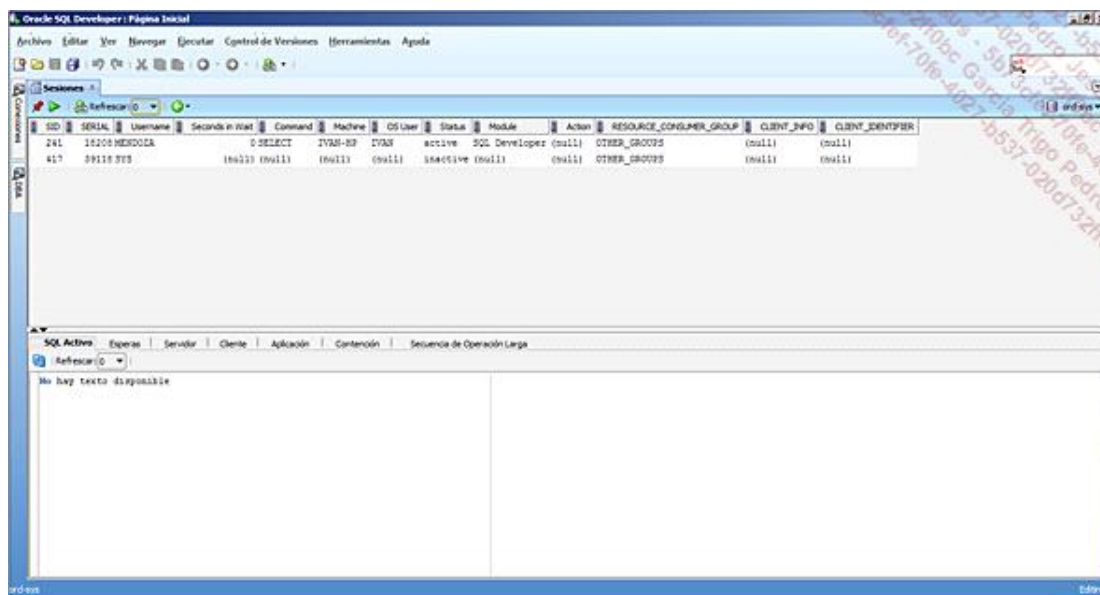
### Ejemplo:

```
ALTER SYSTEM KILL SESSION '10,10494';
```

Las sentencias SQL `ALTER SYSTEM KILL SESSION` y `ALTER SYSTEM DISCONNECT SESSION ... IMMEDIATE` son equivalentes: cierran la sesión inmediatamente, sin esperar al final de una eventual transacción en ejecución (esta última se anula). Por el contrario, la sentencia SQL `ALTER SYSTEM DISCONNECT SESSION ... POST_TRANSACTION`, espera a que la transacción en ejecución termine.

Un usuario que está ejecutando una consulta tiene un estado `ACTIVE` (`INACTIVE`, en caso contrario). Si un usuario se desconecta mientras está activo, su consulta se interrumpe y un mensaje de error le indica que ha sido desconectado (`ORA-00028: su sesión se ha cerrado`) y la sesión desaparece de `V$SESSION`. Si está inactivo, la conexión se cierra, pero la sesión sigue estando visible en `V$SESSION` con el estado `KILLED`, hasta que se notifique al usuario la desconexión durante su próxima acción (con el mismo error `ORA-00028`).

En Oracle SQL Developer, puede seleccionar el elemento **Controlar sesiones...** del menú **Herramientas** con el objetivo de mostrar la lista de sesiones diferentes a las que se corresponden con procesos batch:



En la parte inferior del panel **Sesiones**, varias pestañas permiten obtener información acerca de la actividad de la sesión.

Haciendo clic con el botón derecho en un registro de la lista se abre un menú contextual que fundamentalmente permite cerrar la sesión (menú **Matar sesión**).

# Utilizar EM Express

## 1. Usuarios

En EM Express, seleccione el elemento **Usuarios** del menú **Seguridad** para acceder a la página de gestión de usuarios:

Nombre	Estado de la Cuenta	Fecha de Caducidad	Tablespace por Defecto	Tablespace Temporal	Perfil	Creación
ANONYMOUS	🔒	jue 11 de sep de 2014 11:15:19	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:54:42
APEX_040200	🔒	jue 11 de sep de 2014 10:33:25	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 10:29:19
APEX_PUBLIC_USER	🔒	jue 11 de sep de 2014 10:29:19	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 10:29:19
APPQOSSYS	🔒	jue 11 de sep de 2014 8:54:34	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:54:34
AUDSYS	🔒	jue 11 de sep de 2014 8:40:57	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 8:40:57
BI	🔒	dom 30 de nov de 2014 19:33:46	USERS	TEMP	DEFAULT	dom 30 de nov de 2014 19:33:46
CTXSYS	🔒	dom 30 de nov de 2014 19:33:46	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:40:42
DESKMP	🔒	jue 11 de sep de 2014 8:54:32	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:54:32
DIP	🔒	jue 11 de sep de 2014 8:44:33	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 8:44:33
DVF	🔒	jue 11 de sep de 2014 11:15:19	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 11:15:19
DVSYS	🔒	jue 11 de sep de 2014 11:15:19	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 11:15:19
FLOWSEEDS	🔒	jue 11 de sep de 2014 10:29:19	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 10:29:19
GPMACHTN_INTERNAL	🔒	jue 11 de sep de 2014 8:44:25	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:44:25
GPMACHTN_USER	🔒	jue 11 de sep de 2014 8:58:54	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 8:58:54
GPMACHTN	🔒	jue 11 de sep de 2014 8:44:25	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 8:44:25
HR	🔒	dom 30 de nov de 2014 19:33:46	USERS	TEMP	DEFAULT	dom 30 de nov de 2014 19:33:46
DI	🔒	dom 30 de nov de 2014 19:33:46	USERS	TEMP	DEFAULT	dom 30 de nov de 2014 19:33:46
IMACSYS	🔒	jue 11 de sep de 2014 11:15:19	SYSTEM	TEMP	DEFAULT	jue 11 de sep de 2014 11:15:19
MDSYS	🔒	jue 11 de sep de 2014 11:15:19	USERS	TEMP	DEFAULT	jue 11 de sep de 2014 11:15:19
MDSYS	🔒	jue 11 de sep de 2014 8:40:50	SYSAUX	TEMP	DEFAULT	jue 11 de sep de 2014 8:40:50
OE	🔒	dom 30 de nov de 2014 19:33:46	EXAMPLE	TEMP	DEFAULT	dom 30 de nov de 2014 19:33:46

Desde esta página puede realizar diversas acciones relacionadas con la gestión de usuarios:

Acciones	Estado de la Cuenta	Fecha de Caducidad	Tablespace por Defecto
🔒	🔒	jue 11 de sep de 2014 11:15:19	SYSAUX
🔒	🔒	jue 11 de sep de 2014 10:33:25	SYSAUX
🔒	🔒	jue 11 de sep de 2014 10:29:19	USERS
🔒	🔒	jue 11 de sep de 2014 8:54:34	SYSAUX
🔒	🔒	jue 11 de sep de 2014 8:40:57	USERS
🔒	🔒	dom 30 de nov de 2014 19:33:46	USERS
🔒	🔒	dom 30 de nov de 2014 19:33:46	SYSAUX

- crear un nuevo usuario (menú o botón **Crear por**);
- crear un nuevo usuario idéntico a otro usuario existente (menú o botón **Crear como**);
- eliminar un usuario (menú o botón **Borrar Usuario**);
- mostrar los detalles de un usuario (menú **Ver Detalles** o clic en el nombre del usuario en la lista);
- modificar las propiedades básicas (autenticación, contraseña o perfil) de un usuario (menú **Modificar Cuenta**);
- modificar los tablespaces por defecto de un usuario (menú **Modificar Tablespaces**);
- modificar los permisos de sistema y los roles asignados a un usuario (menú **Modificar Privilegios y Roles**);
- modificar los permisos de objeto asignados a un usuario (menú **Otorgar Privilegios de Objeto**).

Existen diferentes cuadros de diálogo para realizar estas acciones:

- Crear un usuario:



**Creado por**

☒ **Cuenta de Usuario**
☐ Tablespaces
 ☐ Privilegio

Nombre \*

Autenticación \* ☒ Contraseña ☐ Externo ☐ Global

Contraseña \*

Confirmar Contraseña \*

Perfil

Contraseña Caducada ☐

Cuenta Bloqueada ☐

- Modificar la cuenta:

**Modificar Cuenta**

Nombre \*

Autenticación ☒ Contraseña ☐ Externo ☐ Global

Contraseña

Confirmar Contraseña

Perfil

Contraseña Caducada ☐

Cuenta Bloqueada ☐

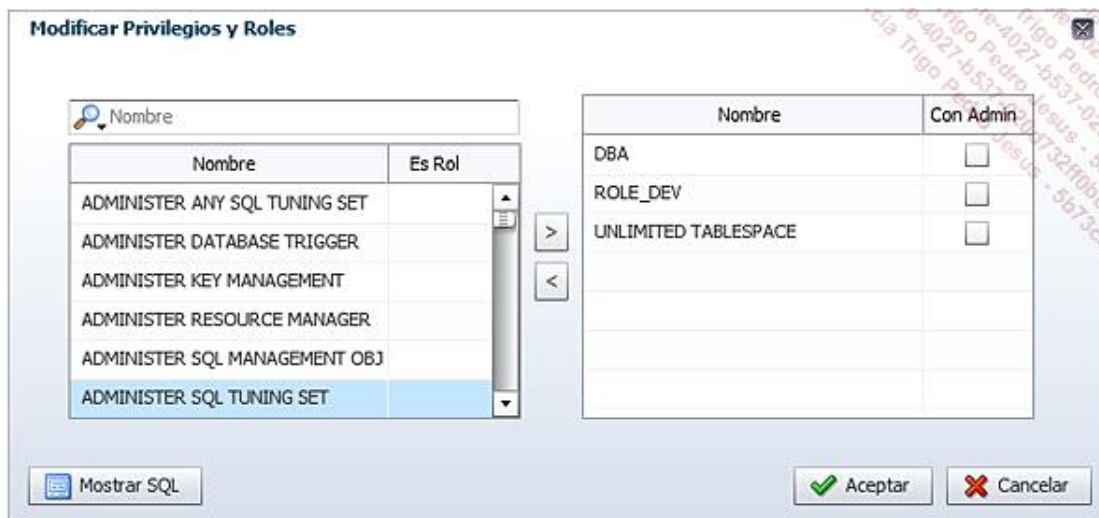
- Modificar los tablespaces:

**Modificar Tablespaces**

Tablespace por Defecto

Tablespace Temporal

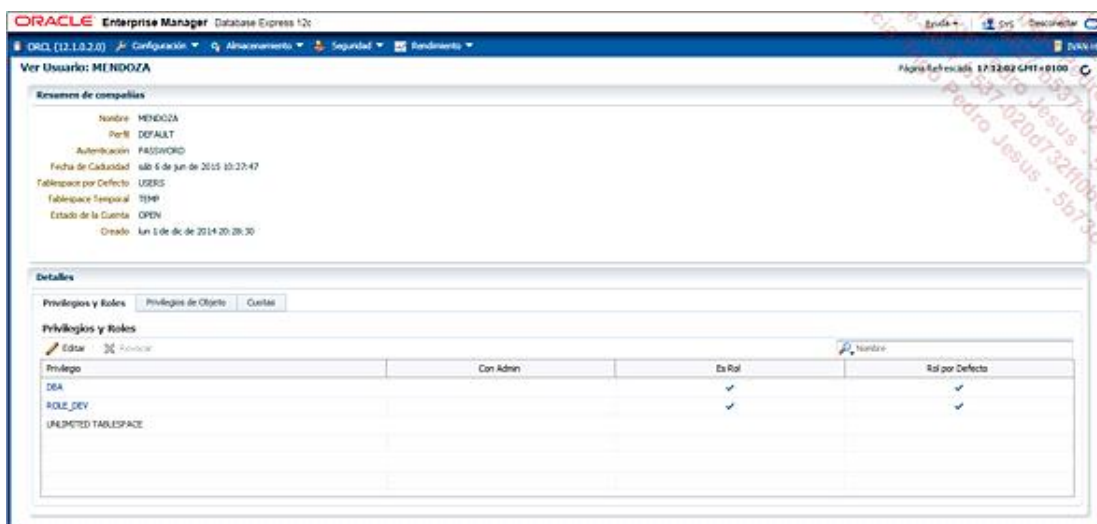
- Modificar los privilegios y los roles:



- Asignar privilegios de objeto:



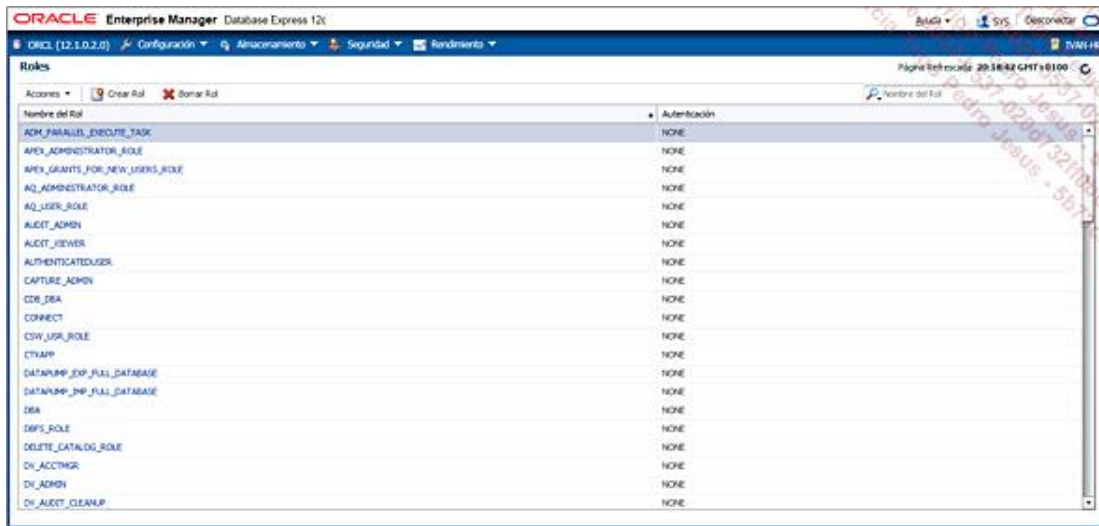
Haciendo clic en el enlace del nombre del usuario o seleccionando el elemento **Ver detalles**, del menú **Acciones**, se accede a una página que permite consultar las características de un usuario:



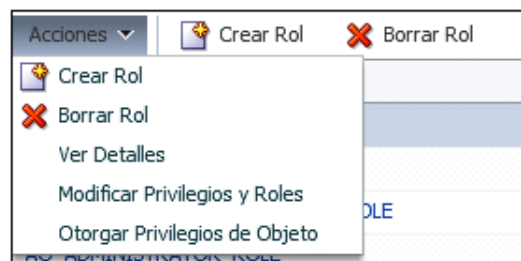
Las pestañas del panel **Detalles** permiten consultar y modificar los permisos del usuario.

## 2. Roles

En EM Express, seleccione el elemento **Roles** del menú **Seguridad** para acceder a la página de gestión de roles:



Desde esta página puede realizar diversas acciones relacionadas con la gestión de roles:



- crear un nuevo rol (menú o botón **Crear Rol**);
- eliminar un rol (menú o botón **Borrar Rol**);
- mostrar los detalles de un rol (menú **Ver Detalles** o clic en el nombre del rol en la lista);
- modificar los permisos de sistema y los roles asignados a un rol (menú **Modificar Privilegios y Roles**);
- modificar los permisos de objeto asignados a un rol (menú **Otorgar Privilegios de Objeto**).

Hay diferentes cuadros de diálogo para realizar estas acciones:

- Crear un rol:



- Modificar privilegios y roles:

**Modificar Privilegios y Roles**

Nombre

Nombre	Es Rol
ADMINISTER ANY SQL TUNING SET	
ADMINISTER DATABASE TRIGGER	
ADMINISTER KEY MANAGEMENT	
ADMINISTER RESOURCE MANAGER	
ADMINISTER SQL MANAGEMENT OB.	
ADMINISTER SQL TUNING SET	

> <

Nombre	Con Admin
CREATE SESSION	<input type="checkbox"/>
CREATE TABLE	<input type="checkbox"/>
CREATE VIEW	<input type="checkbox"/>
CREATE PROCEDURE	<input type="checkbox"/>
CREATE SEQUENCE	<input type="checkbox"/>
CREATE TRIGGER	<input type="checkbox"/>

Mostrar SQL

Aceptar Cancelar

- Asignar privilegios de objeto:

**Otorgar Privilegios de Objeto**

Seleccionar Esquema y Tipo de Objeto    Seleccionar Objetos    Otorgar Privilegios de Objeto

Esquema \* APPQOSSYS

Tipo de Objeto \* TABLE

Nombre del Objeto

Mostrar SQL

Aceptar Cancelar

Haciendo clic en el enlace del nombre del rol o seleccionando el elemento **Ver Detalles** del menú **Acciones**, se accede a una página que permite consultar las características de un rol:

**ORACLE Enterprise Manager Database Express 12c**

Ver Rol: ROLE\_DEV

Resumen de propiedades

Nombre del Rol: ROLE\_DEV

Autenticación: NONE

**Detalles**

Privilegios y Roles    Privilegios de Objeto

Privilegios y Roles

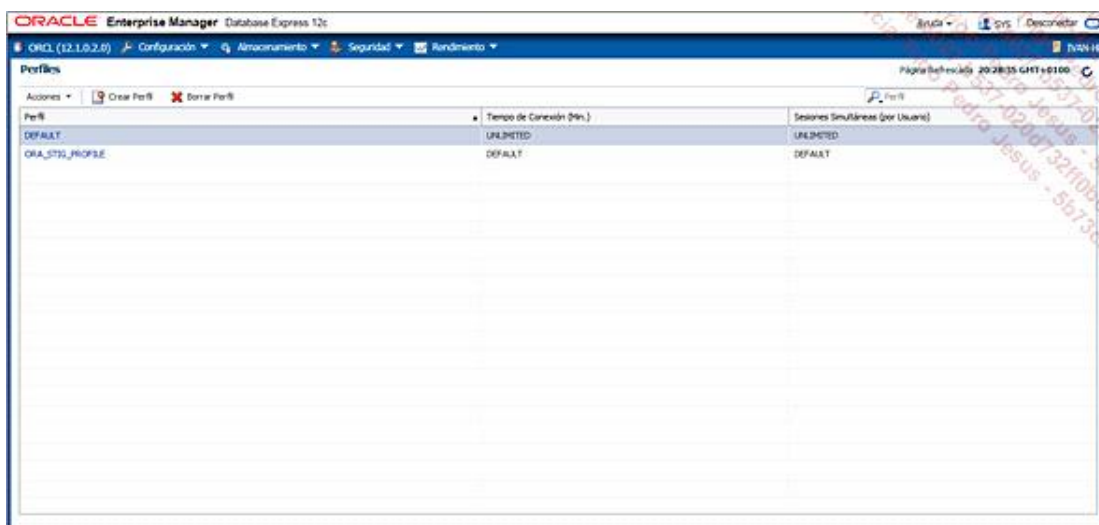
Editar    Enlace

Privilegio	Con Admin	Es Rol	Rol por Defecto
CREATE PROCEDURE			
CREATE SEQUENCE			
CREATE SESSION			
CREATE TABLE			
CREATE TRIGGER			
CREATE VIEW			

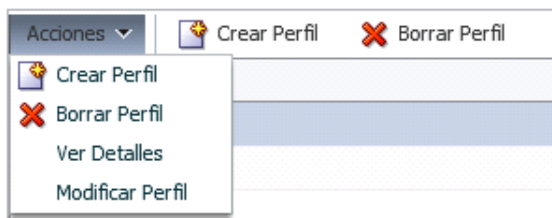
Las pestañas del panel **Detalles** permiten consultar y modificar los permisos del rol.

### 3. Perfiles

En EM Express, seleccione el elemento **Perfiles** del menú **Seguridad** para acceder a la página de gestión de perfiles:



Desde esta página puede realizar diversas acciones relacionadas con la gestión de perfiles:



- crear un nuevo perfil (menú o botón **Crear Perfil**);
- eliminar un perfil (menú o botón **Borrar Perfil**);
- mostrar los detalles de un perfil (menú **Ver Detalles** o clic en el nombre del perfil en la lista);
- modificar un perfil (menú **Modificar Perfil**).

Hay diferentes cuadros de diálogo para realizar estas acciones:

- Crear un perfil:



- Modificar el perfil:

**Modificar Perfil**

General Contraseña

CPU/Sesión (Seg./100) \* Ilimitado ▼

CPU/Llamada (Seg./100) \* Ilimitado ▼

Tiempo de Conexión (Min.) \* Ilimitado ▼

Tiempo de Inactividad (Min.) \* Ilimitado ▼

Sesiones Simultáneas (por Usuario) \* Ilimitado ▼

Lecturas/Sesión (Bloques) \* Ilimitado ▼

Lecturas/Llamada (Bloques) \* Ilimitado ▼

SGA Privada \* Ilimitado ▼ ⓘ

Límite Compuesto (Unidades de Servicio) \* Ilimitado ▼

Mostrar SQL Aceptar Cancelar

Haciendo clic en el enlace del nombre del perfil o seleccionando el elemento **Ver Detalles** del menú **Acciones**, se accede a una página que permite consultar las características de un perfil:

**Oracle Enterprise Manager Database Express 12c**

Ver Perfil: DEFAULT

**General**

Límite Compuesto (Unidades de Servicio)	UNLIMITED
SGA Privada	UNLIMITED
Tiempo de Conexión (Min.)	UNLIMITED
Tiempo de Inactividad (Min.)	UNLIMITED
Lecturas/Llamada (Bloques)	UNLIMITED
Lecturas/Sesión (Bloques)	UNLIMITED
CPU/Llamada (Seg./100)	UNLIMITED
CPU/Sesión (Seg./100)	UNLIMITED
Sesiones Simultáneas (por Usuario)	UNLIMITED

**Contraseña**

Blockear (días después de vencimiento)	7
Número de Días de Blockear	3
Función de Complejidad	NONE
Número de Contraseñas a Mantener	UNLIMITED
Número de Días a Mantener	UNLIMITED
Intentar en (días)	30
Número de Intentos de Conexión fallidos para Blockear después	30