USUARIOS. ACCESO A LA INFORMACIÓN

Índice

1.- INTRODUCCIÓN

En temas anteriores, hemos hablado del diccionario de datos o catálogo. Desde el punto de vista de una administrador de base de datos, las consultas al diccionario de datos son muy importantes, porque en ellas se proporciona información sobre el contenido de la base de datos (estructura de almacenamiento, usuarios, valor de sus parámetros, etc.)

El diccionario de datos pertenece al usuario SYS y se almacena en el tablespace SYSTEM.

El diccionario de datos (DD) se modifica cada vez que lanzamos una sentencia DDL.

Tiene dos tipos de vistas: estáticas y dinámicas.

Las vistas estáticas que forman parte del DD son de tres tipos: dba, all y user. Cada una de ellas tiene un prefijo asociado que la ubica en uno de dichos tipos.

- DBA_: todos los objetos de la BD.
- ALL_: todos los objetos accesibles por el usuario actual.
- USER_: todos los objetos propiedad del usuario actual.

Hay una vista, llamada DICTIONARY, que contiene una lista de todas las vistas del DD.

La vista DICT_COLUMNS muestra el significado de las columnas de cada vista.

Algunos ejemplos de vistas del DD son:

Objetos de la BD: dba_objects, dba_tables, dba_indexes, dba_tab_columns, dba_views dba_ind_columns, dba_constraints,.

Espacio ocupado: dba_data_files, dba_segments, dba_extents.

Estructura de la BD: dba_tablespaces, dba_data_files.

Las vistas dinámicas, se diferencia porque tiene el prefijo V$(como V$SESSION). Se crean al arrancar la instancia y residen en memoria. Cuando cerramos la BD (y por tanto la instancia), desaparecen y con ellas su contenido.

2- USUARIOS

Un usuario es un nombre definido en la base de datos que se puede conectar a ella y acceder a determinados objetos. Todo usuario se identifica por un nombre y una contraseña

Debemos tener en cuenta que al momento de crear la base de datos se crean automáticamente dos usuarios con todos los privilegios, SYS y SYSTEM. Para crear algún usuario o rol, o asignar privilegios o roles debemos conectarnos con alguno de estos dos usuarios.

El usuario SYS es el propietario del diccionario y el SYSTEM es el encargados de las tareas de administración (creación de usuario, roles, establecer perfiles, etc.)

En la creación de usuarios se piden los siguientes requisitos:

- Nombre: nombre del usuario
- Password: se debe especificar la clave del usuario
- Profile (perfil, por defecto: default)
- Default Tablespace: Aquí se debe seleccionar el tablespace donde se crean los objetos del usuario
- Tempory Tablespace: generalmente se coloca el TableSpace temporal de la creación de la base de datos, pero puede usarse cualquiera. Hay que tener en cuenta que aquí se hacen las consultas y ordenaciones de este usuario.
- Quotas: se le indica al usuario el espacio en Kb o Mb que podrá manipular, se puede indicar un tamaño especifico o ilimitado, por defecto no puede manipularlo. No solo se puede asignar espacio a sus tablespace sino que a otros que no le pertenezcan.

Además a un usuario, le podemos asignar:

- Privilegios del sistema
- Privilegios sobre objetos
- Roles (Conjunto de privilegios)

Todos los objetos perteneciente a un usuario es lo que se denomina ESQUEMA.

CREACIÓN DE USUARIO

Hay que poseer el privilegio CREATE USER

Sintaxis:

CREATE USER nombre_usuario
IDENTIFIED BY clave_acceso
[DEFAULT TABLESPACE espacio_tabla]
[TEMPORARY TABLESPACE espacio_tabla]
[QUOTA (entero {K|M} | UNLIMITED} ON espacio_tabla]
[PROFILE perfil]
[ACCOUNT{LOCK | UNLOCK }]
[PASSWORD EXPIRE];

Para que un usuario se pueda conectar ha de poseer el privilegio CREATE SESION

GRANT create sesion TO usuario;

MODIFICACIÓN DE USUARIO

Hay que poseer el privilegio ALTER USER

Un usuario sólo puede cambiarse su clave de acceso

Sintaxis:

ALTER USER nombre_usuario
[IDENTIFIED BY clave_acceso]
[DEFAULT TABLESPACE espacio_tabla]
[TEMPORARY TABLESPACE espacio_tabla]
[QUOTA (entero {K|M} | UNLIMITED} ON espacio_tabla]
[PROFILE perfil]
[ACCOUNT{LOCK | UNLOCK }]
[PASSWORD EXPIRE];

BORRADO DE USUARIOS

Para que un usuario pueda borrar a otro usuario tiene que poseer el privilegio DROP USER

Sintaxis:

DROP USER usuario [CASCADE];

La opción cascade permite borrar los usuarios con objetos

VISTAS: ALL_USERS, DBA_USERS

3.- PRIVILEGIOS

Un privilegio es la capacidad que tiene un usuario de la base de datos para poder realizar determinadas operaciones o a acceder a determinados objetos de otros usuario. Ningún usuario puede llevar a cabo una operación si antes no se le ha concedido permiso. Hay dos tipos de privilegios:

PRIVILEGIOS SOBRE EL SISTEMA

Estos están asignados básicamente al tratamiento de objetos como crear tablas, crear vistas, etc. Cabe destacar que la mayoría de estos privilegios se encuentran en los roles del sistema.

OTORGAR PRIVILEGIOS

```
GRANT {privilegio_sistema | rol} [,{privilegio_sistema | rol}, …]
TO {usuario | rol | PUBLIC}[,{usuario | rol | PUBLIC}] …
[WITH ADMIN OPTION];
```

Ejemplo: Permitir conexión a un usuario: GRANT create session TO usuario1

RETIRAR PRIVILEGIOS SOBRE SISTEMAS

```
REVOKE privilegio_sistema | rol} [,{privilegio_sistema | rol}, …]
FROM {usuario | rol | PUBLIC}[,{usuario | rol | PUBLIC}] …;
```

Ejemplo: Retirar conexión a un usuario: REVOKE create session FROM usuario1;

PRIVILEGIOS SOBRE LOS OBJETOS.

Permite acceder y realizar cambios en los datos de otros usuarios (consultar una tabla, insertar en una tabla, etc)

OTORGAR PRIVILEGIOS

```
GRANT {priv_objeto  [,priv_objeto] ... | ALL [PRIVILEGES]}[(columna [,columna]...)]
ON [usuario.]objeto
TO {usuario | rol | PUBLIC} [,{usuario | rol | PUBLIC} … ]
[WITH GRANT OPTION];
```

Ejemplo: Conceder al usuario2 todos los privilegios sobre la tabla del usuario1

```
GRANT ALL ON usuario1.tabla1 TO usuario2
```

RETIRAR PRIVILEGIOS

```
REVOKE {priv_objeto  [,priv_objeto] ... | ALL [PRIVILEGES]}
ON [usuario.]objeto
FROM usuario | rol | PUBLIC}
 [,{usuario | rol | PUBLIC} …;
```

Ejemplo: Retirar al usuario2 todos los privilegios sobre la tabla del usuario1

```
REVOKE ALL ON usuario1.tabla1 FROM usuario2;
```

4.- ROLES

Son un conjunto de privilegios que se asignan a los usuarios para trabajar en su entorno. Por defecto, vienen predefinidos una serie de roles, pero también se pueden crear roles personalizados.

A continuación se detallan algunos de los roles predefinidos:

- DBA: corresponde al administrador, es el que tiene asignado implícitamente todos los privilegios del sistema y demás roles. Los usuarios SYS y SYSTEM tienen este rol. Puede realizar todas las operaciones sobre almacenamiento, usuarios, objetos, etc.
- CONNECT: este rol permite al usuario conectarse a la base de datos y abrir una sesión.
- RESOURCE: este rol le posibilita al usuario tener un conjunto de privilegios de sistema como crear tablas, procedimientos, dar espacio ilimitado a los tablespace, etc.
- EXP_FULL_DATABASE: este rol se asigna a usuarios para que puedan realizar backups de la base de datos usando la utilidad EXP.
- IMP_FULL_DATABASE: este rol se asigna a usuarios para que puedan restaurar backups usando la utilidad IMP.

A los roles también se le pueden anexar otros roles y heredar sus privilegios.

- CREACIÓN DE ROL

  CREATE ROLE nombreRol [IDENTIFIED BY contraseña]

- SUPRESION DE ROL

  DROP ROLE nombreRol

- ESTABLECER ROL POR DEFECTO

  ALTER USER NombreUsuario
  DEFAULT {[ROLE nombreRol] | [NONE] };

5.- PERFILES

Un perfil es un conjunto de límites a los recursos de la base de datos. Algunos de los recursos que podemos limitar son:

- Tiempo de CPU: se indica en seg/100 el tiempo máximo que el usuario puede usar la cpu.
- Tiempo de Conexión: se indica el tiempo máximo de conexión en minutos.
- Tiempo Desocupado: se indica en minutos el tiempo máximo que el usuario puede permanecer desocupado.
- Sesiones concurrentes: la cantidad máxima se sesiones conectadas a la base de datos.

CREACIÓN DE PERFIL

CREATE PROFILE nombreperfil LIMIT
{SESSIONS_PER_USER | //número de sesiones múltiples concurrentes
CPU_PER_SESSION | // Limita el tiempo máximo de cpu por sesión
CPU_PER_CALL | // Limita el tiempo máximo de cpu por llamada
CONNECT_TIME | // Limita el tiempo de conexión permitido por sesión
IDLE_TIME | // limita el tiempo de inactividad permitido
FAILED_LOGIN_ATTEMPTS | // número de fallo permitido antes de bloquerlo
LOGICAL_READS_PER_SESSION | // bloque datos leído por sesion
LOGICAL_READS_PER_CALL | // bloque datos leídos por llamada
PRIVATE_SGA | // nº de bytes enteros de espacio privado en SGA
COMPOSITE_LIMIT } // limita coste total de recursos
{Entero [K|M] | UNLIMITED | DEFAULT} ;

BORRADO DE PERFIL

DROP PROFILE nombreperfil [CASCADE].
Con la opción cascade se borra el perfil aunque haya usuario asignado a dicho perfil

Para activar el uso de perfil en el sistema el parámetro RESOURCE_LIMIT ha de estar a TRUE

alter system set resource_limit=true;

Cada usuario puede tener un único perfil.
Antes de asignar un perfil a un usuario es necesario que este perfil exista en la base de datos.

A los usuarios se le asigna el perfil DEFAULT por defecto que define recursos ilimitado

6.- OBJETOS DEL USUARIO

Una vez creada una base de datos y un usuario cualquiera, podemos pasar al paso siguiente que es la creación de objetos para dicho usuario. La creación de objetos esta íntimamente ligada a los usuarios de la Base de Datos. Se dice que dichos usuarios tienen un esquema de trabajo.

Entre los objetos mas comunes de un usuario Oracle, se encuentran: Tablas[1], Índices, Vistas, Sinónimos, Secuencias, …

6.1.-VISTAS

Una vista es una tabla lógica que permite acceder a la información de una o varias tablas. No contiene información por sí misma, sino que su información está basada en la que contienen otras tablas llamadas tablas base y siempre refleja los datos de estas tablas. Si se suprime la tabla, la vista asociada se invalida, pero no se borra.

Al ser una tabla lógica no necesita espacio de almacenamiento para los datos.

Creación de Vistas

```
CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW [usuario.]NombreVista
[(Columna [,columna])] AS
(Subconsulta) [WITH READ ONLY];
```

Borrado Vistas

```
DROP VIEW [usuario.]NombreVista [CASCADE CONSTRAINTS[2]]
```

Alter view

Sólo se puede usar para las constraints o para compilarla después de modificar la estructura de una de las tablas asociadas a ellas.

```
ALTER VIEW [usuario.] view
      { ADD out_of_line_constraint
        | MODIFY CONSTRAINT constraint { RELY | NORELY }
        | DROP { CONSTRAINT constraint
        | PRIMARY KEY
        | UNIQUE (column [, column ]...)
        }
   | COMPILE
  } ;
```

1 Creación de tabla dada en el repaso sql
2 Elimina todas las restricciones de integridad

Manipulación de datos en las vistas:

a.- Consultas de datos en las vistas: Se usa el mismo procedimiento que para las tablas

b.- Borrado de datos en las vistas: Se usa el mismo procedimiento que para las tablas, pero esta se deberán de haber creados:

b.1.- Con filas de una sola tabla

b.2.- Sin haber utilizado la cláusula GROUP BY ni DISTINCT

b.3.- Sin usar funciones de grupos

c.- Actualización datos: Se usa la misma sintaxis que en las tablas

c.1.- Hay que tener en cuenta las mismas restricciones que en el borrado

c.2.- Ninguna de las columnas que se van a actualizar puede estar definida como una expresión

d.- Inserción datos: Se usa la misma sintaxis que en las tablas

d.1.- Hay que tener en cuenta las mismas restricciones anteriores

d.2.- Todas las columnas obligatorias de las tablas implicadas deben estar presentes en la vista.

6.2.- SINÓNIMOS

Un sinónimo es asignar un alias a un objeto. Resulta muy útil cuando se tiene acceso a tablas de otro usuarios.

Creación de Sinónimos

```
CREATE [OR REPLACE ] [PUBLIC] SYNONYM [schema.]NombreSinonimo
FOR [usuario.] NombreTabla;
```

Con la cláusula PUBLIC se crea un sinónimo disponible para todos los usuario, siempre que tenga los privilegios para ellos

Borrado de Sinónimos

```
DROP [PUBLIC] SYNONYM [usuario.]sinónimo [FORCE]
```

6.3- ÍNDICES

Un índice es una estructura diseñada para obtener un acceso más rápido a los datos contenidos dentro de una tabla. Son independientes de los datos almacenados en la tabla. Se mantienen de acuerdo a las inserciones, actualizaciones y eliminaciones de registros de la tabla en la cual se ha implementado.

Hay dos tipos de índices:

1. Implícitos: Se crean con las cláusulas primary key, unique y foreign key
2. Explícitos: Los crea el usuario, estos pueden ser de una sola columna (Simples) o de varias columnas (Compuestos)

Se aconseja crear índices en campos que:

- Contengan una gran cantidad de valores que no estén repetidos.
- Contengan una gran cantidad de nulos, pero las consultas casi siempre seleccionan las filas que tienen algún valor.
- Son parte habitual de cláusulas WHERE, GROUP BY u ORDER BY.
- Son parte de listados de consultas de tablas grandes sobre las que casi siempre se muestran menos del 15% de su contenido.
- Columnas que sirven para unir una tabla con otras

No se aconseja en campos que:

- Pertenezcan a tablas pequeñas
- No se usan a menudo en las consultas
- Pertenecen a tablas que se actualizan frecuentemente
- utilizan expresiones

Las columnas tipo LONG o LONG RAW no pueden ser indexadas

Creación de índices

```
CREATE [UNIQUE] INDEX nombre_indice  ON tabla (columna1 [,columna2...])
[TABLESPACE nombretablespace]
```

La clausula unique indica que la columna (o columnas) por la que indexamos, no admite valores repetidos (es similar a la clausula única en la creación de la tabla)

Ejemplo: CREATE INDEX nombre_completo ON UPERSONA (apellido, nombre);

Modificación de índices

Un índice se puede modificar para:

- Renombrarlo

  ALTER INDEX nombre_indice RENAME TO nuevo_nombre_indice
- Inutilizarlo para que no se aplique sobre una tabla

ALTER INDEX nombre_indice UNUSUABLE

- Reconstruirlo

  ALTER INDEX nombre_indice REBUILD

Si se quiere modificar las columnas que forman parte del índice la única opción es borrar y volver a crear el índice

Borrado de Indices

DROP INDEX nombre_indice [FORCE]

La clausula force permite borrar índices que están siendo utilizados

6.4.- SECUENCIAS

Una secuencia es un objeto de la base de datos que sirve para generar números enteros únicos; es muy util para generar automáticamente valores en las claves primarias.

Creación de Secuencia

```
CREATE SEQUENCE [schema.]nombre_secuencia
INCREMENT BY entero  //cantidad de incremento de la secuencia
START WITH entero   //valor de comienzo de la secuencia
[MINVALUE entero | NOMINVALUE]  // mínimo de la secuencia
[MAXVALUE entero | NOMAXVALUE] // máximo de la secuencia
[NOCYCLE | CYCLE]    // no reanuda | reanuda la secuencia
[CACHE   entero| NOCACHE]  //indica cuantos valores se  almacenan en memoria para una
                            acceso mas rápido
```

Modificación de Secuencia

```
ALTER SEQUENCE [schema.]nombre_secuencia
INCREMENT BY entero
[MINVALUE entero | NOMINVALUE]
[MAXVALUE entero | NOMAXVALUE]
[NOCYCLE | CYCLE]
[CACHE  entero| NOCACHE]
```

Borrado de secuencia

```
DROP SECUENCE nombre_secuencia
```

Las Pseudocolumnas utilizadas en la secuencias son:

nombre_secuencia.CURRVAL // Devuelve el valor actual de la secuencia

nombre_secuencia.NEXTVAL // Devuelve el siguiente valor y aumenta la secuencia.

7.- PAPELERA RECICLAJE

Cuando se borra un objeto va a la palera de reciclaje

Todos los objetos que hay en la papelera, se pueden ver con la vista RECYCLEBIN (equivalente a users_RECYCLEBIN)

El campo CAN_UNDROP me permite averiguar si un objeto se puede recuperar

Para recuperar una tabla de la papelera de reciclaje se usa la sentencia:

```
FLASHBACK TABLE nombretabla TO BEFORE DROP
```

Para vaciar la papelera de reciclaje, se usa la sentecia:

```
PURGE RECYCLEBIN.
```

Nota: Cuando se borrar un usuario, los objetos que pertenecen a ese usuario no se colocan en la papelera de reciclaje y se purgan todos los objetos de la papelera de reciclaje.