## 4.6. Lenguaje de control de datos

El lenguaje de control de datos de un sistema de gestor de bases de datos es el conjunto de comandos que permite el control de acceso a los datos a través de las cuentas de usuarios que acceden a ellos. Este control se realiza a través de mecanismos asociados a los privilegios asignables a un usuario en relación a cómo puede gestionar los objetos de la base de datos e, incluso, parte de dichos objetos. Un subconjunto de privilegios singulares con un nombre se conoce como rol y es posible usar unos existentes y crear unos nuevos.

### 4.6.1. Gestión de cuentas de usuario sobre la base de datos

Para crear un usuario se usa el comando CREATE USER usuario [opciones]; para modificar sus parámetros se emplea ALTER USER usuario [opciones], y para eliminarlo se utiliza DROP USER usuario [CASCADE].

### 4.6.2. Gestión de roles sobre usuarios

Para conceder un rol a un usuario se usa el comando GRANT con la sintaxis siguiente:

```
GRANT nombre_rol TO cuenta_usuario [IDENTIFIED BY contraseña] [WITH ADMIN OPTION];
```

La cláusula IDENTIFIED BY es opcional y se usa para indicar la contraseña de la cuenta de usuario. La opción WITH ADMIN OPTION se utiliza para que el usuario al que se le asigna un rol determinado puede asignar también el mismo rol a otros usuarios.

Por ejemplo, el siguiente comando asigna el rol CONNECT al usuario Alejandro:

```
GRANT CONNECT TO Alejandro;
```

Existe un gran número de posibles roles preestablecidos. Los más importantes se muestra en la Tabla 4.8.

**Tabla 4.8.** Listado de los roles preestablecidos

| Nombre del rol | Detalles |
|---|---|
| AUDIT_ADMIN | Permite que el usuario pueda realizar políticas de auditorías a través de `AUDIT` y `NOAUDIT`. |
| AUDIT_VIEWER | Permite ver y analizar las auditorías de datos. |

| Nombre del rol | Detalles |
|---|---|
| CAPTURE_ADMIN | Permite crear y gestionar privilegios de análisis de políticas. |
| DBFS_ROLE | Para el acceso a los objetos y paquetes del sistema de ficheros. |
| DELETE_CATALOG_ROLE | Se puede borrar registros en la tabla de auditoría del sistema (AUD$). |
| EXECUTE_CATALOG_ROLE | Se puede usar `EXECUTE` sobre los objetos del diccionario de datos. |
| EXP_FULL_DATABASE | Permite exportar copias completas e incrementales. |
| JAVA_ADMIN | Permite administrar las tablas de políticas para las aplicaciones Java. |
| OLAP_DBA | Permite administrar los objetos dimensionados para Oracle OLAP. |
| OLAP_USER | Permite desarrollar aplicaciones que crean objetos dimensionados en esquemas propios para Oracle OLAP. |

Para revocar los roles concedidos a un usuario se utiliza el comando `REVOKE` con la siguiente sintaxis:

```
REVOKE rol FROM usuario;
```

El comando siguiente:

```
REVOKE CAPTURE _ ADMIN, AUDIT VIEWER FROM Alejandro;
```

revocaría los roles a la cuenta de usuario *Alejandro* para gestionar privilegios de análisis de políticas, CAPTURE_ADMIN y para ver y analizar auditorías de datos, AUDIT VIEWER.

### 4.6.3. Gestión de privilegios sobre objetos

Los privilegios sobre los objetos determinan cómo un usuario puede tratar los objetos de los que no es propietario. Por ejemplo, un usuario que trabaja en el departamento de ventas no debería poder tocar el campo Stock de la tabla Producto, pero sin embargo sí puede hacer consultas. Por otro lado, algunos usuarios del departamento de incidencias sí pueden modificar los valores de la tabla Precios.

Los roles otorgan permisos a los usuarios para operar sobre los objetos de su propiedad, no obstante, los privilegios permiten definir el modo en el que otros usuarios van a tratar los objetos de otros usuarios.

Igual que en el caso de roles el comando que se usa es `GRANT`, pero con una sintaxis diferente:

```
GRANT {ALL [PRIVILEGES] | privilegio} ON objeto TO usuario [WITH GRANT OPTION];
```

La opción WITH GRANT OPTION hace lo mismo que lo que se explicó cuando se conceden roles, es decir, el usuario puede asignar el mismo privilegio a otro usuario.

El privilegio sobre un objeto lo otorga el propietario del objeto, es decir, aquel que lo creó. También lo puede otorgar un usuario con el rol DBA.

Para otorgar al usuario David que pueda insertar registros en la tabla Producto, se ejecutará el siguiente comando:

```
GRANT INSERT ON Producto TO David;
```

Si se quiere asignar todos los privilegios posibles que se pueden asignar a un objeto dependiendo de su tipo, se usa la opción ALL [PRIVILEGES]. La palabra PRIVILEGES es opcional.

Como ya se dijo anteriormente, se pueden asignar privilegios a algunas columnas de algunas tablas. Por ejemplo, supóngase la tabla Precio con los campos CodProd, fecha, precioVenta, precioCoste, tipoIVA. Si se desea que el usuario Antonio no pueda ver la columna precioCoste, el comando sería el siguiente:

```
GRANT SELECT(CodProd, fecha, precioVenta, tipoIVA) ON Precio TO
Antonio;
```

Se ha asignado al usuario Antonio el privilegio de consulta, privilegio SELECT, a todas las columnas de la tabla Precio excepto la columna precioCoste.

La tarea para la gestión de los privilegios es fácil, solo basta conocer todos los posibles privilegios asignables a cada tipo de objeto.

Para revocar los privilegios sobre objetos se usa también el comando `REVOKE`, con la sintaxis

```
REVOKE {ALL [PRIVILEGES] | privilegio} ON objeto FROM usuario [WITH GRANT OPTION];
```

Para revocar al usuario David los privilegios de consultas e inserción sobre la tabla Productos se ejecutaría el comando

```
REVOKE SELECT, INSERT ON Productos FROM David;
```

Para quitar todos los permisos al usuario Paco se ejecutaría el comando

```
REVOKE ALL PRIVILEGES FROM Paco;
```

Para quitar todos los permisos a Antonio sobre las tablas Venta y Compra se usaría el comando

```
REVOKE ALL ON Venta, Compra FROM Antonio;
```

## 4.7. Lenguaje de control de transacciones

Las transacciones suponen una acción atómica que afecta a las bases de datos. Estas acciones paralizan el acceso a los datos que participan en la transacción, para así modificar la instancia de la base de datos y pasar a la nueva. Esto conlleva que los datos que afectan a este proceso no estén disponibles para ningún usuario mientras dura el proceso. Además, estos procesos deben aplicar eficientes técnicas tolerantes a fallos. A nadie le gustaría que el importe del valor de su cuenta bancaria se viera reducido por una operación errónea.

Estas transacciones deben cumplir una serie de propiedades que se especifican en la siguiente lista:

- **Aislamiento:** el resultado de una transacción no es visto por otras transacciones hasta que esta termina en su totalidad.
- **Atomicidad:** todas las acciones de una transacción o se hacen correctamente, sin ningún error, o no se hacen.
- **Consistencia:** cuando las acciones asociadas a una transacción comienzan se tiene la certeza de que, antes de su inicio, la base de datos estaba en un estado consistente. Cuando finaliza el estado nuevo también debe ser totalmente consistente.
- **Durabilidad:** a partir de que una transacción finaliza con éxito, no se puede volver al estado anterior, es decir, el nuevo estado continua en el tiempo hasta que se ejecuta una nueva transacción.

El lenguaje de control de transacciones está compuesto por un conjunto de comandos que permite la gestión de las transacciones en un sistema gestor de bases de datos.

Cuando se ejecutan comandos del DML, por ejemplo, inserción, modificación y eliminación de registros, estas acciones no se ejecutan realmente en la colección de datos, sino que se apuntan en un cuaderno de bitácoras. Estas acciones pendientes comportan un conjunto de transacciones pendientes de llevar a cabo. El usuario que ejecuta las acciones tiene una visión del nuevo estado como si se hubiera ejecutado realmente en la base de datos. Esto permite volver a estados anteriores si no se ha ejecutado la transacción pendiente.

Estos comandos son `COMMIT`, `ROLLBACK`, `SAVEPOINT`, `ROLLBACK TO SAVEPOINT` y `SET TRANSACTION`.

Con `COMMIT` se llevan a cabo todas las acciones desde el último estado COMMIT ejecutado. Para deshacer todas las acciones y que la base de datos vuelva al estado en el que se encontraba en el último COMMIT, se usa `ROLLBACK`. Se puede establecer un punto intermedio en la transacción a través del comando `SAVEPOINT`. Para volver a él se usa `RALLBACK TO SAVEPOINT`, pero la transacción no se lleva a cabo, de tal modo que si se avanza y se hace un RALLBACK, se vuelve al punto donde se hizo el último COMMIT. A través de `SET TRANSACTION` se puede parametrizar la configuración de deshacer transiciones.

Todos los comandos del lenguaje DDL, por ejemplo, `CREATE`, `ALTER`, `DROP` o `RENAME`, se ejecutan implícitamente con un COMMIT.

## 4.8. Gestión del almacenamiento en Oracle

Los **espaclos de tablas** permiten el almacenamiento y administración de objetos en grupos individuales. Suponen una división lógica del conjunto de tablas, columnas, restricciones, secuencias, etc., para hacer más fácil la gestión de los archivos que almacenan los objetos asociados a un espacio de tabla. Por defecto, una base de datos tiene el espacio de tabla en el que trabaja cada usuario. Cuando se crea un usuario se debe indicar qué espacio de tabla usará.

Cuando se crea una tabla, esta se almacena en el **espaclo de tabla por defecto** del usuario que la crea. De este modo, si al conectarse a una base de datos se autentica como sys as sysdba, el espacio de tabla por defecto que usa este usuario es SYSTEM. Este espacio de tabla está dirigido al almacenamiento del diccionario de datos, propiedad del administrador. En el espacio de tabla SYSTEM no se debería crear ningún objeto, ya que este está implementado para contener el diccionario de datos.

Para indicar el espacio de tabla en el que se crea una tabla se especifica este después del paréntesis de cierre del comando CREATE TABLE. Por ejemplo, el siguiente comando crea la tabla Ciclo en el espacio de tabla Administracion. Evidentemente, el espacio de tablas debe estar creado y el usuario que crea un objeto en él debe estar autorizado para ello.

```
CREATE TABLE Ciclo(
        CodCF NUMBER(2) PRIMARY KEY,
        Nombre VARCHAR2(60) NOT NULL) TABLESPACE Administracion;
```

Este comando ha creado la tabla Ciclo en el espacio de tabla Administracion. Un espacio de tabla no supone una división de la base de datos, es decir, la tabla Ciclo no se puede crear de nuevo en otro espacio de tabla, solo una para toda la base de datos.

Lo más importante que se debe tener en cuenta ahora mismo sobre la gestión en espacios de tablas son los archivos de datos físicos asociados a cada uno de ellos.

Una base de datos tiene siempre, como mínimo, dos tablespaces llamados SYSTEM y SYSAUX (el tablespace SYStem AUXiliar). El tablespace SYSTEM contiene el diccionario de datos. El tablespace SYSAUX contiene los datos de ciertos componentes de Oracle. Estos tablespaces no deben contener datos de usuarios. Un tablespace puede estar accesible ONLINE, o no, OFFLINE, siendo esto un modo de hacer que ciertos datos de la aplicación sean temporalmente inaccesibles, o si la base de datos contiene varias aplicaciones, hacer que una aplicación sea inaccesible sin tocar a ninguna otra.

El tablespace SYSTEM debe estar siempre ONLINE. Un tablespace puede ser READ WRITE o READ ONLY, siendo esta opción última una forma sencilla de garantizar que los datos que contiene nunca serán modificados.

### 4.8.1. Comando CREATE TABLESPACE

La sintaxis del comando CREATE TABLESPACE es:

```
CREATE TABLESPACE nombreEspacioTabla
DATAFILE archivosDatos SIZE tamanio
DEFAULT STORAGE clausulaStorage
ONLINE|OFFLINE
```

La cláusula **DATAFILE** es la que permite declarar cada uno de los ficheros físicos asociados a un espacio de tablas. A un espacio de tablas se le puede agregar más archivos de datos conforme sea necesario. El uso de estos ficheros por parte de Oracle y de la administración de la base de datos es una tarea fundamental para estos últimos, con el fin de gestionar el espacio físico real del que se dispone. Una incorrecta configuración sobre alguna tabla consumiría mucho espacio, provocando incorrectos almacenamientos en otras tablas.

La sintaxis de esta cláusula es `'nombre_fichero [SIZE valor [K|M|G|T]] [REUSE] [cláusula_ autoextension_automatica]`, y se pueden indicar tantos ficheros separados por coma como se quiera. El valor de SIZE indica el tamaño del fichero que crear en disco. REUSE permite usar un fichero que ya está creado, vaciando su contenido y dejándolo listo con el tamaño que se indica con SIZE. Estas son las opciones más usuales:

- **Nombre_fichero:** la ruta completa al fichero de datos, normalmente en un repositorio de datos (oradata).
- **SIZE:** indica el tamaño inicial del fichero; se puede omitir si se usa REUSE y el fichero ya existe.
- **REUSE:** si el fichero existe, se borra y se reutiliza; si no se indica REUSE y el fichero existe, se obtendrá un mensaje de error y no se creará el espacio de tablas. Se recomienda no usar REUSE.

- **Clausula_autoextension_automatica:** AUTOEXTEND OFF|AUTOEXTEND ON [NEXT tamaño] [MAXSIZE {UNLIMITED|tamaño}]. AUTOEXTEND indica si el fichero de datos puede o no crecer una vez que se ha usado todo el espacio inicialmente asignado. NEXT indica el espacio mínimo asignado al fichero en el momento de la extensión.

Cuando Oracle crea un fichero de un tamaño, esto se hace para reservar del sistema de almacenamiento ese espacio para todos los objetos que se van a crear en ese espacio de tabla.

El siguiente comando crea un espacio de tabla con un archivo de datos de 500 MB llamado data01.dbf en el directorio d:\oradata\postigo. Se ha indicado una autoextensión de 100 MB cuando esos 500 MB se han llenado, aumentado de 100 en 100 hasta un máximo de 800 MB.

```
CREATE TABLESPACE datos
DATAFILE 'd:\oradata\postigo\data01.dbf' SIZE 500M
AUTOEXTEND ON NEXT 100M MAXSIZE 800M;
```

## 4.8.2. Cláusula de almacenamiento

La cláusula de almacenamiento hace referencia a cómo se crea cualquier objeto en un espacio de tabla. Si al crear una tabla no se indica su cláusula de almacenamiento, será almacenado con la que se ha indicado en **DEFAULT STORAGE** en el momento en el que se creó el espacio de tablas. La sintaxis de DEFAULT STORAGE tiene las siguientes opciones:

- **INITIAL tamaño:** tamaño inicial del segmento.
- **NEXT tamaño:** tamaño del segundo segmento.
- **MINEXTENTS tamaño:** número mínimo de segmentos.
- **MAXEXTENTS tamaño:** número máximo de segmentos.
- **PCTINCREASE porcentaje:** tamaño en porcentaje de los siguientes segmentos atendiendo al segundo segmento.

En el siguiente ejemplo se crea un espacio de tablas llamado TablasCrecimientoRapido con dos archivos de datos de 500 MB y 250 MB, respectivamente. Se indica con la cláusula DEFAULT STORAGE el almacenamiento por defecto que tendrán los objetos creados en el espacio de tabla.

```
CREATE TABLESPACE TablasCrecimientoRapido
DATAFILE 'C:\oracle\tablasRap01.dbf' SIZE 500M, 'C:\oracle\tablasRap02.dbf' SIZE 250M
DEFAULT STORAGE (INITIAL 15M NEXT 50M PCTINCREASE 40);
```

Al crear la siguiente tabla

```
CREATE TABLE Ciclo(
        CodCF NUMBER(2) PRIMARY KEY,
        Nombre VARCHAR2(50)
);
```

se crea en el espacio de tablas por defecto que tenga el usuario que ejecuta el comando. Si se hubiera escrito

```
CREATE TABLE Ciclo(
        CodCF NUMBER(2) PRIMARY KEY,
        Nombre VARCHAR2(50)
) TABLESPACE TablasCrecimientoRapido;
```

la tabla se hubiera creado en el espacio de tabla TablasCrecimientoRapido y se hubiera escrito en el primer espacio libre del fichero c:\oradata\tablasRap01.dbf. El espacio que hubiera tomado en dicho archivo es de 15 MB, que corresponde con la opción INITIAL de la cláusula DEFAULT STORAGE. Cuando esta tabla se llene, los 15 MB tomarán del archivo 50 MB para ella. Esta es la opción NEXT de la misma cláusula. Una vez llenos estos 50 MB extras, se cogerá una extensión del 40 % de los 50 MB, es decir, 20 MB. La siguiente extensión será del 40 % de estos 20 MB, es decir, 8 MB, y así sucesivamente. Esta es la opción PCTINCREASE de la misma cláusula.

Esto sucede con cualquier objeto que se crea en este espacio de tablas. Si un objeto no debe crearse con este almacenamiento por defecto, se puede indicar la cláusula STORAGE en la creación de la tabla. El siguiente ejemplo aclara este punto:

```
CREATE TABLE Director(
        Dni NUMBER(8),
        Nombre VARCHAR2(30) NOT NULL,
        PrApellido VARCHAR2(30) NOT NULL,
        SgApellido VARCHAR2(20),
        Domicilio VARCHAR2(50) NOT NULL,
        Telefono NUMBER(9) NOT NULL,
        Email VARCHAR2(320),
        CONSTRAINT pk_director PRIMARY KEY(dni)
) TABLESPACE TablasCrecimientoRapido STORAGE(INITIAL 5M NEXT 20M PCTINCREASE 50);
```

Como se ha declarado una gestión de almacenamiento a través de la cláusula STORAGE en el momento de crear la tabla, esta gestión no es la de por defecto, sino que esta tabla, que se crea en el espacio de tabla TablasCrecimientoRapido, se inicia con un tamaño de 5 MB, crece 20 MB cuando sea necesario y las sucesivas extensiones son del 50 % de la extensión anterior.

### 4.8.3. Gestión de espacios de tablas

Cuando un usuario crea un segmento sin precisar, el tablespace Oracle almacena el segmento en el tablespace por defecto del usuario. Esta se define con la cláusula DEFAULT TABLESPACE en `CREATE USER` y `ALTER USER`. Si no se indica, se creará en SYSTEM. El tablespace por defecto se define usando DEFAULT TABLESPACE de `CREATE DATABASE`.

## Modificar un espacio de tabla

Para modificar un tablespace permanente se puede usar `ALTER TABLESPACE` o `ALTER DATABASE`. Para renombrar un tablespace se usa el comando `ALTER TABLESPACE nombre_antiguo RENAME TO nombre_nuevo;`

Para añadir un fichero de datos a un espacio de tabla se emplea el comando `ALTER TABLESPACE nombre ADD DATAFILE especificación_datafile`. Por ejemplo,

```
ALTER TABLESPACE Datos ADD DATAFILE 'd:\oradata\postigo\datos02.dbf' SIZE 100M AUTOEXTEND ON NEXT 100M MAXSIZE 500M;
```

Para modificar el tamaño de un fichero de datos se utiliza la cláusula RESIZE en ADD DATAFILE. Por ejemplo,

```
ALTER TABLESPACE Datos ADD DATAFILE 'd:\oradata\postigo\datos02.dbf' RESIZE 100M;
```

Para modificar el crecimiento automático de un fichero de datos se usa AUTOEXTEND en la cláusula DATAFILE. Por ejemplo,

```
ALTER TABLESPACE Datos DATAFILE 'd:\oradata\postigo\datos02.dbf' SIZE 100M AUTOEXTEND ON NEXT 150M MAXSIZE 900M;
```

Para posibles tareas de administración, un espacio de tabla puede ser cambiado de modo activo a modo no activo con el comando ALTER TABLESPACE nombreTablespace OFFLINE. Para activarlo de nuevo, se usa la opción ONLINE.

## Eliminar un fichero de datos

Para eliminar un fichero de datos se usa ALTER TABLESPACE nombre DROP DATAFILE nombre_fichero. Por ejemplo,

```
ALTER TABLESPACE Datos DROP DATAFILE 'd:\oradata\postigo\datos02.dbf';
```

## Eliminar un espacio de tablas

Para la eliminación de un tablespace permanente se usa el comando DROP TABLE con sintaxis

```
DROP TABLESPACE nombre [INCLUDING CONTENTS [AND DATAFILES] [CASCADE CONSTRAINTS]];
```

Por ejemplo,

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

Las opciones son:

- **INCLUDING CONTENTS:** es necesario que el tablespace no esté vacío.
- **AND DATAFILES:** elimina, además, los archivos de datos.
- **CASCADE CONSTRAINTS:** elimina las restricciones de integridad referencial definidas en las tablas fuera del espacio de tablas.