Resumen DDL

Data Definition Language (**Lenguaje de definición de datos**).

Permite crear, borrar y modificar la estructura de las tablas de la base de datos. Lo forman las instrucciones **CREATE**, **ALTER**, **DROP**, **RENAME** y **TRUNCATE**.

Esquemas de usuario y objetos

Cada usuario de una base de datos posee un **esquema**. El esquema tiene el mismo nombre que el usuario y sirve para almacenar los objetos de esquema, es decir los objetos que posee el usuario.

Esos objetos pueden ser: tablas, vistas, secuencias, índices, sinónimos e instantáneas. Esos objetos son manipulados y creados por los usuarios. En principio sólo los administradores y los usuarios propietarios pueden acceder a cada objeto, salvo que se modifiquen los privilegios del objeto para permitir su acceso por parte de otros usuarios.

Creación de tablas

El nombre de una tabla debe cumplir las siguientes reglas:

- Comenzar con una letra
- No debe tener más de 30 caracteres
- Sólo se permiten utilizar letras del alfabeto (inglés), números o el signo de subrayado (también el signo $ y #, pero esos se utilizan de manera especial por lo que no son recomendados)
- No puede haber dos tablas con el mismo nombre para el mismo usuario (pueden coincidir los nombres si están en distintos esquemas)
- No puede coincidir con el nombre de una palabra reservada de Word

orden CREATE TABLE

Es la orden SQL que permite crear una tabla. Por defecto será almacenada en el tablespace por defecto del usuario que crea la tabla.

Sintaxis:

```
CREATE TABLE [esquema.] nombreDeTabla
   (nombreDeLaColumna1 tipoDeDatos [, ...]);
```

Ejemplo:

```
CREATE TABLE proveedores (nombre varchar2(25));
```

Crea una tabla con un solo campo de tipo **varchar2.**

Sólo se podrá crear la tabla si el usuario posee los permisos necesarios para ello. Si la tabla pertenece a otro esquema (suponiendo que el usuario tenga permiso para grabar tablas en ese otro esquema), se antepone al nombre de la tabla , el nombre del esquema:

```
CREATE TABLE otroUsuario.proveedores (nombre varchar2(25));
```

Borrar tablas

La orden **DROP TABLE** seguida del nombre de una tabla, permite eliminar la tabla en cuestión.

Al borrar una tabla:

- Desaparecen todos los datos
- Sólo es posible realizar esta operación si se es el propietario de la tabla o se posee el privilegio DROP ANY TABLE

El borrado de una tabla es irreversible, y no hay ninguna petición de confirmación, por lo que conviene ser muy cuidadoso con esta operación.

Alguno tipos de datos

Textos

Para los textos disponemos de los siguientes tipos:

- **VARCHAR2.** Para textos de longitud variable de hasta 4000 caracteres
- **CHAR.** Para textos de longitud fija de hasta 2000 caracteres.
- **NCHAR.** Para el almacenamiento de caracteres nacionales de texto fijo
- **NVARCHAR2.** Para el almacenamiento de caracteres nacionales de longitud variable.

En todos estos tipos se indican los tamaños entre paréntesis tras el nombre del tipo. Ese tamaño en el caso de los tipos VARCHAR2 es obligatorio, en el caso de los tipos CHAR son opcionales (de no ponerlos se toma el uno).

Números

El tipo NUMBER es un formato versátil que permite representar todo tipo de números. Su rango recoge números de entre 10^{-130} y $9{,}99999999999 * 10^{128}$. Fuera de estos rangos Oracle devuelve un error.

Los números decimales (números de coma fija) se indican con **NUMBER(*p*,*s*)**, donde *p* es la precisión máxima y *s* es la escala (número de decimales a la derecha de la coma). Por ejemplo, NUMBER (8,3) indica que se representan números de ocho cifras de precisión y tres decimales. Los decimales en Oracle se presenta con el **punto y no con la coma.**

Para números enteros se indica **NUMBER(p)** donde *p* es el número de dígitos. Eso es equivalente a NUMBER(*p*,0).

Para números de coma flotante (equivalentes a los **flota** o **double** de muchos lenguajes de programación) simplemente se indica el texto **NUMBER** sin precisión ni escala.

Fechas y horas

DATE

El tipo **DATE** permite almacenar fechas. Las fechas se pueden escribir en formato día, mes y año entre comillas. El separador puede ser una barra de dividir, un guión y casi cualquier símbolo.

Para almacenar la fecha actual basta con utilizar la función **SYSDATE** que devuelve esa fecha.

Modificar tablas

Cambiar de nombre

La orden **RENAME** permite el cambio de nombre de cualquier objeto. Sintaxis:

```
RENAME nombreViejo TO nombreNuevo
```

Borrar contenido de tablas

La orden **TRUNCATE TABLE** seguida del nombre de una tabla, hace que se elimine el contenido de la tabla, pero no la tabla en sí. Incluso borra del archivo de datos el espacio ocupado por la tabla.

Esta orden no puede anularse con un ROLLBACK.

Modificar tablas

La versátil **ALTER TABLE** permite hacer cambios en la estructura de una tabla.

Añadir columnas

```
ALTER TABLE   nombreTabla ADD(nombreColumna TipoDatos
[Propiedades]
         [,columnaSiguiente tipoDatos [propiedades]...)
```

Permite añadir nuevas columnas a la tabla. Se deben indicar su tipo de datos y sus propiedades si es necesario (al estilo de CREATE TABLE).

Las nuevas columnas se añaden al final, no se puede indicar otra posición.

Borrar columnas

```
ALTER TABLE nombreTabla DROP column columna;
```

Elimina la columna indicada de manera irreversible e incluyendo los datos que contenía. No se puede eliminar la última columna (habrá que usar DROP TABLE).

Modificar columna

Permite cambiar el tipo de datos y propiedades de una determinada columna. Sintaxis:

```
ALTER TABLE nombreTabla MODIFY(columna tipo [propiedades]
   [columnaSiguiente tipo [propiedades] ...]
```

Los cambios que se permiten son:

- Incrementar precisión o anchura de los tipos de datos
- Sólo se puede reducir la anchura si la anchura máxima de un campo si esa columna posee nulos en todos los registros, o todos los valores so o no hay registros
- Se puede pasar de CHAR a VARCHAR2 y viceversa (si no se modifica la anchura)

Valor por defecto

A cada columna se le puede asignar un valor por defecto durante su creación mediante la propiedad DEFAULT. Se puede poner esta propiedad durante la creación o modificación de la tabla, añadiendo la palabra DEFAULT tras el tipo de datos del campo y colocando detrás el valor que se desea por defecto.

Ejemplo:

```
CREATE TABLE articulo (cod NUMBER(7), nombre VARCHAR2(25),
  precio NUMBER(11,2) DEFAULT 3.5);
```

Restricciones

Una restricción es una condición de obligado cumplimiento para una o más columnas de la tabla. A cada restricción se le pone un nombre, en el caso de no poner un nombre (en las que eso sea posible) entonces el propio Oracle le coloca el nombre que es un mnemotécnico con el nombre de tabla, columna y tipo de restricción.
Se pueden especificar al crear la tabla, o una vez creada se pueden añadir o borrar

Su sintaxis general es:

```
CREATE TABLE nombreTabla |
  (campo tipo [propiedades] [,...]
 CONSTRAINT nombreRestricción tipoRestricción (columnas)
 [,CONSTRAINT nombrerestricción tipoRestricción (columnas) ...)
```

```
ALTER TABLE nombreTabla ADD CONSTRAINT nombreRestricción
           tipoRestricción (columnas)
```

```
ALTER TABLE nombreTabla DROP CONSTRAINT nombreRestricción
```

Las restricciones tienen un nombre, se puede hacer que sea Oracle el que les ponga nombre, pero entonces será críptico. Por eso es mejor ponerle uno mismo.

Los nombres de restricción no se pueden repetir para el mismo esquema, por lo que es buena idea incluir de algún modo el nombre de la tabla, los campos involucrados y el tipo de restricción en el nombre de la misma. Por ejemplo *pieza_id_pk* podría indicar que el campo *id* de la tabla *pieza* tiene una clave principal (**P**RIMARY **K**EY).

Prohibir nulos

La restricción NOT NULL permite prohibir los nulos en una determinada tabla. Eso obliga a que la columna tenga que tener obligatoriamente un valor para que sea almacenado el registro.

Se puede colocar durante la creación (o modificación) del campo añadiendo la palabra NOT NULL tras el tipo:

```
CREATE TABLE cliente(dni VARCHAR2(9) NOT NULL);
```

En ese caso el nombre le coloca Oracle. La otra forma (que admite nombre) es:

```
CREATE TABLE cliente(dni VARCHAR2(9)
 CONSTRAINT dni_sinnulos NOT NULL(dni));
```

Valores únicos

Las restricciones de tipo UNIQUE obligan a que el contenido de uno o más campos no puedan repetir valores. Nuevamente hay dos formas de colocar esta restricción:

```
CREATE TABLE cliente(dni VARCHAR2(9) UNIQUE);
```

En ese caso el nombre de la restricción la coloca el sistema Oracle. Otra forma es:

```
CREATE TABLE cliente(dni VARCHAR2(9) CONSTRAINT dni_u UNIQUE);
```

Esta forma permite poner un nombre a la restricción. Si la repetición de valores se refiere a varios campos, la forma sería:

```
CREATE TABLE alquiler(dni VARCHAR2(9),
        cod_pelicula NUMBER(5),
        CONSTRAINT alquiler_uk UNIQUE(dni,cod_pelicula) ;
```

La coma tras la definición del campo *cod_pelicula* hace que la restricción sea independiente de ese campo. Eso obliga a que, tras UNIQUE se indique la lista de campos.

Los campos UNIQUE son las claves candidatas de la tabla (que habrán sido detectadas en la fase de diseño de la base de datos).

Clave primaria

La clave primaria de una tabla la forman las columnas que indican a cada registro de la misma. La clave primaria hace que los campos que la forman sean NOT NULL (sin posibilidad de quedar vacíos) y que los valores de los campos sean de tipo UNIQUE (sin posibilidad de repetición).

Si la clave está formada por un solo campo basta con:

```
CREATE TABLE cliente(
 dni VARCHAR2(9) PRIMARY KEY,
 nombre VARCHAR(50)) ;
```

O, poniendo un nombre a la restricción:

```
CREATE TABLE cliente(
 dni VARCHAR2(9) CONSTRAINT cliente_pk PRIMARY KEY,
 nombre VARCHAR(50)) ;
```

Si la clave la forman más de un campo:

```
CREATE TABLE alquiler(dni VARCHAR2(9),
        cod_pelicula NUMBER(5),
        CONSTRAINT alquiler_pk PRIMARY KEY(dni,cod_pelicula) ;
```

Clave ajena o foránea

Una clave ajena o foránea, es uno o más campos de una tabla que están relacionados con la clave principal de los campos de otra tabla. La forma de indicar una clave foránea es:

```
CREATE TABLE alquiler(dni VARCHAR2(9),
        cod_pelicula NUMBER(5),
        CONSTRAINT alquiler_pk PRIMARY KEY(dni,cod_pelicula),
        CONSTRAINT dni_fk FOREIGN KEY (dni)
              REFERENCES clientes(dni),
        CONSTRAINT pelicula_fk FOREIGN KEY (cod_pelicula)
              REFERENCES peliculas(cod)
        );
```

Esta completa forma de crear la tabla alquiler incluye sus claves foráneas, el campo *dni* hace referencia al campo dni de la tabla clientes y el campo *cod_pelicula* que hace referencia al campo *cod* de la tabla *peliculas*. También hubiera bastado con indicar sólo la tabla a la que hacemos referencia, si no se indican los campos relacionados de esa tabla, se toma su clave principal (que es lo normal).

Esto forma una relación entre dichas tablas, que además obliga al cumplimiento de la **integridad referencial.** Esta integridad obliga a que cualquier *dni* incluido en la tabla *alquiler* tenga que estar obligatoriamente en la tabla de clientes. De no ser así el registro no será insertado en la tabla (ocurrirá un error).

Otra forma de crear claves foráneas (sólo válida para claves de un solo campo) es:

```
CREATE TABLE alquiler(
   dni VARCHAR2(9) CONSTRAINT dni_fk   REFERENCES clientes(dni),
   cod_pelicula NUMBER(5) CONSTRAINT pelicula_fk
            REFERENCES peliculas(cod)
);
```

Esta definición de clave foránea es idéntica a la anterior, sólo que no hace falta colocar el texto FOREIGN KEY.

La integridad referencial es una herramienta imprescindible de las bases de datos relacionales.

Pero provoca varios problemas. Por ejemplo, si borramos un registro en la tabla principal que está relacionado con uno o varios de la secundaria ocurrirá un error, ya que de permitírsenos borrar el registro ocurrirá fallo de integridad (habrá claves secundarios refiriéndose a una clave principal que ya no existe).

Por ello Oracle nos ofrece dos soluciones a añadir tras la cláusula REFERENCES:

- **ON DELETE SET NULL.** Coloca nulos todas las claves ajenas relacionadas con la borrada.
- **ON DELETE CASCADE.** Borra todos los registros cuya clave ajena es igual que la clave del registro borrado.

Si no se indica esta cláusula, no se permite el borrado de registros relacionados.

El otro problema ocurre si se desea cambiar el valor de la clave principal en un registro relacionado con claves secundarias. En muchas bases de datos se implementan soluciones consistentes en añadir ON UPDATE CASCADE o ON UPDATE SET NULL. Oracle no implementa directamente estas soluciones. Por lo que habrá que hacerlo de otra forma

La sintaxis completa para añadir claves foráneas es:

```
CREATE TABLE tabla(lista_de_campos,
        CONSTRAINT nombreRestriccion FOREIGN KEY (listaCampos)
              REFERENCES tabla(clavePrincipalRelacionada)
              [ON DELETE {SET NULL | CASCADE}]
);
```

Si es de un solo campo existe esta alternativa:

```
CREATE TABLE tabla(lista_de_campos tipos propiedades,
        nombreCampoClaveSecundaria
              CONSTRAINT nombreRestriccion
                    REFERENCES tabla(clavePrincipalRelacionada)
                    [ON DELETE {SET NULL | CASCADE}]
);
```

Restricciones de validación

Son restricciones que dictan una condición que deben cumplir los contenidos de una columna. La expresión de la condición es cualquier expresión que devuelva verdadero o falso, pero si cumple estas premisas:

- No puede hacer referencia a números de fila
- No puede hacer referencia a objetos de SYSTEM o SYS
- No se permiten usar las funciones SYSDATE, UID, USER y USERENV
- No se permiten referencias a columnas de otras tablas (sí a las de la misma tabla)

Una misma columna puede tener múltiples CHECKS en su definición (se pondrían varios CONSTRAINT seguidos, sin comas). Ejemplo:

```
CREATE TABLE ingresos(cod NUMBER(5) PRIMARY KEY,
   concepto VARCHAR2(40) NOT NULL,
      importe NUMBER(11,2) CONSTRAINT importe_min
                CHECK (importe>0)
          CONSTRAINT importe_max
                 CHECK (importe<8000)
);
```

Para poder hacer referencia a otras columnas hay que construir la restricción de forma independiente a la columna:

```
CREATE TABLE ingresos(cod NUMBER(5) PRIMARY KEY,
  concepto VARCHAR2(40) NOT NULL,
  importe_max NUMBER(11,2),
  importe  NUMBER(11,2),
  CONSTRAINT importe_maximo CHECK (importe<importe_max)
);
```

Añadir restricciones

Es posible querer añadir restricciones tras haber creado la tabla. En ese caso se utiliza la siguiente sintaxis:

```
ALTER TABLE tabla
ADD [CONSTRAINT nombre] tipoDeRestricción(columnas);
```

tipoRestricción es el texto CHECK, PRIMARY KEY o FOREIGN KEY.
Las restricciones NOT NULL deben indicarse al modificar un campo, mediante
ALTER TABLE .. MODIFY nombrecampo NOT NULL

Borrar restricciones

Sintaxis:

```
ALTER TABLE tabla
   DROP PRIMARY KEY | UNIQUE(campos) |
   CONSTRAINT nombreRestricción [CASCADE]
```

La opción PRIMARY KEY elimina una clave principal (también quitará el índice UNIQUE sobre las campos que formaban la clave. UNIQUE elimina índices únicos. La opción CONSTRAINT elimina la

restricción indicada.

La opción CASCADE hace que se eliminen en cascada las restricciones de integridad que dependen de la restricción eliminada. Por ejemplo en:

```
CREATE TABLE curso(
 cod_curso CHAR(7) PRIMARY KEY,
 fecha_inicio DATE,
 fecha_fin DATE,
 tItulo VARCHAR2(60),
 cod_siguientecurso CHAR(7),
 CONSTRAINT fecha_ck CHECK(fecha_fin>fecha_inicio),
 CONSTRAINT cod_ste_fk FOREIGN KEY(cod_siguientecurso)
       REFERENCES curso ON DELETE SET NULL
 );
```

Tras esa definición de tabla, esta instrucción:

```
ALTER TABLE curso DROP PRIMARY KEY;
```

Produce este error:

```
ORA-02273: a esta clave única/primaria hacen referencia
 algunas claves ajenas
```

Para ello habría que utilizar esta instrucción:

```
ALTER TABLE curso DROP PRIMARY KEY CASCADE;
```

Esa instrucción elimina la clave ajena antes de eliminar la principal.

También produce error esta instrucción:

```
ALTER TABLE curso DROP(fecha_inicio);
ERROR en línea 1:
ORA-12991: se hace referencia a la columna en una restricción de
 multicolumna
```

El error se debe a que no es posible borrar una columna que forma parte de la definición de una instrucción. La solución es utilizar CASCADE CONSTRAINT elimina las restricciones en las que la columna a borrar estaba implicada:

```
ALTER TABLE curso DROP(fecha_inicio) CASCADE CONSTRAINTS;
```

Esta instrucción elimina la restricción de tipo CHECK en la que aparecía la *fecha_inicio* y así se puede eliminar la columna.

Desactivar restricciones

A veces conviene temporalmente desactivar una restricción para saltarse las reglas que impone. La sintaxis es:

```
ALTER TABLE tabla DISABLE CONSTRAINT nombre [CASCADE]
```

La opción CASCADE hace que se desactiven también las restricciones dependientes de la que se desactivó.

Activar restricciones

Anula la desactivación. Formato:

```
ALTER TABLE tabla ENABLE CONSTRAINT nombre [CASCADE]
```

Sólo se permite volver a activar si los valores de la tabla cumplen la restricción que se activa.

Si hubo desactivado en cascada, habrá que activar cada restricción individualmente.

Cambiar de nombre a las restricciones

Para hacerlo se utiliza este comando:

```
ALTER TABLE table RENAME CONSTRAINT
   nombreViejo TO nombreNuevo;
```