BASES DE DATOS

Grado en Ingeniería Informática

- Lenguaje de Definición de Datos

- Instrucciones de Actualización

- Diccionario de datos de Oracle

Curso 2019- 20

1 El lenguaje de definición de datos (LDD) de SQL

El lenguaje SQL puede utilizarse como DDL (*Data Definition Language*) o Lenguaje de Definición de Datos (LDD). El LDD permite:

- Definir y crear tablas
- Suprimir tablas
- Modificar la definición de las tablas
- Definir tablas virtuales (vistas) de datos
- Construir índices para hacer más rápido el acceso a las tablas

NOTA: la notación utilizada para la especificación de los comandos de SQL es la siguiente:

• las palabras clave en se indican en mayúsculas
• los corchetes **[ ]** indican opcionalidad
• las llaves **{ }** delimitan alternativas separadas por el símbolo **|**
• los puntos suspensivos (**...**) indica repetición

1.1 Sentencias sobre Tablas

Las tres operaciones que se pueden realizar sobre una tabla son: Crear, Eliminar y Modificar.

1.1.1 Crear una Tabla

Para la creación de tablas con SQL se utiliza el comando **CREATE TABLE**. Este comando tiene una sintaxis más compleja de la que aquí se expone, pero sólo vamos a estudiar las cláusulas necesarias para llevar a cabo el diseño de las tablas. La tabla tiene como propietario al usuario que las crea. Otro usuario que quiera utilizar nuestras tablas debe tener autorización y hacer referencia a la tabla como 'propietario.tabla'.

La sintaxis del comando es la siguiente:

```
CREATE TABLE nombre_tabla (
{ nombre_atributo tipo_dato [restricción_atributo] ...
| restricción_tabla} [,
{ nombre_atributo tipo_dato [restricción_atributo] ...
| restricción_tabla}] ...
);
```

Donde *restricción_atributo* tiene la sintaxis:

```
[CONSTRAINT nombre_restricción]
{ [NOT] NULL
| DEFAULT valor
| { UNIQUE | PRIMARY KEY }
```

```
| REFERENCES nombre_tabla [(nombre_atributo)] [ON DELETE CASCADE]
| CHECK (condición)
}
```

y *restricción_tabla* tiene la sintaxis:

```
[CONSTRAINT nombre_restricción]
{ {UNIQUE | PRIMARY KEY} (nombre_atributo [,nombre_atributo] ...)
| FOREIGN KEY (nombre_atributo [,nombre_atributo] ...)
REFERENCES nombre_tabla [(nombre_atributo [,nombre_atributo] ...)]
[ON DELETE CASCADE | SET NULL]
| CHECK (condición)
}
```

Ejemplo:

```
CREATE TABLE CURSO (
      nomCurso varchar2(20) NOT NULL,
      codCurso char(3),
      profesor char(9),
      maxAlum number(2, 0),
      fechaInic date,
      fechaFin date,
      horas number(3, 0) NOT NULL,
      CONSTRAINT cursosClave PRIMARY KEY (codCurso),
      CONSTRAINT cursosFechasValidas
        CHECK (fechaFin > fechaInic),
      CONSTRAINT cursosCodigosValidos
        CHECK (codCurso IN ('C01','C02','C03','C04','C05')),
      CONSTRAINT cursosAjena
        FOREIGN KEY (profesor) REFERENCES PROFESOR(dni),
      CONSTRAINT cursosNomUnico UNIQUE (nomCurso));
```

Restricciones de atributo

- **NOT NULL.** El atributo no permite valores nulos.
- **DEFAULT *valor***. Permite asignar un valor por defecto en el atributo. Oracle no permite asignar un nombre a esta restricción cuando se utiliza como restricción de atributo.
- **PRIMARY KEY.** Permite indicar que un atributo forma la clave primaria. Se utiliza cuando la clave primaria es simple.
- **REFERENCES tabla [(atr)].** Es la manera de indicar que un atributo es clave ajena o externa y hace referencia a la clave primaria de otra tabla. Opcionalmente se puede poner, entre paréntesis, el nombre del atributo que hace de clave principal.
- **UNIQUE.** Obliga a que los valores del atributo tomen valores únicos (no puede haber dos filas con igual valor en ese atributo). Se implementa creando un índice para dicho atributo.
- **CHECK (condición).** Establece una condición que deben cumplir los valores introducidos para un atributo.

La cláusula **CONSTRAINT** sirve para asignarle un nombre a una **restricción**. De esta forma, el diccionario de datos almacenará el nombre de la restricción y ésta podrá ser consultada o eliminada de forma independiente.

Restricciones de tablas

Sirven para definir restricciones conjuntas sobre una combinación de atributos. Se especifican después de haber descrito todos los campos de la tabla. Las restricciones más habituales son:

- **UNIQUE (atr1 [, atr2] ...)**: el(los) atributos(s) NO pueden contener valores duplicados conjuntamente.

- **PRIMARY KEY (atr1 [, atr2] ...)**: el(los) atributos(s) forman la clave primaria. Por lo tanto, tienen valor ÚNICO y NO NULO.

- **FOREIGN KEY (atr1 [, atr2] ...) REFERENCES tabla (atr1 [, atr2] ...)**: el (los) atributos(s) forman una clave ajena. La cláusula REFERENCES indica la tabla referenciada. Si los atributos clave de la tabla referenciada no tienen el mismo nombre que los atributos que forman la clave ajena, deben especificarse los nombres de los atributos clave.

- **CHECK (condición)**: establece una condición que deben cumplir los valores introducidos para un atributo o entre atributos.

La cláusula **REFERENCES** en las restricciones de atributo y la cláusula **FOREIGN KEY** en las restricciones de tabla tienen una opción optativa con el siguiente significado:

Cuando existen relaciones entre tablas, se puede incumplir una restricción de integridad referencial al eliminar tuplas que están siendo referenciadas. El diseñador del esquema puede especificar la acción que ha de emprenderse si se incumple una restricción de integridad referencial al eliminarse una tupla referenciada.

Existen, en general, tres posibles acciones a realizar:

- **RESTRICT**. Operación restringida. Es la opción por defecto
- **CASCADE**. Operación en cascada
- **SET NULL**. Poner a nulo

Ejemplos:

```
CREATE TABLE ASIGNATURA
(
      ...
CONSTRAINT asig-prof FOREIGN KEY (prof) REFERENCES PROFESOR (nPr)
ON DELETE SET NULL );
```

Esta sentencia indica que si eliminamos un profesor de la tabla PROFESOR, las tuplas que hacían referencia a dicho profesor en la tabla ASIGNATURA ponen un valor nulo en el atributo prof.

```
CREATE TABLE ASIGNATURA
(
      ...
CONSTRAINT asig-prof FOREIGN KEY (prof) REFERENCES PROFESOR (nPr)
ON DELETE CASCADE);
```

Esta sentencia indica que si eliminamos un profesor responsable de alguna asignatura, se eliminará la asignatura correspondiente en la tabla ASIGNATURA.

Tipos de datos

- Numéricos

SQL	Tipo	Oracle
INTEGER	Números enteros de distintos tamaños	NUMBER(38, 0)
INT		NUMBER(38, 0)
SMALLINT		NUMBER(38, 0)
FLOAT(*n*)	Números reales de distinta precisión	NUMBER
REAL		NUMBER
DOUBLE PRECISION		NUMBER
DECIMAL(*p*, *e*)	Número real con *p* dígitos y *e* decimales	NUMBER(*p*, *e*)
DEC(*p*, *e*)		NUMBER(*p*, *e*)
NUMERIC(*p*, *e*)		NUMBER(*p*, *e*)

- Cadenas de caracteres

SQL	Tipo	Oracle
CHAR(*n*)	Cadena de caracteres de longitud fija	CHAR(*n*)
CHARACTER(*n*)		CHAR(*n*)
VARCHAR(*n*)	Cadena de caracteres de longitud variable	VARCHAR2(*n*)

- Cadenas de bits

SQL	Tipo	Oracle
BIT(*n*)	Longitud fija con *n* bits	RAW(n)
BIT VARYING(*n*)	Longitud variable con *n* bits como máximo	LONG RAW

- Fecha y hora

SQL	Tipo	Oracle
DATE	Año, mes y día	DATE
TIMESTAMP	Fecha y hora	TIMESTAMP(*p*), siendo p el número de dígitos de la parte fraccionaria de los segundos. Por defecto, *p* = 6.

1.1.2 Eliminar una tabla

```
DROP TABLE nombre_tabla [CASCADE CONSTRAINT];
```

Si se utiliza la cláusula CASCADE CONSTRAINT se elimina la tabla y todas las restricciones asociadas. Si se omite esta cláusula, la tabla no se borra si existen restricciones sobre sus atributos en otras tablas y Oracle devuelve un mensaje de error.

1.1.3 Modificar una tabla

La definición de una tabla se puede modificar mediante el comando **ALTER TABLE** (modificar tabla). Las posibles acciones de modificar tablas incluyen la inserción o eliminación de una columna (atributo), la modificación de la definición de una columna y la inserción o eliminación de restricciones de tabla.

```
ALTER TABLE nombre_tabla
{ADD nuevo_nombre_atributo tipo [NOT NULL] [CONSTRAINT restricción]
  |  MODIFY nombre_atributo tipo_nuevo [CONSTRAINT restricción]
  |  DROP COLUMN nombre_atributo [CASCADE CONSTRAINT]
  |  DROP (nombre_atributo1, nombre_atributo2, …)
  |  ADD CONSTRAINT nombre_restricción restricción
  |  { DROP | ENABLE | DISABLE } CONSTRAINT nombre_restricción [CASCADE]
 };
```

La opción **ADD** permite añadir un nuevo atributo a la tabla, con un valor NULL inicial para todas las tuplas. **MODIFY** cambia la definición del atributo que ya existe en la tabla. Las restricciones a tener en cuenta son las siguientes:

- Se puede cambiar el tipo del atributo o disminuir su tamaño sólo si todas las tuplas tienen en ese atributo el valor NULL.
- Un atributo NOT NULL únicamente se puede añadir a una tabla sin tuplas.
- Un atributo ya existente sólo se puede hacer NOT NULL si todas las tuplas tienen en ese atributo un valor distinto de NULL.
- Cuando se modifica un atributo, todo lo no especificado en la modificación se mantiene tal y como estaba.

Ejemplo:

```
ALTER TABLE ASIGNATURA DROP CONSTRAINT cursos;
ALTER TABLE ASIGNATURA ADD CONSTRAINT cursos
            CHECK (curso IN ('1','2','3','4','5'));
```

1.2 Sentencias sobre Índices

1.2.1 Crear un índice

```
CREATE [UNIQUE] INDEX nombre_índice
ON nombre_tabla (nombre_atributo1 [, nombre_atributo2] ...);
```

UNIQUE. El índice no admite claves duplicadas.

1.2.2 Borrar un índice

```
DROP INDEX nombre_índice;
```

2 Instrucciones de actualización en SQL

Los tres comandos que nos ofrece SQL para actualizar las bases de datos son: INSERT, DELETE y UPDATE.

2.1 El comando INSERT

En su forma más simple, sirve para añadir una sola tupla a una tabla. Debemos especificar el nombre de la tabla y una lista de valores para la tupla. Los valores deberán listarse en el mismo orden en que se especificaron los atributos cuando se creó la tabla.

La sintaxis del comando es la siguiente:

```
INSERT INTO nombre_tabla
[(nombre_atributo [, nombre_atributo] ...)]
{VALUES (valor [, valor] ...)
};
```

Por ejemplo, para añadir una nueva tupla a la tabla ORDENADOR, podemos hacer:

```
INSERT INTO ORDENADOR
VALUES ('Ord220', 'Ordenador Multimedia', 'Aula 8');
```

Se puede especificar explícitamente cuáles son los campos a los que se le va a insertar un valor. A los atributos que no se indican se le asigna el valor NULL o su valor por defecto (si lo tiene). Por ejemplo, si queremos introducir un profesor, pero no conocemos todos sus datos, podemos realizar la siguiente instrucción:

```
INSERT INTO PROFESOR (nPr, dni, nombre)
VALUES ('30', '29.555.555', 'Antonio Díaz Sotelo');
```

Una variación de la instrucción INSERT inserta múltiples tuplas en una relación y la carga con el resultado de una consulta

Ejemplo:

Creación de la tabla:

```
CREATE TABLE NUM_MATRICULADOS (
   nombreAsig varchar2(40),
   numero number(4, 0));
```

Carga con el resultado de la consulta:

```
INSERT INTO NUM_MATRICULADOS
SELECT A.nombre, COUNT(*)
FROM ASIGNATURA A INNER JOIN MATRICULA M
  USING (idAsig)
GROUP BY A.nombre;
```

2.2 El comando DELETE

Se utiliza para eliminar tuplas de una tabla. Su sintaxis es la siguiente:

```
DELETE FROM nombre_tabla
[WHERE condición];
```

Está formada por una cláusula WHERE similar a la de las consultas SQL (se verá más adelante) para seleccionar las tuplas que se van a eliminar. La omisión de la cláusula WHERE indica que deben eliminarse todas las tuplas de la tabla.

Ejemplos:

1. Borra de la tabla ORDENADOR todos los ordenadores que sean estaciones SUN.

```
DELETE FROM ORDENADOR
WHERE tipo = 'Estación Sun';
```

2. Borra de la tabla MATRICULA todas las tuplas de las asignaturas de primero.

```
DELETE FROM MATRICULA
WHERE idAsig IN  ( SELECT idAsig
                   FROM ASIGNATURA
                   WHERE curso = 1);
```

2.3 El comando UPDATE

Sirve para modificar los valores de los atributos en una o más tuplas seleccionadas. La sintaxis del comando es la siguiente:

```
UPDATE nombre_tabla
SET {nombre_atr = valor [, nombre_atr = valor, ...]
| nombre_atr [, nombre_atr, ...] = (subconsulta)
}
[WHERE condición];
```

También utiliza una cláusula WHERE para seleccionar las tuplas que se van a modificar.

Ejemplos:

1. Modifica el cuatrimestre y la especialidad de la asignatura 'Análisis Numérico'.

```
UPDATE ASIGNATURA
SET cuat = 2, esp = 'S'
WHERE nombre = 'Análisis Numérico';
```

2. Sube un punto en la convocatoria de junio de 2001 a todos los alumnos en 'Bases de Datos I'

```
UPDATE MATRICULA M
SET feb_jun = feb_jun + 1
WHERE M.año = 2001  AND M.idAsig =
        (   SELECT A.idAsig
            FROM ASIGNATURA A
            WHERE A.nombre = 'Bases de Datos I');
```

3 El diccionario de datos de Oracle

El comando **`describe nombre_tabla`** permite ver la definición de una tabla concreta de la base de datos. En esta descripción aparecerán los nombres de los atributos, el tipo de datos de cada atributo y si tiene o no permitidos valores nulos.

Ejemplo: **`describe ALUMNO`**

Toda la información de las tablas creadas con el comando CREATE TABLE queda registrada en el diccionario de datos de Oracle. Este diccionario está formado por tablas y vistas que pueden ser consultadas por los usuarios.

Las vistas del diccionario de datos son de tres tipos:

- Las que comienzan con el prefijo **all**_ pueden ser consultadas por todos los usuarios y ofrecen información sobre todos los objetos del sistema.
- Las que comienzan con el prefijo **dba**_ sólo son accesibles para tareas de administración.
- Las que comienzan con el prefijo **user**_ ofrecen información de los objetos propios de un usuario.

Podemos usar el comando describe para conocer los nombres de los atributos de las tablas.

Ejemplo: **`describe user_tables`**

Algunas de las **Tablas del diccionario de datos** son

- **user_objects**

Contiene información sobre todos los objetos definidos por el usuario actual.

Ejemplo:

```
select object_name, last_ddl_time
from user_objects
where object_type='TABLE';
```

- **user_tables**

Contiene información sobre las tablas de un usuario. Podemos usar el sinónimo **tabs**.

Ejemplo:

```
select table_name
from user_tables;
```

- **user_tab_columns**

Contiene información sobre los atributos de una tabla. Podemos usar el sinónimo **cols**.

Ejemplo:

```
select column_name, data_type, data_length
from user_tab_columns
where table_name='ALUMNO';
```

- **user_constraints**

Contiene información sobre las restricciones definidas por el usuario.

Ejemplo:

```
select constraint_name, constraint_type, search_condition, status
from user_constraints
where table_name='ALUMNO';
```

- **user_cons_columns**

Contiene información de las restricciones definidas sobre cada atributo.

Ejemplo:

```
select constraint_name, column_name
from user_cons_columns
where table_name='ALUMNO';
```

- **user_triggers**

Contiene información sobre los disparadores creados por el usuario.

Ejemplo:

```
select trigger_name
from user_triggers
where table_name='ALUMNO';
```

- **user_procedures**

Contiene información sobre los procedimientos y funciones, creados por el usuario, almacenados en la base de datos.

Ejemplo:

```
select *
from user_procedures;
```

- **user_source**

Contiene información sobre los bloques de código (procedimientos, funciones y disparadores) creados por el usuario.

Ejemplo:

```
select distinct name, type
from user_source;
```