DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Escuela Técnica Superior de Ingeniería Informática

etsinf

Bases de Datos y Sistemas de Información

Grado en Ingeniería Informática

Unidad Didáctica 2: El lenguaje SQL: definición de datos

Parte 3: LDD: Lenguaje de Definición de Datos

(Doc. UD2.3)

Curso 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Índice

1 LENGUAJE DE DEFINICIÓN DE DATOS (LDD)

Este subconjunto de SQL permite crear, modificar y eliminar componentes de bases de datos. Se presentarán en este documento una parte de las instrucciones del LDD que están disponibles en los sistemas comerciales concretamente en ORACLE que es el SGBD donde se van a realizar las prácticas de SQL en esta asignatura.

De nuevo, para poder describir la sintaxis de este lenguaje se utiliza una notación especial basada en la notación BNF. La sintaxis informa sobre cuál es el orden de las distintas partes o cláusulas de la instrucción, además de indicar cuáles son obligatorias y cuáles opcionales, y qué elementos se pueden incluir en cada caso.

La notación, aunque ya se incluyó en el documento UD2.1, se recuerda a continuación:

Cursiva	: se usa para nombres de componentes de la base de datos.
MAYÚSCULAS	: se escriben así las palabras reservadas de SQL.
Texto normal	: se usa para los elementos a definir más adelante.
$E_1, E_2,..., E_n$	: lista de elementos E_i separados por comas, donde i>0.
|	: separa opciones alternativas.
[ ]	: indica contenido opcional.
{ }	: indica contenido obligatorio.

Cuando un contenido opcional incluye varias alternativas a elegir (separadas por la barra vertical), siempre debe estar definida una de esas alternativas como valor por defecto, que se aplicará en cado de que no se incluya ninguna de las alternativas.

2 COMPONENTES DE UN ESQUEMA RELACIONAL

En SQL se dispone de instrucciones para crear, modificar y eliminar componentes de distintos tipos que, juntos, constituirán un esquema relacional de base de datos.

Las componentes de las que se va a presentar las instrucciones de definición son las siguientes:

- Relación o tabla.
- Vista
- Permiso

Otros tipos de componentes como las restricciones de integridad generales (*assertions*) no se presentan ya que no están disponibles actualmente en casi ningún SGBD. Tampoco se van a presentar los *disparadores* o *triggers* que son unos elementos que incorporan al sistema un comportamiento activo[1].

3 DEFINICIÓN DE TABLA

Antes de mostrar la sintaxis se presentan unos ejemplos de definición de tablas utilizadas en prácticas, tal y como pueden encontrarse en el documento UD2.2.

Ejemplo 1

Definición de la tabla *Puerto* de la base de datos CICLISMO.

```
CREATE TABLE puerto
(nompuerto VARCHAR2(35) CONSTRAINT PK_puerto PRIMARY KEY,
```

[1] Podéis encontrar información sobre los disparadores en cualquier texto sobre bases de datos como por ejemplo en "*Sistemas de Bases de Datos: un enfoque práctico para diseño, implementación y gestión"* de T. M. Conolly y C.E. Begg.

```
 altura NUMBER(4),
 categoria CHAR(1),
 pendiente NUMBER(3,2),
 netapa NUMBER(2) NOT NULL CONSTRAINT FK_puerto_eta REFERENCES etapa (netapa),
 dorsal NUMBER(3) CONSTRAINT FK_puerto_cicli REFERENCES ciclista (dorsal));
```

Puede verse que la instrucción comienza con unas palabras reservadas (CREATE TABLE) y a continuación se da el nombre a la nueva tabla (*puerto*), seguido por la definición de los elementos que la constituyen encerrados entre paréntesis. En este caso hay seis elementos que corresponden a los seis atributos de dicha tabla.

Ejemplo 2

Definición de la tabla *Llevar* de la base de datos CICLISMO.

```
CREATE TABLE llevar
(dorsal NUMBER(3) NOT NULL CONSTRAINT FK_llevar_cicli
                                    REFERENCES ciclista (dorsal),
 netapa NUMBER(2) CONSTRAINT FK_llevar_etapa REFERENCES etapa (netapa),
 codigo CHAR(3) CONSTRAINT FK_llevar_mai REFERENCES maillot (codigo),
 CONSTRAINT PK_lle PRIMARY KEY (netapa, codigo));
```

En la definición de esta tabla puede verse que hay cuatro elementos, tres de ellos corresponden a los tres atributos con sus restricciones y el cuarto es una restricción que afecta a varios atributos (la de clave primaria).

Así, la sintaxis de una instrucción de definición de tabla es la siguiente:

CREATE TABLE *nom_tabla* (elemento_tabla$_1$, elemento_tabla$_2$, ..., elemento_tabla$_n$)

La sintaxis de un elemento de tabla es:

definición_atributo | restricción_tabla

A continuación, se describe la sintaxis de la definición de atributo:

nom_atributo tipo _datos [DEFAULT {valor | NULL}]
  [restricción_atributo$_1$ restricción_atributo$_2$ … restricción_atributo$_n$]

Para cada atributo se define el nombre, el tipo de datos sobre el que podrá tomar valores, y la lista de restricciones del atributo (separadas por espacios en blanco, no por comas).

Los tipos de datos que se pueden usar son los siguientes (se presentan sólo algunos):

{VARCHAR [(*n*)]| CHAR [(*n*)]| INTEGER [(*n* [,*n*])] | DATE}

Como puede verse en la sintaxis, se puede definir un valor por defecto para cada atributo, que deberá ser del tipo de datos definido, o el valor nulo.

Las restricciones definidas sobre un atributo tienen la sintaxis siguiente:

```
[CONSTRAINT nombre_restricción]
  {NOT NULL
  | UNIQUE
  | PRIMARY KEY
  | REFERENCES nom_relación [(nom_atributo)]
     [ON DELETE {CASCADE| SET NULL| SET DEFAULT| NO ACTION}]
     [ON UPDATE {CASCADE| SET NULL| SET DEFAULT| NO ACTION}][2]
  | CHECK (condición_búsqueda)
  [cuándo_comprobar]
```

Las restricciones pueden tener un nombre asociado, que se asigna utilizando la palabra CONSTRAINT y que permitirá que posteriormente se cite la restricción por su nombre y, si el nombre es significativo, informará de qué restricción se trata. En caso de no darles nombre, el SGBD le da a cada una un nombre que no tiene ningún significado. Las distintas restricciones que pueden definirse son las correspondientes a las

[2] Esta directriz no está disponible en Oracle

restricciones estudiadas en el modelo relacional, es decir: valor no nulo (NOT NULL), unicidad (UNIQUE), clave primaria (PRIMARY KEY), clave ajena (REFERENCES *nom_relación* [(*nom_atributo*)]) donde:

- El tipo de integridad referencial que se comprueba en Oracle es el débil, no siendo posible, hoy por hoy, especificar ni el parcial ni el completo.
- La directriz de borrado (ON DELETE {CASCADE| SET NULL| SET DEFAULT| NO ACTION}), cuyas opciones corresponden a las estudiadas en el modelo relacional.
- La directriz de modificación (ON UPDATE {CASCADE| SET NULL| SET DEFAULT| NO ACTION}) también vistas en el modelo relacional.

También es posible especificar restricciones específicas sobre el atributo (CHECK…).

Todas las restricciones tienen una propiedad que indica al SGBD cuándo se debe comprobar. En SQL existen dos instantes posibles de evaluación de las restricciones frente a la ejecución de las operaciones de actualización:

- IMMEDIATE: inmediato, es el valor por defecto. La evaluación inmediata implica que después de cada operación de actualización se evalúan las restricciones que pudieran verse afectadas por la actualización y, en el caso de que alguna se viole, se rechaza la operación.[3]
- DEFERRED: diferido. En la evaluación diferida, el SGBD espera hasta que el usuario pida la confirmación de la transacción que está en curso (que puede incluir varias operaciones de actualización) y antes de confirmar ésta comprueba la integridad de la base de datos. Si alguna restricción es violada, el SGBD deshace la transacción completa.

También es posible indicar que la restricción puede cambiar el instante de evaluación definido (de inmediato a diferido o de diferido a inmediato). Para permitir este cambio, la restricción se debe definir como DEFERRABLE. Si no se desea permitir el cambio la restricción siempre se evaluará en el instante que se le haya definido, en este caso se debe especificar NOT DEFERRABLE (es el valor por defecto).

La sintaxis de definición de cuándo_comprobar es la siguiente:

```
[[NOT] DEFERRABLE] [INITIALLY {IMMEDIATE | DEFERRED}]
```

Esta cláusula, teniendo en cuenta las partes opcionales, da lugar a las siguientes combinaciones que se aclaran a continuación:

1. NOT DEFERRABLE INITIALLY IMMEDIATE: define una restricción como no diferible y con modo inicial inmediato. La restricción se evaluará tras cada instrucción que pueda violarla. Si la restricción se viola, la instrucción es abortada pero la transacción continúa. Además, al ser no diferible, el usuario no podrá cambiar el momento de la evaluación.
2. DEFERRABLE INITIALLY IMMEDIATE: define una restricción como diferible y con modo inicial inmediato. La restricción se evaluará tras cada instrucción que pueda violarla. Si la restricción se viola, la instrucción es abortada pero la transacción continúa. Además, al ser diferible, el usuario podrá cambiar la evaluación al modo diferido.
3. DEFERRABLE INITIALLY DEFERRED: define una restricción como diferible y con modo inicial diferido. La restricción se evaluará tras cada transacción que contenga una instrucción que pueda violarla. Si la restricción se viola, la transacción es abortada en su totalidad. Además, al ser diferible, el usuario podrá cambiar la evaluación al modo inmediato.
4. DEFERRABLE: equivale a DEFERRABLE INITIALLY IMMEDIATE (caso 2).
5. NOT DEFERRABLE: equivale a NOT DEFERRABLE INITIALLY IMMEDIATE (caso 1).
6. INITIALLY IMMEDIATE: equivale a NOT DEFERRABLE INITIALLY IMMEDIATE (caso 1).
7. INITIALLY DEFERRED: equivale a DEFERRABLE INITIALLY DEFERRED (caso 3).
8. NOT DEFERRABLE INITIALLY DEFERRED: este caso está prohibido. Es decir, cuando una restricción es inicialmente definida como diferible, siempre se permite el cambio a inmediato.
9. Si no se especifica ninguna parte de esta cláusula la restricción es NOT DEFERRABLE INITIALLY

[3] Hay que darse cuenta de que este comportamiento implica que la transacción puede no ejecutarse completa implicando por tanto que no se cumpliría la atomicidad en el procesamiento de esa transacción.

IMMEDIATE (caso 1).

El cambio entre los dos estados de una restricción definida como DEFERRABLE se hace dinámicamente dentro de una transacción y sólo tiene efecto mientras la transacción está activa. La instrucción es la siguiente:

```
SET CONSTRAINT {nombre_restricción1,…, nombre_restricciónn| ALL}
    {IMMEDIATE  | DEFERRED}
```

La opción ALL hace que se aplique el cambio a todas las restricciones diferibles de la base de datos.

Esta instrucción actúa sobre la transacción en la que se incluye hasta la siguiente aparición de la misma instrucción o hasta el final de la transacción.

Como se ha visto más arriba, un elemento de una tabla puede ser también una restricción de tabla que, normalmente, afecta a más de un atributo de la tabla en la que se define (aunque en realidad se puede incluir cualquier restricción, aunque afecte sólo a un atributo). La sintaxis de estas restricciones es la siguiente:

```
[CONSTRAINT nombre_restricción]
  {UNIQUE (nom_atributo1, nom_atributo2, ..., nom_atributon)
  | PRIMARY KEY (nom_atributo1, nom_atributo2, ..., nom_atributon)
  | FOREIGN KEY (nom_atributo1, nom_atributo2, ..., nom_atributon)
  REFERENCES nom_tabla [(nom_atributo1, nom_atributo2, ..., nom_atributon)]
     [ON DELETE {CASCADE| SET NULL| SET DEFAULT| NO ACTION}]
     [ON UPDATE {CASCADE| SET NULL| SET DEFAULT| NO ACTION}]
  | CHECK (condición_búsqueda)}
  [cuándo_comprobar]
```

Se puede apreciar que las restricciones posibles son las mismas que en el caso de restricciones de atributo (excepto NOT NULL), pero la sintaxis cambia ya que se debe incluir entre paréntesis el nombre de los atributos afectados por la restricción, y en el caso de la clave ajena, se incluyen las palabras FOREIGN KEY.

4 MODIFICACIÓN DE LA DEFINICIÓN DE TABLA

La definición de una relación o tabla se puede modificar con la instrucción siguiente:

```
ALTER TABLE nombre_tabla
    {ADD definición_atributo
    |MODIFY [COLUMN ] (nombre_atributo)
    {DROP DEFAULT|
       SET DEFAULT {literal | funcion_sistema | NULL}|
       ADD definicion_restriccion|
       DROP nombre_restriccion }
    | DROP [COLUMN ]  nombre_atributo {RESTRICT | CASCADE}}
```

Con esta instrucción se puede modificar la definición de una tabla. Se pueden añadir nuevos atributos, o modificar la definición de un atributo o eliminar un atributo con la opción de que sea restrictivo o en cascada en caso de que otros atributos dependan de éste.

Ejemplo 3

Para añadir un nuevo atributo a la tabla *Puerto* con la provincia donde está el puerto, se ejecutaría la siguiente instrucción:

```
ALTER TABLE Puerto
    ADD provincia VARCHAR(30) NOT NULL;
```

5 BORRADO DE TABLAS

Para eliminar una relación o tabla de una base de datos se usa la instrucción

```
DROP TABLE nom_relación {CASCADE CONSTRAINTS}
```

Con la opción CASCADE CONSTRAINTS se eliminarían todas las claves ajenas que hicieran referencia a la

tabla borrada.

Ejemplo 4

Para eliminar la tabla *Llevar* del esquema de ciclismo:

```
DROP TABLE Llevar;
```

6 DEFINICIÓN DE VISTAS

Si se quiere que, para determinados usos, esté disponible la información sólo de una parte de la BD, hay que definir una vista.

Ejemplo 5

Se desea tener disponible toda la información de las etapas que tienen puertos de montaña, ya que se van a hacer consultas frecuentes sobre estas etapas.

Lo más conveniente es definir una vista, para poder consultar directamente sobre ella la información que se desee. Esto se haría de la siguiente forma:

```
CREATE VIEW Etapas_con_puertos AS
  SELECT * FROM Etapa
  WHERE netapa IN (SELECT netapa FROM Puerto);
```

A partir de la creación de una vista, se pueden definir consultas sobre ella, como la siguiente:

Ejemplo 6

Obtener la longitud máxima de las etapas que tienen puertos de montaña.

```
SELECT MAX(km)
FROM Etapas_con_puertos;
```

Una vista es una consulta sobre la BD a la que se le da un nombre. A una vista también se le llama relación o tabla derivada, ya que su contenido (filas) se obtiene a partir de otras tablas que aparecen en la definición de la vista (las filas de la vista no se almacenan en disco). Por oposición a este nombre, a las relaciones definidas con CREATE TABLE se las llama relaciones básicas.

La definición de vistas se hace utilizando una instrucción con la sintaxis siguiente:

```
CREATE VIEW nombre_vista
[(nombre_atributo1, nombre_atributo2, … nombre_atributon)]
AS sentencia_SELECT [WITH CHECK OPTION]
```

Así pues, cualquier consulta puede almacenarse como un objeto más de la BD, en forma de vista. El SGBD almacenará la definición de la vista, pero no almacenará datos de la misma, sino que, cada vez que se use, se ejecutará la consulta que está en la definición de la vista para obtener la información deseada. Como puede verse en la sintaxis de la definición de vistas, puede darse un nombre a los atributos de la vista diferente del que tienen en las tablas de las que proceden.

Las vistas pueden utilizarse, en principio, como una tabla básica, es decir, se pueden consultar (sin limitaciones) y actualizar (con limitaciones). La actualización de una vista la realiza el SGBD trasladando la modificación a las tablas básicas que aparecen en la definición de la vista. Esta traslación está obviamente muy limitada, en algunos sistemas más que en otros, y sólo puede realizarse cuando no hay posible ambigüedad.

Ejemplo 7

Sea la vista que almacena las etapas de más de 100 km:

```
CREATE VIEW Etapa_larga AS
SELECT * FROM Etapa
WHERE km>100;
```

Las siguientes instrucciones de actualización serían posibles:

```
INSERT INTO Etapa_larga (netapa,km,salida,llegada)
  VALUES (23,120,'Valencia', 'Teruel');

DELETE FROM Etapa_larga WHERE KM=150;
```

La primera sería realizada por el SGBD insertando la fila (123, 120, 'Valencia', 'Teruel', NULL) en la tabla básica *Etapa* y la segunda eliminado de esta misma tabla todas las filas que cumplieran la condición de km=150.

Sea, sin embargo, la vista que almacena el nombre de los ciclistas, su dorsal y la cantidad de etapas que han ganado si es que han ganado alguna:

```
CREATE VIEW Ganadores AS
  SELECT C.dorsal,C.nombre,COUNT(*) AS Etapas_ganadas
  FROM Ciclista C, Etapa E
  WHERE C.dorsal=E.dorsal
  GROUP BY C.dorsal, C.nombre
```

Esta vista no podría actualizarse ya que cada fila de la vista es producida por una fila de la tabla *Ciclista* y por un conjunto de filas de la tabla *Etapa*. Así ante una instrucción como:

```
INSERT INTO Ganadores VALUES(150, 'Pepe Pérez', 5);
```

el SGBD no sabría cómo trasladar esta información (un nuevo ciclista que ha ganado 5 etapas) a las tablas básicas.

Cuando se usa la opción WITH CHECK OPTION se le indica al SGBD que las modificaciones o inserciones de tuplas que se realicen sobre la vista deben cumplir siempre la definición de la misma.

Ejemplo 8

Sea de nuevo la vista de etapas largas:

```
CREATE VIEW Etapa_larga AS
   SELECT * FROM Etapa
   WHERE km>100
```

Y sea la siguiente instrucción:

```
INSERT INTO Etapa_larga (netapa,km,salida,llegada)
       VALUES (23,40,'Teruel', 'Zaragoza');
```

El SGBD trasladaría esta modificación a la tabla básica *Etapa* insertando la fila (123, 40, 'Teruel', 'Zaragoza' NULL) pero, curiosamente, si después de la inserción se consultan las filas de la vista, la fila recién insertada no aparecería porque no cumple la condición de definición de la vista (40 km son menos que 100 km). Este comportamiento "anómalo" puede evitarse especificando la opción WITH CHECK OPTION en la definición de la vista:

```
CREATE VIEW Etapa_larga AS
   SELECT * FROM Etapa
   WHERE km>100
WHITH CHECK OPTION
```

Si la vista está definida así, el SGBD rechazará la operación de inserción propuesta.

En SQL estándar estas operaciones no están apenas restringidas, pero en los sistemas de gestión comerciales tienen bastantes limitaciones, permitiéndose sólo modificaciones o inserciones cuando en la definición de la vista no intervengan funciones agregadas, ni operadores conjuntistas, ni la palabra reservada DISTINCT. En el caso en que una vista esté definida sobre varias tablas, sólo estarán permitidas aquellas operaciones que afecten a una sola tabla, que será la que tenga como clave primaria la misma que serviría de clave primaria a la vista (si hubiera que definirla, cosa que no se hace).

Como se ha mostrado en el Ejemplo 6, las vistas pueden ser usadas para obtener información de la BD de la misma forma que se usa una tabla o relación, es decir, se pueden hacer consultas usando como nombre de tabla un nombre de vista, y en ese caso el SGBD ejecutará la consulta de la vista combinada con la consulta en donde se está utilizando ésta, y se obtendrá el resultado.

Las ventajas de las vistas son varias:

- Permiten definir esquemas externos, es decir, vistas parciales de la BD para distintos usuarios, porque necesiten acceder a distintos aspectos de los datos, o porque no deban acceder a algunos datos reservados.
- Permiten tener preparadas y guardadas consultas, bien porque se usen a menudo o bien porque sean costosas de definir.

7 BORRADO DE VISTAS

Cuando no sea necesaria una vista, se puede eliminar de la BD con una instrucción según la sintaxis siguiente:

```
DROP VIEW nombre_vista {CASCADE CONSTRAINTS}
```

Con la opción CASCADE CONSTRAINTS se eliminarían todas las claves ajenas que hicieran referencia a la vista borrada.

8 LA GESTIÓN DE AUTORIZACIONES EN SQL

La gestión de autorizaciones forma parte del DDL y consta de las instrucciones GRANT y REVOKE, para otorgar y revocar autorizaciones.

El propietario del esquema de una base de datos es el propietario de todos los elementos definidos en él. Para que otro usuario pueda realizar operaciones sobre los elementos de la base de datos debe tener la autorización necesaria; estas autorizaciones deben concederlas el propietario del objeto por medio de la sentencia GRANT. La sentencia REVOKE es la que permite eliminar privilegios recibidos.

```
GRANT
   { ALL |
    SELECT |
    INSERT [(nom_atr1,…, nom_atrn)] |
    DELETE |
    UPDATE [(nom_atr1,…, nom_atrn)]}
ON nom_relación TO {usuario1,…, usuariom | PUBLIC}
[WITH GRANT OPTION]

REVOKE
   { ALL |
    SELECT |
    INSERT [(nom_atr1,…, nom_atrn)] |
    DELETE |
    UPDATE [(nom_atr1,…, nom_atrn)]}
ON nom_relación FROM {usuario1,…, usuariom | PUBLIC}
```

Donde:

- *Nom_relación* es el nombre de una de las relaciones definidas en el esquema.
- *Usuario*$_i$ es el identificador de un usuario. Con la cláusula PUBLIC se otorgan o eliminan los privilegios a todos los usuarios
- Los privilegios otorgados o eliminados son:
  - SELECT*:* permite que el usuario consulte la relación.
  - INSERT [(*nom_atr*$_1$,…, *nom_atr*$_n$)]: permite que el usuario inserte tuplas en la relación dando valor sólo a los atributos especificados.
  - DELETE: permite que el usuario elimine tuplas de la relación.
  - UPDATE [(*nom_atr*$_1$,…, *nom_atr*$_n$)]: permite que el usuario modifique el valor los atributos especificados.
  - ALL: se otorgan todos los privilegios anteriores.
- La cláusula WITH GRANT OPTION otorga permiso para ceder a terceros los privilegios obtenidos.

Ejemplo 9

La instrucción que permitiría dar privilegio de lectura al usuario de login “Pepe” sobre la tabla *Puerto* sería la siguiente:

```
GRANT SELECT ON Puerto to Pepe;
```

Y para quitarle el privilegio:

```
REVOKE SELECT ON Puerto FROM Pepe;
```