Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

DEFINICIÓN DE DATOS

CREACIÓN DE TABLAS: TIPOS DE DATOS, RESTRICCIONES DE SEGURIDAD

Índice

Presentación

En esta unidad de aprendizaje estudiaremos el **LDD** que incorpora SQL. Recordamos que el LDD es un **sublenguaje para la definición de datos** que permite crear y modificar el esquema de la BD, chequear la integridad y definir autorizaciones. Tal y como ya vimos, el LDD nos debe permitir **definir la estructura** de la BD.

Al finalizar este tema podrás:

- **Crear el esquema de la BD**. Definir las relaciones que componen su esquema estableciendo los atributos de los que está compuesta cada una, sus tipos y la clave primaria.
- **Definir reglas para chequear la integridad de la BD** de forma que pueda implementar las restricciones semánticas que se dan en el mundo real. Estas **restricciones** deben permitirle indicar cosas como:
- Que es imprescindible almacenar el número de cuenta de un cliente.
- Que los valores para el atributo *puesto* de un empleado pueden ser comercial, administrativo o director.
- Que cada empleado se identifica por el DNI y por tanto no podemos repetir este atributo para dos empleados distintos.
- Que un pedido tiene que ser realizado por un cliente que previamente exista en la base de datos, en caso contrario, habría que darlo de alta.

La sentencia create table

La sentencia CREATE TABLE nos permite definir el esquema de las relaciones de la BD. La sintaxis es:

```
CREATE TABLE <tabla>
(       <columna1> <tipo> <restricción_dato_requerido>
<restricción_dominio>,
        <columna2> <tipo> <restricción_dato_requerido>
<restricción_dominio>,
…
        <columnan> <tipo> <restricción_dato_requerido>
<restricción_dominio>,
        PRIMARY KEY (<lista_columnas>),
        FOREIGN KEY (<lista_columnas>) REFERENCES <tabla_ext>
(<lista_col_ext>)
) ;
```

Donde:

- ***<tabla>*** es el nombre que le daremos a esta relación. Dicha tabla tiene **n atributos** llamados *<columna1>*, *<columna2>*,…,*<columnan>*.
- Para cada atributo de la relación se especifica:
    - El **tipo de dato** que es, *<tipo>*. El tipo de dato siempre tiene que estar especificado.
    - Si tiene o no que **aparecer obligatoriamente** y si es **único** para cada fila de la tabla, *<restricción_dato_requerido>*. Puede no aparecer.
    - Cuál es el **dominio** de este atributo, *<restricción_dominio>* . Puede no aparecer.
- **PRIMARY KEY** indica el conjunto de atributos que formarán la clave principal de la tabla. Sólo habrá una cláusula PRIMARY KEY por tabla.
    - *<lista_columnas>* es la lista de los atributos separados por comas que forman la clave primaria.
    - No se podrá crear una fila cuyos valores de la clave primaria están duplicados, garantizando así la unicidad de la clave principal.
- **FOREIGN KEY** especifica las claves externas (o ajenas).

adulto_con_bebe

Ejemplo

ADULTO_CON_BEBE (dni, nombre_bebe, fecha_nac_bebe)

```
CREATE TABLE adulto_con_bebe (
     dni VARCHAR(9),
     nombre_bebe VARCHAR(20),
     fecha_nac_bebe DATE,
     PRIMARY KEY (dni),
     FOREIGN KEY (dni) REFERENCES pasajero(dni));
```

Tipos de datos. Escalares

Los tipos de datos que se van a exponer aquí se corresponden con el **estándar ISO**. Existen variaciones en función del SGBD concreto que se esté utilizando (Oracle, MySQL, etc.).

Los tipos de datos escalares que se pueden utilizar son:

Booleanos	Almacenan valores **TRUE** y **FALSE**. `BOOLEAN`

Caracteres	Especifican cadenas de caracteres. Pueden tener una longitud fija ó variable.
	
	`CHARACTER ó CHAR (<longitud>)`
	`CHARACTER VARYING ó VARCHAR (<longitud>)`
	
	• **<Longitud>**: nº máximo de caracteres. Por defecto 1. **Ejemplo**.
	• A las cadenas de caracteres se les pueden aplicar las siguientes **funciones**.

Datos de bit	Utilizado para definir secuencias de bits (0 y 1). `BIT (<Longitud>)` Ejemplo: **cadena_bits BIT (5): 00110**

Ejemplo

Cadena de caracteres

```
cadena CHAR(7) Hola___: rellena con blancos

cadena VARCHAR(7) Hola : ahorra espacio
```

funciones

- CHAR_LENGTH: longitud de una cadena de caracteres. CHAR_LENGTH(hola): 4
- ||: concatena dos cadenas de caracteres o bits. nombre || apellido
- LOWER: convierte a minúsculas. LOWER(SELECT...)
- UPPER: convierte a mayúsculas

Tipos de datos. Datos numéricos y de fecha y hora

Tipos de datos numéricos

```
NUMERIC  (<precisión>, <escala>)
DECIMAL  (<precisión>, <escala>) ó DEC
INTEGER ó INT
SMALLINT
FLOAT <precisión>
REAL
DOUBLE PRECISION
```

- **NUMERIC Y DECIMAL**: almacenan números en notación decimal.
  Incluye **< precisión>** y **< escala >**.
- **INTEGER**: para números enteros de gran tamaño (positivos y negativos).
- **SMALLINT**: para números enteros de pequeño tamaño (positivos y negativos).
- **FLOAT**, **REAL** Y **DOUBLE PRECISION** se utilizan para representar los números reales. PRECISION: indica la precisión de la mantisa. La precisión de REAL y DOUBLE PRECISION varía según la implementación.

Tipos de datos de fecha y hora

```
DATE
TIME <precisiónTemporal> WITH TIME ZONE
TIMESTAMP <precisiónTemporal> WITH TIME ZONE
```

- **DATE**: almacena fechas usando los campos **YEAR**, **MONTH** y **DAY**.
- **TIME**: almacena horas usando los campos **HOUR**, **MINUTE** y **SECOND**.
- **TIMESTAMP**: para almacenar fecha y hora.
- **<precisiónTemporal>** permite especificar el número de decimales usados para los segundos (0 por defecto).**WITH TIME ZONE** se usa para indicar la zona horaria. Se pueden utilizar los operadores **CURRENT_DATE**, fecha actual de la zona horaria del usuario, y **CURRENT _TIME**, hora actual del usuario.

Características de mejora de integridad

Enunciaremos algunas funciones propuestas por el estándar SQL para controlar la integridad. EL objetivo es cumplir las restricciones impuestas en el mundo real para proteger los datos ante posibles incoherencias. Estas restricciones se usan al crear o modificar las tablas. Además de las generales, consideraremos los siguientes tipos de restricciones:

- **Restricciones sobre datos requeridos**

En este caso es posible obligar al usuario a que introduzca un valor en una determinada columna de la tabla. De esta forma nos aseguramos de que, por ejemplo, si necesitamos obligatoriamente saber el número de cuenta de un cliente, no exista posibilidad de que el administrativo no la introduzca.

- **Restricciones sobre el dominio**

Se utilizan para obligar a que los valores introducidos coincidan con un conjunto determinado. Por ejemplo si vamos a almacenar el estado civil, aunque el tipo de dato sea un *varchar*, obligar a que solo pueda ser *soltero, casado, separado, viudo ó divorciado.*

- **Restricciones sobre la integridad de las entidades**

Sirven para asegurarse de que una misma instancia de una relación no se introduzca dos veces en la tabla. Nos tenemos que asegurar de que las claves primarias, que identifican de forma única cada instancia, tengan valores distintos para cada fila que insertemos en la tabla.

- **Restricciones sobre la integridad de las referencias**

Son útiles en los casos en los que al insertar el valor para el atributo que es clave externa de otra tabla, se compruebe que realmente existe una instancia con este valor en la tabla externa.

En las siguientes pantallas veremos en detalle cada una de estas restricciones.

Restricciones sobre datos requeridos

Al insertar una fila en una tabla es posible no especificar los valores para una determinada columna. De esta forma se puede utilizar la sentencia:

```
INSERT INTO clientes(dni) VALUES (23456789)
```

Con esta sentencia se inserta un cliente cuyo único dato conocido es su **DNI** en la base de datos. El resto de atributos como: calle, ciudad, tarjeta de crédito, etc., los establece el SGBD a **NULL**.

Existen ocasiones en las que, en el mundo real, no se debe permitir que esto ocurra. Por ejemplo lo más normal es que se obligue a almacenar, al menos, el número de cuenta de un cliente al que poder cargar los pedidos que realiza. Para estos casos se puede utilizar la restricción sobre datos requeridos.

Para ello, al definir ó modificar el esquema de la tabla, se especifica que dicha columna no puede ser nula, **NOT NULL**.

```
<columna1> <tipo> NOT NULL
```

Ejemplo:

```
n_tarjeta VARCHAR(15) NOT NULL
```

Restricciones sobre el dominio

Las restricciones sobre el dominio son aquellas que permiten implementar situaciones en el mundo real donde el dominio de un determinado atributo se restringe a un conjunto de valores posibles.

El estándar SQL provee de dos maneras de especificar el dominio de los valores de un atributo (o columna):

- **CHECK**: define una restricción sobre una columna o tabla completa.

```
CHECK (<condicion>)
```

Ejemplo: estadoCivil VARCHAR(7)CHECK (estadoCivil IN (casado, soltero))

- **CREATE DOMAIN**: define dominios. Permite definir un tipo de dato especial donde se especifican los valores que va a contener. Luego el dominio creado se utiliza en la sentencia CREATE TABLE en lugar del tipo del campo.

```
CREATE DOMAIN <nombreDominio> [AS] <tipoDatos> DEFAULT
<valorPorDefecto> CHECK VALUE
```

IN (<**listaValoresValidos** >);

Lista valores válidos

<listaValoresValidos> puede ser una selección de datos de una tabla.

Ejemplo:

```
CREATE DOMAIN nombresClientes AS VARCHAR(25) CHECK (VALUE IN
(SELECT nombre FROM cliente));
```

CREATE DOMAIN

CREATE DOMAIN

```
CREATE DOMAIN TipoEstadoCivil AS VARCHAR
DEFAULT soltero
CHECK (VALUE IN (casado, soltero)) ;

estadoCivil TipoEstadoCivil NOT NULL
```

Los dominios pueden eliminarse usando **DROP DOMAIN**:

```
DROP DOMAIN NombreDominio [RESTRICT | CASCADE]
```

- **RESTRICT**: impide eliminar el dominio si está siendo usado.
- **CASCADE**: las columnas de tabla que usen el dominio adoptarán el tipo de datos sobre el que se construye dicho dominio.

Restricciones de integridad de las entidades

La **restricción de integridad de las entidades** es aquella que obliga a que la clave principal de cada relación (o tabla) tenga un valor único y no nulo para cada tupla (o fila).

Para definir una clave primaria se usa la sentencia **PRIMARY KEY**:

```
PRIMARY KEY (lista_de_atributos)
```

Ejemplo:

```
PRIMARY KEY (dni)
PRIMARY KEY (nombre, teléfono)
```

La cláusula PRIMARY KEY sólo puede usarse una vez por tabla.

Para garantizar la unicidad de otros atributos se puede usar la sentencia **UNIQUE**. De esta forma, la columna que vaya acompañada de la cláusula UNIQUE, tanto en la creación como en la alteración del esquema de la tabla, no podrá contener valores duplicados. Si intentamos insertar dos filas en la tabla con el mismo valor para este atributo el SGBD creará un error y no nos permitirá insertarlas. Puede haber tantas como sean necesarias.

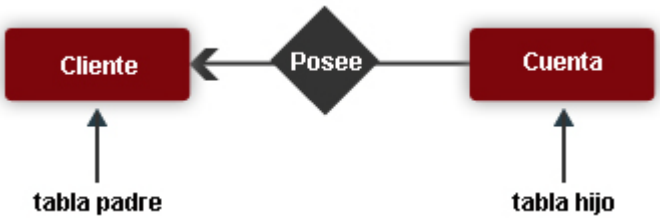
Cada **columna** (atributo) que se especifique como UNIQUE, debe ser declarado también como **NOT NULL**. Si permitimos introducir valores desconocidos para un atributo definido como UNIQUE, se estarían introduciendo valores **NULL** en el campo correspondiente. Estaríamos, por tanto, repitiendo valores y contradiciendo la cláusula.

Restricciones de integridad de las referencias

La **restricción de integridad referencial** obliga a que, si una clave externa posee un valor, este debe hacer referencia a uno que exista y sea válido en la tabla para la que es clave primaria. De esta forma no se podrá hacer un INSERT o UPDATE en el que se trate de dar un valor a una clave externa que no se corresponda con un valor válido para la tabla de la que procede.

Cuenta (ccc, saldo, dni)

- ccc: clave principal/ propia
- dni: clave externa/ ajena de cuenta
- dni: clave principal de cliente



No podemos introducir una tupla en **cuenta** con un **dni** que no exista previamente en la **tabla cliente**.

Para definir restricciones de integridad de referencias se usa la cláusula **FOREING KEY** que puede aparecer dentro de una sentencia **CREATE TABLE** o una sentencia **ALTER TABLE**:

```
FOREIGN KEY (<claveexterna>) REFERENCES <tablaexterna>
ON DELETE <accion>
ON UPDATE <accion>
```

Donde:

- **<claveexterna>** es el nombre del atributo que es clave externa.
- **<tablaexterna>** es la tabla donde esta clave externa es clave primaria ó candidata.
- **ON DELETE <accion>** y **ON UPDATE <accion>** permite definir la acción a realizar cuando se elimina la instancia de la tabla externa.

Ejemplo:

```
FOREIGN KEY (dni) REFERENCES cliente
```

Acciones en las restricciones de integridad de las referencias

Cuando se hace un **DELETE** sobre un valor de la tabla **padre** que es clave externa en otra, SQL lleva a cabo las acciones indicadas con la cláusula **ON DELETE** al definir dicha clave ajena con **FOREIGN KEY**. Estas son:

CASCADE	Borra la fila en la tabla **padre** y las correspondientes en la tabla **hijo**. Si hay otras claves externas, se aplicarán las reglas definidas para estas, y así sucesivamente en cascada.
SET NULL	Se elimina la fila de la tabla **padre** y se pone a **NULL** los valores de la clave externa en la tabla **hijo**, siempre que no tenga la restricción **NOT NULL**.
SET DEFAULT	Borra la fila de la tabla **padre**, y asigna el valor por defecto a las claves externas en la tabla **hijo**, siempre que se definiese dicho valor.
NO ACTION	Es la acción predeterminada si no se indica la regla **ON DELETE**. Impide borrar la fila de la tabla **padre**.

Ejemplo:

```
FOREIGN KEY (dni) REFERENCES cliente ON DELETE SET NULL
```

A la hora de actualizar un valor con **UPDATE**, se trabaja de modo similar que con la cláusula **ON UPDATE**, y puede indicarse así el modo en que se modificarán los datos de la tabla **hijo** afectados por el cambio en la tabla **padre**.

Ejemplo:

```
FOREIGN KEY (dni) REFERENCES cliente ON UPDATE CASCADE
```

Resumen

El LDD de SQL permite **crear el esquema**, **chequear la integridad**, **modificar el esquema** y **definir autorizaciones** en una base de datos. En este tema hemos visto los dos primeros.

El esquema se crea a través de la sentencia **CREATE TABLE** con la siguiente sintaxis:

```
CREATE TABLE <tabla>
(<columna1> <tipo> <resDatoRequerido> <resDominio>,
    PRIMARY KEY (<listaAtributos>),
    FOREIGN KEY (<listaAtributos>) REFERENCES <tabla_ext>);
```

Se permite chequear la integridad de:

- **Datos requeridos**. Utilizando la sintaxis **NOT NULL** en <resDatoRequerido>.
- **Dominio**. A través de la cláusula **CHECK** en <resDominio> y de la sentencia **CREATE DOMAIN** para crear el dominio y utilizarlo en tipo>
- **Entidades**. A través de **PRIMARY KEY** y **UNIQUE**.
- **Referencias**. A través de **FOREIGN KEY**.