tisBDdocs

P02 Definición de datos

CONTENIDOS

1 Definición de datos
1.1 Create table
1.1.1 Restricciones
1.2 Drop table
1.3 Alter table
2 Manipulación de datos
2.1 Insert
2.2 Nulos (NULL)
2.3 Inserción de más de una fila
2.4 Delete

Definición de datos

Create table

La definición de tablas es el primer paso en la creación de una base de datos. El conjunto de descripciones de tablas conforma el esquema de base de datos y representa a un sistema de información concreto.

Supongamos que vamos a implementar un esquema de base de datos relacional de profesores, asignaturas (sólo es un listado de profesores y asignaturas, sin relaciones entre ellos). En primer lugar debemos decidir cuáles son los atributos de cada uno de ellos y sus tipos de datos:

PROFESORES	ASIGNATURAS
DNI: varchar(10), **nombre**: varchar(40), **categoría**: char(4), **ingreso**: date	**código**: char(5), **descripción**: varchar(35), **créditos**: decimal(3,1), **créditosp**: decimal(3,1)

Para cumplir con las restricciones del modelo relacional, además, debemos elegir las claves primarias adecuadas : DNI para profesores y código para asignaturas. Obviamente, la forma que tienen estas tablas ha sido una decisión nuestra como diseñadores de esta base de datos concreta, en otra situación probablemente hubiéramos decidido definir otros atributos y otras tablas.

La orden CREATE TABLE nos permite crear cada una de las tablas necesarias para nuestra base de datos:

```
CREATE TABLE nombreTabla ( {listaColumnas} [,{restricciones}] )
```

La lista de columnas, en su forma más sencilla, es un conjunto de expresiones (tantas como columnas deseemos, y separadas por comas) del tipo:

```
columna tipoDatos[,columna tipoDatos[, ...]]
```

La totalidad de tipos de datos que maneja MySQL, siendo la mayoría comunes con ligeras diferencias a cualquier motor de base de datos, se puede encontrar en http://dev.mysql.com/doc/refman/5.1/en/data-types.html.

Restricciones

Las restricciones son reglas, que normalmente se establecen en el momento de crear una tabla, para garantizar la integridad de los datos.

Básicamente, las restricciones obligan a cumplirse ciertas reglas cuando una fila es insertada, borrada o modificada, de forma que la operación se llevará a efecto sólo si se cumplen las restricciones definidas en la tabla.

Podemos contemplar los siguientes tipos de restricciones de integridad de datos:

- *NOT NULL*: especifica que la columna no puede contener un valor nulo.
- *PRIMARY KEY*: identifica de manera única a cada fila de la tabla mediante una o varias columas, estas columnas que forman la clave primaria no pueden tener valores nulos.
- *FOREIGN KEY*: establece una relación entre una(s) columna(s) de la tabla y otra(s) columna(s) de la tabla referenciada, siendo esta última(s) columna(s) la PRIMARY KEY
- *UNIQUE*: no permite duplicados; combinado con NOT NULL es la forma de definir una clave alternativa.
- *CHECK*: especifica una condición que se debe evaluar a "cierto".

De las restricciones, sólo vamos a utilizar, de momento, la clave primaria, que puede contener tantas columnas como se necesiten:

```
PRIMARY KEY (columna[,columna[, ...]])
```

Las siguientes órdenes crean las tablas PROFESORES y ASIGNATURAS

```
create table profesores (
DNI varchar(10),
nombre varchar(40),
categoria char(4),
ingreso date,
primary key (DNI));

create table asignaturas (
codigo char(5),
descripcion varchar(35),
creditos decimal(3,1),
creditosp decimal(3,1),
primary key (codigo));

create table imparte (
dni varchar(10),
asignatura char(5),
primary key (dni,asignatura));
```

Drop table

Si queremos borrar una tabla debemos ordenárselo al SGBD mediante DROP TABLE:

DROP TABLE nombreTabla

Borra la tabla asignaturas

```
drop table asignaturas
```

Al utilizar esta orden también se eliminan los datos (las filas) que pudiera contener (en este caso, ninguna). Se puede borrar y crear la tabla tantas veces como queramos.

Alter table

En ocasiones nos equivocamos o necesitamos cambiar la definición de una tabla. A nadie se le escapa que eliminando la tabla y volviéndola a crear (*drop table...; create table...*) se soluciona el problema pero no siempre es más rápido o recomendable hacerlo así. Por

ejemplo, si la tabla ya tiene datos o si tiene relación con otras tablas es preferible utilizar ALTER TABLE.

Esta es una orden compleja por la cantidad de opciones que ofrece. Podemos añadir, eliminar y modificar casi cualquier detalle de una tabla. Supongamos una tabla mitabla ya creada a la que queremos añadir una columna más:

```
ALTER table mitabla ADD nuevacol integer not null
```

Acabamos de añadir una columna "nuevacol" de tipo entero (*integer*) y, además, le hemos añadido una restricción de valor no nulo (*not null*).

También podemos eliminar una columna:

```
ALTER table mitabla DROP nuevacol
```

O añadir una restricción de clave primaria —suponiendo que existe una columna "id"—:

```
ALTER table mitabla ADD primary key (id)
```

En este caso el SGBD comprueba que en esa columna no haya ni duplicados ni nulos. De haberlos, mostraría un mensaje de error y no haría nada.

También podemos cambiar el nombre de una columna:

```
ALTER table mitabla CHANGE id ident
```

Hemos sustituido el nombre de la columna "id" por "ident".

O cambiar el tipo de datos de una columna:

```
ALTER table mitabla MODIFY id VARCHAR(25)
```

Como decíamos, hay muchas opciones y no todas son estándar por lo que mejor consultemos el manual cuando necesitemos esta orden: http://dev.mysql.com/doc/refman/5.5/en/alter-table.html.

Manipulación de datos

Insert

Para introducir datos nuevos en una base de datos vamos a utilizar la orden INSERT de SQL. Con la sintaxis que se muestra a continuación seremos capaces de introducir datos nuevos en cualquiera de las tablas que componen una determinada BD. En principio, veremos la expresión mínima de la orden, formada por dos cláusulas, INTO y VALUES.

```
INSERT INTO nombreTabla VALUES ( listaExpresiones )
```

Alta un nuevo profesor con los siguientes datos:

- dni 55555555
- nombre PATRICIO MARTÍNEZ,
- categoría TU
- fecha de incorporación 01/01/2000

```
insert into profesores
values ('55555555','PATRICIO MARTINEZ','TU','2000-1-1')
```

Alta un nuevo profesor con los siguientes datos:

- dni 66
- nombre ERNESTO PEREZ
- categoría ASO
- fecha de incorporación 01/01/1985

```
insert into profesores
values ('66','ERNESTO PEREZ','ASO','1985-1-1');
```

El resultado que devuelve una orden INSERT, será siempre el número de filas insertadas, en el caso de que la ejecución haya sido correcta. Para los casos en que la ejecución de la sentencia viole alguna restricción de la BD y por tanto, su ejecución no sea correcta, el resultado indicará cuál es la restricción violada.

El SGBD, cada vez que insertamos un nuevo dato en una tabla, se encarga de verificar las restricciones activas, en nuestro caso las claves primarias, que como sabemos, no admiten valores duplicados, ni valores nulos.

Alta un nuevo profesor con los siguientes datos:

- dni 66
- nombre JUAN JUANÍTEZ
- categoría XXX
- fecha de incorporación 01/01/1905

```
insert into profesores
values ('66','JUAN JUANITEZ','XXX','1905-1-1')
```

A veces no nos interesa o no podemos darle valor a todas y cada una de las columnas, o lo vamos a hacer en un orden distinto al que tienen las columnas en el create table que la definió. Especificar una lista de columnas antes de VALUES permite decirle al sistema qué columnas van a tener valor y cuál es.

Da de alta a un profesor con DNI 88888888 y nombre ARMANDO SUÁREZ

```
insert into profesores (dni, nombre)
values ('88888888', 'ARMANDO SUAREZ');
```

El sistema intentará asignar a las columnas no indicadas el valor por defecto, si se ha definido, o valor nulo.

Es recomendable acostumbrarse a poner siempre las columnas a las que se va a dar valor, sean o no todas las de la tabla. Las razones que lo aconsejan son:

- No habrá que fijarse en si se va a dar valor a todas o sólo a alguna de las columnas para acomodar la sintaxis de la sentencia INSERT.
- Si por alguna razón se modifica la estructura de una tabla, es decir, se añaden columnas nuevas, y tenemos costumbre de no indicar las columnas cuando se inserta valor a todas, con la modificación dejarán de funcionar las sentencias que tuviéramos escritas.

Nulos (NULL)

Las BD relacionales trabajan con un valor especial, **NULL**, que significa "ignorancia", se desconoce si tiene valor o no, y en el caso de tenerlo, cuál es. Nótese que NULL no es "cadena vacía" ni "blanco"; éstos son valores concretos, pertenecientes al tipo de datos cadena de caracteres. Normalmente, aunque es discutible, se simplifica su significado dejándolo en "no tiene valor".

A la pregunta de "asignaturas que no tienen créditos prácticos" todas estas soluciones dan resultados vacíos o erróneos

```
select *
from asignaturas
where creditosp = ''
```

```
--comilla-simple + comilla-simple = cadena vacía
```

```
select *
from asignaturas
where creditosp = ' '
```

```
--comilla-simple + espacio + comilla-simple = espacio_en_blanco
```

```
select *
from asignaturas
where creditosp = 0
```

```
select *
from asignaturas
where creditosp = NULL
```

Lo que estamos buscando realmente es

```
select *
from asignaturas
where creditosp is NULL
```

codigo	descripcion	creditos	creditosp
HI	HISTORIA DE LA INFORMATICA	4.5	*null*

> Efectivamente, hay una asignatura que no tiene créditos prácticos. Esto se debe a que, en la carga de la BD, se introdujo el valor NULL en la columna *creditosp* de la fila de la asignatura *HI*.
>
> Nótese que ni tan siquiera es posible utilizar la comparación habitual (signo "igual"), es obligado utilizar el operador ***IS NULL*** o ***IS NOT NULL***.

Existe la posibilidad de hacer uso del valor `NULL`, siempre que la columna lo admita. Aunque se suele simplificar por "ausencia de valor", recuérdese que NULL significa realmente ignorancia (no sabemos si tiene valor ni, si lo tuviera, cuál es). En cualquier caso, si la columna los admite se puede especificar en la orden de inserción:

```
insert into profesores (dni,nombre, categoría, ingreso)
values ('88888888','ARMANDO SUAREZ', NULL, NULL);
```

En realidad el efecto es el mismo que el de la orden anterior propuesta —ejecutarla ahora provocaría un error por duplicado en clave primaria—.

Igualmente, el cambio de orden de las columnas se debe corresponder con la posición exacta del valor a asignar:

```
insert into profesores ( categoria, dni, nombre)
values (NULL, '88888888','ARMANDO SUAREZ');
```

Inserción de más de una fila

Supongamos que en la BD Ejemplo existiera una tabla llamada OPTATIVAS que contuviera los códigos y los créditos de aquellas asignaturas de carácter optativo.

Vamos a crear dicha tabla, eligiendo como clave primaria el código de la asignatura y poniendo además otra restricción, que todas las filas tengan un valor no nulo en la columna créditos

```
create table optativas (
asignatura char (5),
creditos decimal(3,1) not null,
primary key (asignatura));
```

Existe la posibilidad de insertar el resultado de una consulta en lugar de indicar la lista concreta de valores a insertar. Esto nos permite insertar varias filas en una tabla con una sola operación, en concreto, tantas filas como tuplas devuelva la ordenSELECT.

INSERT INTO nombreTabla [(listaColumnas)] **consulta**

Supongamos que serán optativas todas las asignaturas que tengan menos de 9 créditos. Se trata de introducir los códigos de dichas asignaturas en la tabla OPTATIVAS. Una opción es insertar en una sola operación el resultado de la consulta en la tabla OPTATIVAS.

```
insert into optativas (asignatura, creditos)
    select codigo, creditos from asignaturas where creditos < 9;
```

El resultado de la orden *select* deberá ser coherente en cantidad de columnas y tipos de datos. Las siguientes órdenes provocarán errores de compilación:

```
insert into optativas (asignatura)
     select codigo, creditos from asignaturas where creditos < 9;

insert into optativas (asignatura, creditos)
     select codigo from asignaturas where creditos < 9;

insert into optativas (asignatura, creditos)
     select dni, ingreso from profesores;
```

Este caso ya es diferente, hay que asegurarse de que el resto de columnas permiten la inserción:

```
insert into optativas (asignatura) select codigo from asignaturas
where creditos < 9
```

La restricción NOT NULL sobre la columna creditos impide que se realice la inserción de filas, para asegurar la integridad de los datos, evitando que se pongan valores nulos en esa columna.

Delete

Antes de comenzar, limpiemos la base de datos

```
drop table if exists imparte;
drop table if exists profesores;
drop table if exists asignaturas;

create table profesores (
DNI varchar(10),
nombre varchar(40),
categoria char(4),
ingreso date,
primary key (DNI));

create table asignaturas (
codigo char(5),
descripcion varchar(35),
creditos decimal(3,1),
creditosp decimal(3,1),
primary key (codigo));

create table imparte (
dni varchar(10),
asignatura char(5),
primary key (dni,asignatura));

insert into asignaturas (codigo, descripcion, creditos,
creditosp)
values ('HI', 'HISTORIA DE LA INFORMATICA', 4.5, NULL);
insert into asignaturas (codigo, descripcion, creditos,
creditosp)
values ('FBD', 'FUNDAMENTOS DE LAS BASES DE DATOS', 6.0,
1.5);
insert into asignaturas (codigo, descripcion, creditos,
creditosp)
values ('DGBD', 'DISEÑO Y GESTION DE BASES DE DATOS', 6.0,
3.0);
insert into asignaturas (codigo, descripcion, creditos,
creditosp)
values ('PC', 'PROGRAMACION CONCURRENTE', 6.0, 1.5);
insert into asignaturas (codigo, descripcion, creditos,
creditosp)
values ('FP', 'FUNDAMENTOS DE LA PROGRAMACION', 9.0, 4.5);


insert into profesores (dni, nombre, categoria, ingreso)
values ('21111222','EVA GOMEZ','TEU','1993-10-01');
insert into profesores (dni, nombre, categoria, ingreso)
values ('21222333','MANUEL PALOMAR','TEU','1989-06-16');
insert into profesores (dni, nombre, categoria, ingreso)
```

```
values ('21333444','RAFAEL ROMERO','ASO6','1992-06-16');

insert into imparte (dni, asignatura) values
('21111222','FBD');
insert into imparte (dni, asignatura) values
('21111222','DGBD');
insert into imparte (dni, asignatura) values
('21333444','PC');
```

La sentencia DELETE nos permite borrar las filas contenidas en una tabla.

```
DELETE [FROM] nombreTabla [WHERE condición]
```

Borrar todas las filas de la tabla IMPARTE

```
-- para ver lo que hay ahora
select * from imparte;
```

dni	asignatura
21111222	DGBD
21111222	FBD
21333444	PC

```
delete from imparte;

-- para ver el resultado
select * from imparte;
```

dni	asignatura

```
0 filas seleccionadas
```

Borra todas las asignaturas de menos de 5 créditos

```
delete from asignaturas where creditos < 5;
```

Para borrar filas de varias tablas habrá que ejecutar tantas sentencias DELETE como de tablas queramos borrar.

(Nota: en MySQL se puede borrar sobre varias tablas pero no se puede asegurar que otros SGBD lo permitan también)

BDgite (GITE-11014-UA),
Departamento de Lenguajes y Sistemas Informáticos,
Universidad de Alicante