INFORMATORIO

BASE DE DATOS

Lenguaje SQL: Inserción y modificación

Subsecretaría de Empleo
Chaco Gobierno de todos

Ministerio de Producción, Industria y Empleo
Chaco Gobierno de todos

CHACO
Gobierno de todos

INFORMATORIO

Parte 2: Inserción y modificación de datos

Vemos como insertar, modificar o borrar datos en nuestras tablas SQL.

2.1.- Creación de tablas

Explicamos la manera de **crear tablas** a partir de **sentencias SQL**. Definimos los **tipos de campos principales** y la **forma de especificar los índices**.

En general, la mayoría de las bases de datos poseen potentes editores de bases que permiten la creación rápida y sencilla de cualquier tipo de tabla con cualquier tipo de formato.

Sin embargo, una vez la base de datos está alojada en el servidor, puede darse el caso de que necesitemos introducir una nueva tabla ya sea con carácter temporal (para gestionar un carrito de compra por ejemplo) o bien permanente por necesidades concretas de nuestra aplicación.

En estos casos, podemos, a partir de una sentencia SQL, crear la tabla con el formato que deseemos lo cual nos puede ahorrar más de un quebradero de cabeza.

Este tipo de sentencias son especialmente útiles para bases de datos como Mysql, las cuales trabajan directamente con comandos SQL y no por medio de editores.

Para crear una tabla debemos especificar diversos datos: El nombre que le queremos asignar, los nombres de los campos y sus características. Además, puede ser necesario especificar cuáles de estos campos van a ser índices y de qué tipo van a serlo.

La sintaxis de creación puede variar ligeramente de una base de datos a otra ya que los tipos de campo aceptados no están completamente estandarizados.

2.1.1.- Sintaxis

```
Create Table nombre_tabla
(
nombre_campo_1 tipo_1
nombre_campo_2 tipo_2
nombre_campo_n tipo_n
Key(campo_x,...)
)
```

Pongamos ahora como ejemplo la creación una tabla de pedidos:

```
Create Table pedidos
(
id_pedido INT(4) NOT NULL AUTO_INCREMENT,
id_cliente INT(4) NOT NULL,
id_articulo INT(4)NOT NULL,

fecha DATE,
cantidad INT(4),
total INT(4), KEY(id_pedido,id_cliente,id_articulo)

)
```

En este caso creamos los campos id los cuales son considerados de tipo entero de una longitud especificada por el número entre paréntesis. Para `id_pedido` requerimos que dicho campo se incremente automáticamente (`AUTO_INCREMENT`) de una unidad a cada introducción de un nuevo registro para, de esta forma, automatizar su creación. Por otra parte, para evitar un mensaje de error, es necesario requerir que los campos que van a ser definidos como índices no puedan ser nulos (`NOT NULL`).

El campo fecha es almacenado con formato de fecha (**DATE**) para permitir su correcta explotación a partir de las funciones previstas a tal efecto.
Finalmente, definimos los índices enumerándolos entre paréntesis precedidos de la palabra **KEY o INDEX**.
Del mismo modo podríamos crear la tabla de artículos con una sentencia como ésta:

```
Create Table articulos
(
id_articulo INT(4) NOT NULL AUTO_INCREMENT,
titulo VARCHAR(50),
autor VARCHAR(25),
editorial VARCHAR(25),
precio REAL,
KEY(id_articulo)
)
```

En este caso puede verse que los campos alfanuméricos son introducidos de la misma forma que los numéricos. Volvemos a recordar que en tablas que tienen campos comunes es de vital importancia definir estos campos de la misma forma para el buen funcionamiento de la base.

Muchas son las opciones que se ofrecen al generar tablas. No vamos a tratarlas detalladamente pues sale de lo estrictamente práctico.

2.2.- Estructuras de las tablas en SQL

Una **base de datos** en un sistema relacional está compuesta por un conjunto de tablas, que corresponden a las relaciones del modelo relacional.

En la terminología usada en SQL no se alude a las relaciones, del mismo modo que no se usa el término atributo, pero sí la palabra columna, y no se habla de tupla, sino de línea.

2.2.1.- Creación de Tablas Nuevas

```
CREATE TABLE tabla (
campo1 tipo (tamaño) índice1,
campo2 tipo (tamaño) índice2,... ,
índice multicampo , ... )
```

En donde:

Tabla	Es el nombre de la tabla que se va a crear.
campo1/ campo2	Es el nombre del campo o de los campos que se van a crear en la nueva tabla. La nueva tabla debe contener, al menos, un campo.
tipo	Es el tipo de datos de campo en la nueva tabla. (Ver Tipos de Datos)
tamaño	Es el tamaño del campo sólo se aplica para campos de tipo texto.
índice1/ índice2	Es una cláusula CONSTRAINT que define el tipo de índice a crear. Esta cláusula en opcional.
Índice multicampos	Es una cláusula CONSTRAINT que define el tipo de índice multicampos a crear. Un índice multicampo es aquel que está indexado por el contenido de varios campos. Esta cláusula en opcional.

2.2.2.- La cláusula CONSTRAINT

Se utiliza la cláusula CONSTRAINT en las instrucciones ALTER TABLE y CREATE TABLE para crear o eliminar índices. Existen dos sintaxis para esta cláusula dependiendo si desea Crear ó Eliminar un índice de un único campo o si se trata de un campo multiíndice. Si se utiliza el motor de datos de Microsoft, sólo podrá utilizar esta cláusula con las bases de datos propias de dicho motor. Para los índices de campos únicos:

```
CONSTRAINT nombre {PRIMARY KEY | UNIQUE | REFERENCES tabla externa [(campo externo1, campo externo2)]}
```

Para los índices de campos múltiples:

```
CONSTRAINT nombre {PRIMARY KEY (primario1[, primario2 [,...]]) | UNIQUE (único1[, único2 [, ...]]) |

FOREIGN KEY (ref1[, ref2 [,...]]) REFERENCES tabla externa [(campo externo1 ,campo externo2 [,...])]}
```

En donde:

nombre	Es el nombre del índice que se va a crear.
primario n	Es el nombre del campo o de los campos que forman el índice primario.
único n	Es el nombre del campo o de los campos que forman el índice de clave única.
ref n	Es el nombre del campo o de los campos que forman el índice externo (hacen referencia a campos de otra tabla).
tabla externa	Es el nombre de la tabla que contiene el campo o los campos referenciados en refN
campos externos	Es el nombre del campo o de los campos de la tabla externa especificados por ref1, ref2,... , refN

Si se desea crear un índice para un campo cuando se esta utilizando las instrucciones ALTER TABLE o CREATE TABLE la cláusula CONTRAINT debe aparecer inmediatamente después de la especificación del campo indexado.

Si se desea crear un índice con múltiples campos cuando se está utilizando las instrucciones ALTER TABLE o CREATE TABLE la cláusula CONSTRAINT debe aparecer fuera de la cláusula de creación de tabla.

Índice	Descripción
UNIQUE	Genera un índice de clave única. Lo que implica que los registros de la tabla no pueden contener el mismo valor en los campos indexados.
PRIMARY KEY	Genera un índice primario el campo o los campos especificados. Todos los campos de la clave principal deben ser únicos y no nulos, cada tabla sólo puede contener una única clave principal.
FOREIGN KEY	Genera un índice externo (toma como valor del índice campos contenidos en otras tablas). Si la clave principal de la tabla externa consta de más de un campo, se debe utilizar una definición de índice de múltiples campos, listando todos los campos de referencia, el nombre de la tabla externa, y los nombres de los campos referenciados en la tabla externa en el mismo orden que los campos de referencia listados. Si los campos referenciados son la clave principal de la tabla externa, no tiene que especificar los campos referenciados, predeterminado por valor, el motor Jet se comporta como si la clave principal de la tabla externa estuviera formada por los campos referenciados.

2.2.3.- Creación de Índices

La sintaxis para crear un índice en una tabla ya definida en la siguiente:

```
CREATE [ UNIQUE ] INDEX índice

ON Tabla (campo [ASC|DESC][, campo [ASC|DESC], ...])

[WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]
```

En donde:

índice	Es el nombre del índice a crear.
tabla	Es el nombre de una tabla existente en la que se creará el índice.
campo	Es el nombre del campo o lista de campos que constituyen el índice.
ASC|DESC	Indica el orden de los valores de los campos ASC indica un orden ascendente (valor predeterminado) y DESC un orden descendente.
UNIQUE	Indica que el índice no puede contener valores duplicados.
DISALLOW NULL	Prohibe valores nulos en el índice
IGNORE NULL	Excluye del índice los valores nulos incluidos en los campos que lo componen.
PRIMARY	Asigna al índice la categoría de clave principal, en cada tabla sólo puede existir un único índice que sea "Clave Principal". Si un índice es clave principal implica que no puede contener valores nulos ni duplicados.

2.2.4.- Modificar el Diseño de una Tabla

Modifica el diseño de una tabla ya existente, se pueden modificar los campos o los índices existentes. Su sintaxis es:

```
ALTER TABLE tabla {ADD {COLUMN tipo de campo[(tamaño)] [CONSTRAINT índice]

CONSTRAINT índice multicampo} |

DROP {COLUMN campo I CONSTRAINT nombre del índice}}
```

En donde:

tabla	Es el nombre de la tabla que se desea modificar.
campo	Es el nombre del campo que se va a añadir o eliminar.
tipo	Es el tipo de campo que se va a añadir.
tamaño	Es el tamaño del campo que se va a añadir (sólo para campos de texto).
índice	Es el nombre del índice del campo (cuando se crean campos) o el nombre del índice de la tabla que se desea eliminar.
Índice multicampo	Es el nombre del índice del campo multicampo (cuando se crean campos) o el nombre del índice de la tabla que se desea eliminar.

Operación	Descripción
ADD COLUMN	Se utiliza para añadir un nuevo campo a la tabla, indicando el nombre, el tipo de campo y opcionalmente el tamaño (para campos de tipo texto).
ADD	Se utiliza para agregar un índice de multicampos o de un único campo.
DROP COLUMN	Se utiliza para borrar un campo. Se especifica únicamente el nombre del campo.
DROP	Se utiliza para eliminar un índice. Se especifica únicamente el nombre del índice a continuación de la palabra reservada CONSTRAINT.

2.3.- Añadir un nuevo registro

Los registros pueden ser introducidos a partir de sentencias que emplean la instrucción Insert.

La sintaxis utilizada es la siguiente:

```
Insert Into nombre_tabla (nombre_campo1, nombre_campo2,...) Values
(valor_campo1, valor_campo2...)
```

Un ejemplo sencillo a partir de nuestra tabla modelo es la introducción de un nuevo cliente lo cual se haría con una instrucción de este tipo:

```
Insert Into clientes (nombre, apellidos, direccion, poblacion, codigopostal,
email, pedidos) Values ('Perico', 'Palotes', 'Percebe nº13', 'Lepe',
'123456', 'perico@empresa.com', 33)
```

Como puede verse, los campos no numéricos o booleanos van delimitados por ‘ ’.
También resulta interesante ver que el código postal lo hemos guardado como un campo no numérico. Esto es debido a que en determinados países los códigos postales contienen también letras.

Nota: Si deseamos practicar con una base de datos que está vacía primero debemos crear las tablas que vamos a llenar. Las tablas también se crean con sentencias SQL.
Aunque, de todos modos, puede que sea más cómodo utilizar un programa con interfaz gráfica,

Por supuesto, no es imprescindible rellenar todos los campos del registro. Eso sí, puede ser que determinados campos sean necesarios. Estos campos necesarios pueden ser definidos cuando construimos nuestra tabla mediante la base de datos.

Nota: Si no insertamos uno de los campos en la base de datos se inicializará con el valor por defecto que hayamos definido a la hora de crear la tabla. Si no hay valor por defecto, probablemente se inicialice como NULL (vacío), en caso de que este campo permita valores nulos. Si ese campo no permite valores nulos (eso se define también al crear la tabla) lo más seguro es que la ejecución de la sentencia SQL nos de un error.

Resulta muy interesante, ya veremos más adelante el por qué, el introducir durante la creación de nuestra tabla un campo autoincrementable que nos permita asignar un único número a cada uno de los registros. De este modo, nuestra tabla clientes presentaría para cada registro un número exclusivo del cliente el cual nos será muy útil cuando consultemos varias tablas simultáneamente.

2.4.- Borrar un registro

Para borrar un registro usamos la instrucción **`Delete`**. En este caso debemos especificar cuál o cuáles son los registros que queremos borrar. Es por ello necesario establecer una selección que se llevará a cabo mediante la cláusula **`Where`**.

```
Delete From nombre_tabla Where condiciones_de_selección
```

Nota: Si deseamos practicar con una base de datos que está vacía primero debemos crear las tablas que vamos a llenar. Las tablas también se crean con sentencias SQL.

Si queremos por ejemplo borrar todos los registros de los clientes que se llaman **Perico** lo haríamos del siguiente modo:

```
Delete From clientes Where nombre='Perico'
```

Hay que tener cuidado con esta instrucción ya que si no especificamos una condición con **`Where`**, lo que estamos haciendo es borrar toda la tabla:

```
Delete From clientes
```

2.5.- Actualizar un registro

`Update` es la instrucción que nos sirve para modificar nuestros registros. Como para el caso de `Delete`, necesitamos especificar por medio de Where cuáles son los registros en los que queremos hacer efectivas nuestras modificaciones. Además, obviamente, tendremos que especificar cuáles son los nuevos valores de los campos que deseamos actualizar. La sintaxis es de este tipo:

```
Update nombre_tabla Set nombre_campo1 = valor_campo1, nombre_campo2 =
valor_campo2,...

Where condiciones_de_selección
```

Un ejemplo aplicado:

```
Update clientes Set nombre='José' Where nombre='Pepe'
```

Mediante esta sentencia cambiamos el nombre Pepe por el de José en todos los registros cuyo nombre sea Pepe.
Aquí también hay que ser cuidadoso de no olvidarse de usar Where, de lo contrario, modificaríamos todos los registros de nuestra tabla.