DDL.- LENGUAJE DE DEFINICIÓN DE DATOS

INTRODUCCIÓN

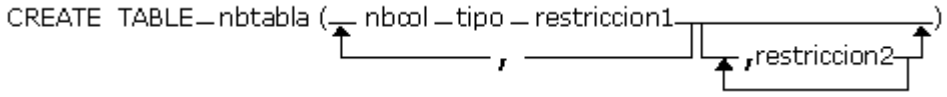
Un lenguaje de definición de datos (Data Definition Language, DDL por sus siglas en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma llevar a cabo las tareas de definición de las estructuras que almacenarán los datos así como de los procedimientos o funciones que permitan consultarlos.

El lenguaje de programación SQL, el más difundido entre los gestores de bases de datos, admite las siguientes sentencias de definición: CREATE, DROP y ALTER, cada una de las cuales se puede aplicar a las tablas, vistas, procedimientos almacenados y triggers de la base de datos.

CREATE TABLE

La sentencia **CREATE TABLE** sirve para **crear la estructura de una tabla** no para rellenarla con datos, nos permite **definir las columnas** que tiene **y ciertas restricciones** que deben cumplir esas columnas.

La **sintaxis** es la siguiente:



nbtabla: nombre de la tabla que estamos definiendo

nbcol: nombre de la columna que estamos definiendo

tipo: **tipo de dato** de la columna, todos los datos almacenados en la columna deberán ser de ese tipo.

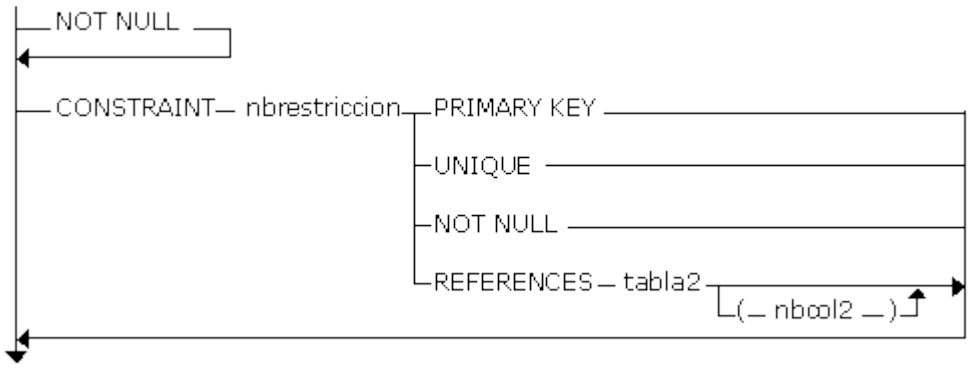
Una **restricción** consiste en la definición de una **característica adicional que tiene una columna** o una combinación de columnas, suelen ser características como valores no nulos (campo requerido), definición de índice sin duplicados, definición de clave principal y definición de clave foránea (clave ajena o externa, campo que sirve para relacionar dos tablas entre sí).

- restricción1: una **restricción de tipo 1** es una restricción que aparece **dentro de la definición de la columna** después del tipo de dato y **afecta** a **una columna**, la que se está definiendo.
- restricción2: una **restricción de tipo 2** es una restricción que se define **después de definir todas las columnas** de la tabla y **afecta** a **una** columna **o** a una **combinación de columnas.**

Para escribir una sentencia **CREATE TABLE** se empieza por indicar el nombre de la tabla que queremos crear y a continuación entre paréntesis indicamos separadas por comas las definiciones de cada columna de la tabla, la definición de una columna consta de su nombre, el tipo de dato que tiene y podemos añadir si queremos una serie de especificaciones que deberán cumplir los datos almacenados en la columna, después de definir cada una de las columnas que compone la tabla se pueden añadir una serie de restricciones, esas restricciones son las mismas que se pueden indicar para cada columna pero ahora pueden afectar a más de una columna por eso tienen una sintaxis ligeramente diferente.

RESTRICCIÓN A NIVEL DE COLUMNA (Restricción tipo 1)

Una **restricción de tipo 1** se utiliza para indicar una característica de la columna que estamos definiendo, tiene la siguiente sintaxis:



La cláusula **NOT NULL** indica que **la columna no podrá contener un valor nulo**, es decir que se deberá rellenar obligatoriamente y con un valor válido (equivale a la propiedad requerido Sí de las propiedades del campo).

La cláusula **CONSTRAINT** sirve para definir una **restricción** que se podrá eliminar cuando queramos sin tener que borrar la columna. A cada restricción se le asigna un nombre que se utiliza para identificarla y para poder eliminarla cuando se quiera.

Como restricciones tenemos la de clave primaria (clave principal), la de índice único (sin duplicados), la de valor no nulo, y la de clave foránea.

La cláusula **PRIMARY KEY** se utiliza para definir la columna como **clave principal de la tabla**. Esto supone que **la columna no puede contener valores nulos ni pueden haber valores duplicados** en esa columna, es decir que dos filas no pueden tener el mismo valor en esa columna.

En una tabla **no pueden haber varias claves principales**, por lo que no podemos incluir la cláusula **PRIMARY KEY** más de una vez, en caso contrario la sentencia da un error. No hay que confundir la definición de varias claves principales con la definición de una clave principal compuesta por varias columnas, esto último sí está permitido y se define con una restricción de tipo 2.

La cláusula **UNIQUE** sirve para definir un **índice único** sobre la columna. Un índice único es un índice que **no permite valores duplicados**, es decir que si una columna tiene definida un restricción de **UNIQUE** no podrán haber dos filas con el mismo valor en esa columna. Se suele emplear para que el sistema compruebe el mismo que no se añaden valores que ya existen, por ejemplo si en una tabla de clientes queremos asegurarnos que dos clientes no puedan tener el mismo D.N.I. y la tabla tiene como clave principal un código de cliente, definiremos la columna dni con la restricción de **UNIQUE**.

La cláusula **NOT NULL** indica que la columna no puede contener valores nulos, cuando queremos indicar que una columna no puede contener el valor nulo lo podemos hacer sin poner la cláusula **CONSTRAINT**, o utilizando una cláusula **CONSTRAINT**.

Una **clave foránea es una columna** o conjunto de columnas **que contiene un valor que hace referencia a una fila de otra tabla**, en una restricción de tipo 1 se puede definir con la cláusula **REFERENCES**, después de la palabra reservada indicamos a qué tabla hace referencia, opcionalmente podemos indicar entre paréntesis el nombre de la columna donde tiene que buscar el valor de referencia, por defecto coge la clave principal de la tabla2, si el valor que tiene que buscar se encuentra en otra columna de tabla2, entonces debemos indicar el nombre de esta columna entre paréntesis, además sólo podemos utilizar una columna que esté definida con una restricción de **UNIQUE**, si la columna2 que indicamos no está definida sin duplicados, la sentencia **CREATE** nos dará un error.

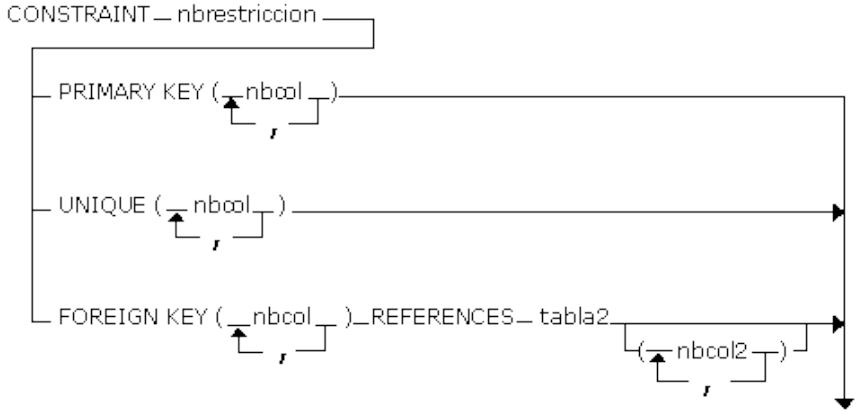
Ejemplo:

```
CREATE TABLE tab1 (
    col1 INTEGER CONSTRAINT pk PRIMARY KEY,
    col2 CHAR(25) NOT NULL,
    col3 CHAR(10) CONSTRAINT uni1 UNIQUE,
    col4 INTEGER Che,
    col5 INT CONSTRAINT fk5 REFERENCES tab2 );
```

Con este ejemplo estamos creando la tabla *tab1* compuesta por: una columna llamada *col1* de tipo entero definida como clave principal, una columna *col2* que puede almacenar hasta 25 caracteres alfanuméricos y no puede contener valores nulos, una columna *col3* de hasta 10 caracteres que no podrá contener valores repetidos, una columna *col4* de tipo entero sin ninguna restricción, y una columna *col5* de tipo entero clave foránea que hace referencia a valores de la clave principal de la tabla *tab2*.

RESTRICCIÓN A NIVEL DEL TABLA (Restricción tipo 2)

Una **restricción de tipo 2** se utiliza **para definir una característica que afecta a una columna o a una combinación de columnas** de la tabla que estamos definiendo, **se escribe después de haber definido todas las columnas** de la tabla.
Tiene la siguiente **sintaxis**:



La sintaxis de una restricción de tipo 2 es muy similar a la **CONSTRAINT** de una restricción 1 la diferencia es que ahora tenemos que indicar sobre qué columnas queremos definir la restricción. Se utilizan obligatoriamente las restricciones de tipo 2 cuando la restricción afecta a un grupo de columnas o cuando queremos definir más de una **CONSTRAINT** para una columna (sólo se puede definir una restricción1 en cada columna).

La cláusula **PRIMARY KEY** se utiliza para definir la **clave principal** de

la tabla. Después de las palabras **PRIMARY KEY** se indica entre paréntesis el nombre de la columna o las columnas que forman la clave principal. Las columnas que forman la clave principal no pueden contener valores nulos ni pueden haber valores duplicados de la combinación de columnas, por ejemplo la tabla pedidos de nuestros ejemplos tiene una clave principal formada por idfab e idproducto, pues no pueden haber dos filas con la misma combinación de idfab con idproducto (*aci,0001* por ejemplo) pero sí pueden haber dos filas con el valor *aci* en la columna *idfab* si tienen valores diferentes en la columna *idproducto*, y pueden haber dos filas con el mismo idproducto pero distinto idfab.

En una tabla **no puede haber varias claves principales**, por lo que no podemos indicar la cláusula **PRIMARY KEY** más de una vez, en caso contrario la sentencia da un error.

La cláusula **UNIQUE** sirve para definir un **índice único** sobre una columna o sobre una combinación de columnas. Un índice único es un índice que **no permite valores duplicados**. Si el índice es sobre varias columnas no se puede repetir la misma combinación de valores en dos o más filas. Se suele emplear para que el sistema compruebe el mismo que no se añaden valores que ya existen.

La cláusula **FOREIGN KEY** sirve para definir una **clave foránea** sobre una columna o una combinación de columnas. Una clave foránea es una columna o conjunto de columnas que **contiene un valor que hace referencia a una fila de otra tabla**, en una restricción 1 se puede definir con la cláusula **REFERENCES**. Para definir una clave foránea en una restricción de tipo 2 debemos empezar por las palabras **FOREIGN KEY** después indicamos entre paréntesis la/s columna/s que es clave foránea, a continuación la palabra reservada **REFERENCES** seguida del nombre de la tabla a la que hace referencia, opcionalmente podemos indicar entre paréntesis el nombre de la/s columna/s donde tiene que buscar el valor de referencia, por defecto coge la clave principal de la tabla2, si el valor que tiene que buscar se encuentra en otra/s columna/s de tabla2, entonces debemos escribir el nombre de esta/s columna/s entre paréntesis, además sólo podemos utilizar una columna (o combinación de columnas) que esté definida con una restricción de **UNIQUE**, de lo contrario la sentencia **CREATE TABLE** nos dará un error.

Las **restricciones CHECK** exigen la integridad del dominio mediante la limitación de los valores que puede aceptar una columna. Son similares a las restricciones FOREIGN KEY porque controlan los valores que se colocan en una columna.

La diferencia reside en la forma en que determinan qué valores son válidos: las restricciones FOREIGN KEY obtienen la lista de valores válidos de otra tabla, mientras que las restricciones CHECK determinan los valores válidos a partir de una expresión lógica que no se basa en datos de otra columna. P

Por ejemplo, es posible limitar el intervalo de valores para una columna salary creando una restricción CHECK que sólo permita datos entre 15.000 y 100.000 dólares. De este modo se impide que se escriban salarios superiores al intervalo de salario normal.

Puede crear una restricción CHECK con cualquier expresión lógica (booleana) que devuelva TRUE (verdadero) o FALSE (falso) basándose en operadores lógicos.

Ejemplo:

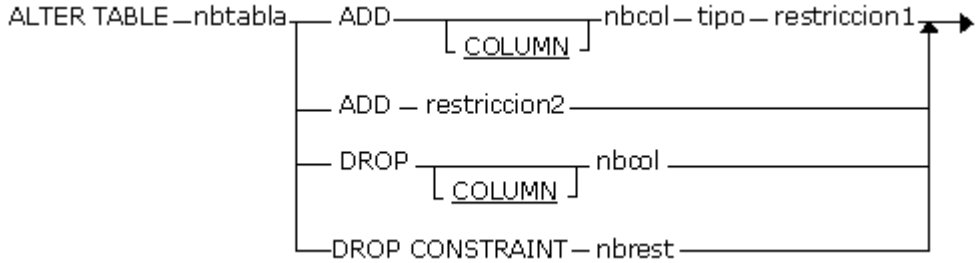
```
CREATE TABLE tab1 (col1 INTEGER,
col2 CHAR(25) NOT NULL,
col3 CHAR(10),
col4 INTEGER,
col5 INT,
CONSTRAINT pk PRIMARY KEY (col1),
CONSTRAINT uni1 UNIQUE (col3),
CONSTRAINT fk5 FOREIGN KEY (col5) REFERENCES tab2 );
```

Con este ejemplo estamos creando la misma tabla *tab1* del ejemplo de la página anterior pero ahora hemos definido las restricciones utilizando restricciones de tipo 2.

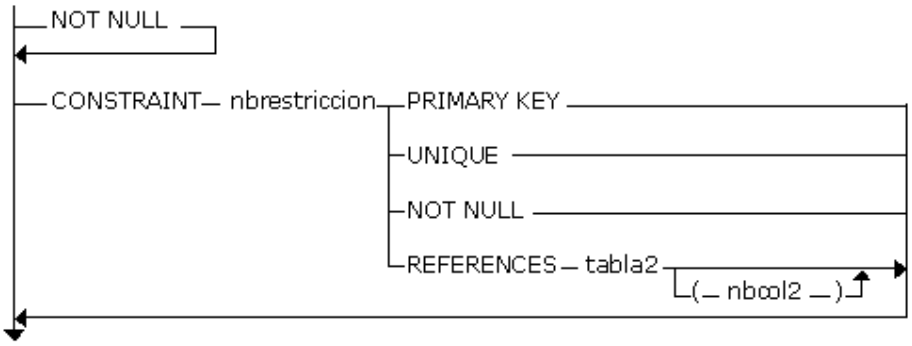
ALTER TABLE

La sentencia **ALTER TABLE** sirve para **modificar la estructura de una tabla** que ya existe. Mediante esta instrucción podemos añadir columnas nuevas, eliminar columnas. Ten cuenta que cuando eliminamos una columna se pierden todos los datos almacenados en ella.

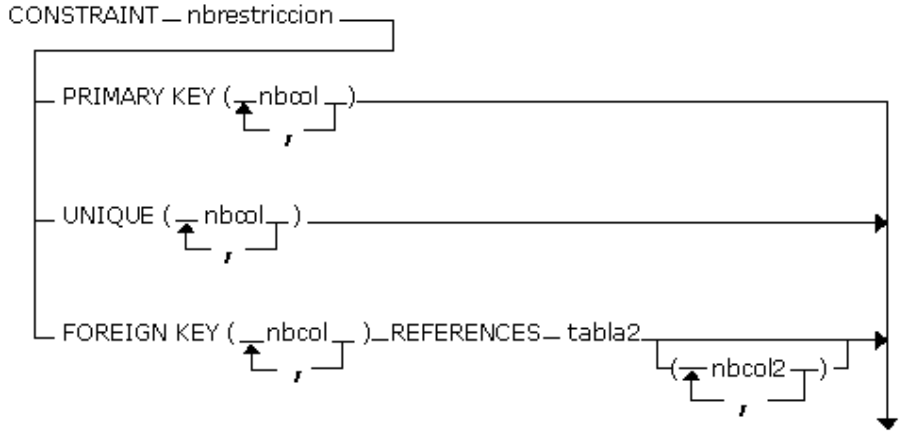
También nos permite crear nuevas restricciones o borrar algunas existentes. La sintaxis puede parecer algo complicada pero sabiendo el significado de las palabras reservadas la sentencia se aclara bastante; ADD (añade), ALTER (modifica), DROP (elimina), COLUMN (columna), CONSTRAINT (restricción).
La **sintaxis** es la siguiente:



La **sintaxis** de **restriccion1** es idéntica a la restricción1 de la sentencia **CREATE TABLE**, te la describimos a continuación.



La **sintaxis** de **restriccion2** es idéntica a la restricción2 de la sentencia **CREATE TABLE**, te la describimos a continuación, si tienes alguna duda repasa la sentencia **CREATE TABLE**.



La cláusula **ADD COLUMN** (la palabra **COLUMN** es opcional) permite

añadir una columna nueva a la tabla. Como en la creación de tabla, hay que definir la columna indicando su nombre, tipo de datos que puede contener, y si lo queremos alguna restricción de valor no nulo, clave primaria, clave foránea, e índice único, **restriccion1 es opcional** e indica una restricción de tipo 1 que afecta a la columna que estamos definiendo.

Ejemplo:

ALTER TABLE tab1 ADD COLUMN col3 integer NOT NULL CONSTRAINT c1 UNIQUE

Con este ejemplo estamos añadiendo a la tabla tab1 una columna llamada col3 de tipo entero, requerida (no admite nulos) y con un índice sin duplicados llamado c1.

Cuando añadimos una columna lo mínimo que se puede poner sería:

ALTER TABLE tab1 ADD col3 integer

En este caso la nueva columna admite valores nulos y duplicados.

AÑADIR UNA NUEVA RESTRICCIÓN

Podemos utilizar la cláusula **ADD restriccion2** (**ADD CONSTRAINT.**..).

Ejemplo:

ALTER TABLE tab1 ADD CONSTRAINT c1 UNIQUE (col3)

Con este ejemplo estamos añadiendo a la tabla tab1 un índice único (sin duplicados) llamado c1 sobre la columna col3.

BORRAR UNA COLUMNA

Basta con utilizar la cláusula **DROP COLUMN** (**COLUMN** es opcional) y el nombre de la columna que queremos borrar, se perderán todos los datos almacenados en la columna.

Ejemplo:

ALTER TABLE tab1 DROP COLUMN col3

También podemos escribir:

ALTER TABLE tab1 DROP col3

El resultado es el mismo, la columna col3 desaparece de la tabla tab1.

BORRAR UNA RESTRICCIÓN

Basta con utilizar la cláusula **DROP CONSTRAINT** y el nombre de la restricción que queremos borrar, en este caso sólo se elimina la definición de la restricción pero los datos almacenados no se modifican ni se pierden.

Ejemplo:

ALTER TABLE tab1 DROP CONSTRAINT c1

Con esta sentencia borramos el índice c1 creado anteriormente pero los datos de la columna col3 no se ven afectados por el cambio.

MODIFICAR UNA COLUMNA

Podemos modificar el tipo de datos de una columna con una simple consulta:

ALTER TABLE <Nombre_tabla> MODIFY <columna> <tipo_dato>

DROP TABLE

La sentencia **DROP TABLE** sirve para **eliminar una tabla**. No se puede eliminar una tabla si está abierta, tampoco la podemos eliminar si el borrado infringe las reglas de integridad referencial (si interviene como tabla padre en una relación y tiene registros relacionados).

La **sintaxis** es la siguiente:

```
DROP TABLE – nbtabla
```

Ejemplo:

DROP TABLE tab1

Elimina de la base de datos la tabla tab1.