4 de marzo 2020

<http://mialtoweb.es/guia-rapida-de-sql-create-table/>

DDL:

* CREATE
* ALTER
* DROP
* TRUNCATE
* RENAME

DML

* SELECT
* UPDATE
* INSERTE
* DELETE

DCL:

* GRANT: Asignar permiso a los objetos
* REVOKE: Quitar permiso a los objetos

TCL: Lenguaje de Control de Transacciones

* COMMIT
* ROLLBACK

**CREATE TABLE : Para crear Tablas**

La sintaxis de la consulta es:

CREATE TABLE nombre\_tabla (

columna\_1 dato(tamaño),

columna\_2 dato(tamaño),

columna\_3 dato(tamaño),

… )

Donde:

nombre\_tabla: indica el nombre que vamos a darle a la tabla

dato: indica el tipo de dato que tendrá la columna

tamaño: le dice a sql que tamaño tendrá el dato que se introduce en el campo

De este modo podemos ver como ejemplo la creación de una tabla llamada personas:

CREATE TABLE personas (

personaId int,

apellido varchar(255),

nombre varchar(255),

direccion varchar(255),

ciudad varchar(255)

)

**Restricciones: CONTRAINS**

Con esto nos referimos a la utilización de reglas especificas para los datos de una tabla. De este modo si ocurre un error de compatibilidad entre el dato y el constraint, la acción se detiene debido al constraint.

Estas reglas se utilizan para controlar algunos de los datos que se utilizan en sql, y se introducen en la creación cuando creamos tablas de la siguiente forma:

CREATE TABLE nombre\_tabla (

columna\_1 dato(tamaño) constraint,

columna\_2 dato(tamaño) constraint,

columna\_3 dato(tamaño) constraint,

… );

Los tipos de constraints son:

NOT NULL: indica que la columna no puede tener el valor NULL

UNIQUE: confirma que cada fila de la columna debe de tener un valor único

PRIMARY KEY: es una combinación entre NOT NULL y UNIQUE, creando una identidad a cada fila para que sea muy fácil y rápido obtener el valor de solo esa fila por medio de su valor identificador.

FOREIGN KEY: Asegura que los datos de la tabla coincidan con los valores de referencia de otra

CHECK: Asegura que el valor de la columna cumple una condición especifica

DEFAULT: Determina un valor por defecto cuando no se especifica un valor para esta columna.

**NOT NULL**

Como ya hemos dicho NOT NULL se utiliza para indicar que el valor de un campo no acepte el valor NULL, es decir que no se le introduzcan valores al crear la tupla.

Siguiendo con nuestro ejemplo vamos a hacer que tanto la id como el apellido no puedan estar en blanco, por lo que haríamos los siguiente:

CREATE TABLE personas (

personaId int NOT NULL,

apellido varchar(255) NOT NULL,

nombre varchar(255),

direccion varchar(255),

ciudad varchar(255)

);

**UNIQUE**

Vamos a hacer que nuestra id sea única, aunque según nuestro sistema de BBDD será de forma diferente de este modo:

CREATE TABLE personas (

personaId int NOT NULL UNIQUE,

apellido varchar(255) NOT NULL,

nombre varchar(255),

direccion varchar(255),

ciudad varchar(255)

);

CREATE TABLE personas (

personaId int NOT NULL,

apellido varchar(255) NOT NULL,

nombre varchar(255),

direccion varchar(255),

ciudad varchar(255),

CONSTRAINT uc\_personaID UNIQUE (personaId, apellido));

Utilizando ALTER TABLE

Si ya tenemos la tabla creada y queremos hacer que uno de sus campos sea único, debemos de usar la siguiente consulta:

ALTER TABLE personas ADD UNIQUE (personaID)

Y si queremos que sean varios campos usamos:

ALTER TABLE personas ADD CONSTRAINT uc\_personaID UNIQUE (personaID, apellido)

Utilizando DROP

Si queremos eliminar la condición de que un campo sea unico depende de nuestra base de datos, de modo que vemos dos versiones:

ALTER TABLE personas DROP CONSTRAINT uc\_personaID

**PRIMARY KEY**

Del mismo modo que con UNIQUE, la definición de este constraint depende del sistema de BBDD donde nos encontremos.

De hecho por defecto PRIMARY KEY contiene la restricción de UNIQUE y NOT NULL.

Una tabla solo puede tener como máximo una columna que hace de clave primaria, y aunque hay tablas donde pueden no ser necesarias, cuando diseño una BBDD me gusta que todas las tablas tengan un campo que sirve como primary key, aunque sea el campo id.

Al igual que sucede con UNIQUE, su definición depende de la BBDD que estemos utilizando, de ese modo debemos hacerlo según nuestro sistema.

CREATE TABLE personas (

personaId int NOT NULL PRIMARY KEY,

apellido varchar(255) NOT NULL,

nombre varchar(255),

direccion varchar(255),

ciudad varchar(255)

);

Utilizando ALTER TABLE

Si ya tenemos la tabla creada y queremos hacer que uno de sus campos sea único, debemos de usar la siguiente consulta:

ALTER TABLE personas ADD PRYMARY (personaID)

Utilizando DROP

Si queremos eliminar la condición de que un campo sea primario depende de nuestra base de datos, de modo que vemos dos versiones:

ALTER TABLE personas DROP CONSTRAINT personaID

**Foreign Key**

Las claves foráneas (foreign key) se utilizan como puntos de unión entre tablas, ya que estos campos hacen referencia a la ID de otras tablas, pudiendo facilitar búsquedas, y consultas.

Como hemos visto en los apartados anteriores es diferente segun la BBDD con la que estemos trabajando, de modo que:

CREATE TABLE Orders

(

O\_Id int NOT NULL PRIMARY KEY,

OrderNo int NOT NULL,

P\_Id int FOREIGN KEY REFERENCES Persons(P\_Id)

);

Utilizando ALTER TABLE

Si la tabla ya esta creada y queremos introducir una clave foránea lo hacemos de la siguiente manera:

ALTER TABLE Orders

ADD FOREIGN KEY (P\_Id)

REFERENCES Persons(P\_Id)

Y si son varias claves lo que queremos introducir:

ALTER TABLE Orders

ADD CONSTRAINT fk\_PerOrders

FOREIGN KEY (P\_Id)

REFERENCES Persons(P\_Id)

Utilizando DROP

Nuevamente depende de nuestra BBDD

ALTER TABLE Orders

DROP CONSTRAINT fk\_PerOrders

**CHECK**

La restricción CHECK se utiliza para limitar el rango de valores que se pueden colocar en una columnas.

Como en todas estas limitaciones, depende la BBDD que usemos el como se debe de escribir su sintaxis:

CREATE TABLE personas (

personaId int NOT NULL,

apellido varchar(255) NOT NULL,

nombre varchar(255),

direccion varchar(255),

ciudad varchar(255),

CONSTRAINT chk\_persona CHECK (personaId>0));

5 de marzo 2020

ALTER TABLE

Modifica laa estructura de una tabla contemplando: Columnas, tipo de datos de una columna, valores por defecto, restricciones, relaciones, tigger, usuario dueño, permisos, comentarios.

Sintaxis:

ALTER TABLE table\_name action;