

REPORTE DE CONcEPTOS

ELABORADO POR JAVIER PEREZ ZAPATA 22/05/2017

A continuación se presentaran algunos conceptos relacionados a base de datos.

Contenido

[CONCEPTO DE CHECK 2](#_Toc483246577)

# CONCEPTO DE CHECK

La palabra check quiere decir “reguistrar” o “reguistro”. Un check es una restricción, una limitación que deben cumplir los datos para que sean considerados válidos y puedan ser grabados.

Por ejemplo si los datos en una base de datos son negativos se debe colocar una check para evitar ingresar datos negativos. Y asi establecer un rango de numeros naturales positivos y de estaforma se evita el uso de datos no aceptados.

Despues de haber creado una table con los datos establecidos , podemos agregar todos los check que necesitemos . Se usa la siguiente sintaxis.

ALTER TABLE MiTabla ADD CONSTRAINT NombreCheck CHECK (MiCondición);

Ejemplo 1. Evitar precios de costo negativos:

ALTER TABLE

PRODUCTOS

ADD CONSTRAINT

CHK\_PRODUCTOS1

CHECK (PRD\_PRECTO >= 0);

Ejemplo 2. Evitar que las notas de los alumnos estén fuera de rango:

ALTER TABLE

EXAMENES

ADD CONSTRAINT

CHK\_EXAMENES1

CHECK(EXA\_NOTAXX >= 0 AND EXA\_NOTAXX <= 100);

**Conclusion**

La restriccion check es muy util para evitar que alguuien ingrese basura o datos innecesarios a nuestra base de datos . si la usamos bien entonces no tenemos que depender de programas de escritorio virtual , simplemente si el dato no cumple con la condicion no se almacenara y sera guardado.

# CONCEPTO DE UNIQUE

crear una restricción UNIQUE en SQL Server 2016 mediante SQL Server Management Studio o Transact-SQL se usa para asegurarse de que no se escribirán valores duplicados en columnas específicas que no participan en una clave principal. Crear una restricción UNIQUE crea automáticamente un índice único correspondiente.

Existen diferentes formas de sistaxis segun el Sistema de base de datos utilizado.

Ejemplo SQL UNIQUE para la base de datos MySQL:

CREATE TABLE personas

{

identificador int NOT NULL,

nombre varchar(255) NOT NULL,

apellido1 varchar(255) NOT NULL,

apellido2 varchar(255),

UNIQUE (identificador)

}

Si intentamos inserter una fila con un identificador que ya exista, nos dara un error, y no nos dejara insertarlo .

Ejemplo SQL UNIQUE para las bases de datos ORACLE, SQLSERVIR, ACCESS

CREATE TABLE personas

{

identificador int NOT NULL UNIQUE,

nombre varchar(255) NOT NULL,

apellido1 varchar(255) NOT NULL,

apellido2 varchar(255),

}

CONCLUCION

El uso de UNIQUE es para no generar o crear datos duplicados en columos no establecidas con una llave principal, y de estaforma crear un indice unico.

# CONCEPTO DE DEFAULT

Cuando se crea una tabla, se puede crear una definición DEFAULT como parte de la definición de dicha tabla. Si la tabla ya existe, puede agregarle una definición DEFAULT. Cada columna de una tabla puede contener una definición DEFAULT.

Si ya existe una definición DEFAULT, puede modificarla o eliminarla. Por ejemplo, puede modificar el valor que se inserta en una columna cuando no se escribe ningún valor.

Cuando se agrega una definición DEFAULT a una columna existente de una tabla, de manera predeterminada Motor de base de datos únicamente aplica el nuevo valor predeterminado a las nuevas filas de datos que se agregan a la tabla. Los datos existentes insertados con la definición DEFAULT anterior no se modifican. No obstante, cuando agrega una nueva columna a una tabla ya existente, puede especificar que Motor de base de datos inserte en la nueva columna el valor predeterminado (especificado mediante la definición DEFAULT), en vez de un valor NULL, para las filas existentes en la tabla.

Sintaxis

CREATE DEFAULT [ schema\_name . ] default\_name

AS constant\_expression [ ; ]

CONCLUCION

El uso de la condicion default es muy utilizado cuando se llenan datos y se repiten datos, se tiene que tomar medidas para el manejo de este metodo donde se utilizen datos ya reguistrados.

# CONCEPTO PK

La clave primaria, PRIMARY KEY, identifica de manera única cada fila de una tabla.

La columna definida como clave primaria (PRIMARY KEY) debe ser UNIQUE (valor único) y NOT NULL (no puede contener valores nulos).

Cada tabla sólo puede tener una clave primaria (PRIMARY KEY).

Ejemplo PRIMARY KEY , clave primaria en MySQL:

CREATE TABLE personas { identificador int NOT NULL, nombre varchar(255) NOT NULL, apellido1 varchar(255) NOT NULL, PRIMARY KEY (identificador) }

La clave primaria (PRIMARY KEY) puede estar compuesta por varias columnas, por ejemplo por las columnas 'identificador' y 'nombre', entonces se define así:

CREATE TABLE personas { identificador int NOT NULL, nombre varchar(255) NOT NULL, apellido1 varchar(255) NOT NULL, CONSTRAINT pers PRIMARY KEY (identificador, nombre) }

CONCLUCION

El uso de una llave primaria es la manera mas adecuada y unica que identifica cada fila de una table. Donde cada table suele tener una sola llave primaria pero llega a ver casos donde es necesario el uso de dos o mas llaves primarias.

CONCEPTO FK

una clave foránea o clave ajena es una limitación referencial entre dos tablas. La clave foránea identifica una columna o grupo de columnas en una tabla (tabla hija o referendo) que se refiere a una columna o grupo de columnas en otra tabla (tabla maestra o referenciada). Las columnas en la tabla referendo deben ser la clave primaria u otra clave candidata en la tabla referenciada.

CLAVES FOREANAS

Adicionar una clave foránea (FK\_) a un campo ya definido en la tabla. El ejemplo consiste en la creación y adición de las llaves foráneas. Adicionalmante eliminación de las tablas.

**CREATE** **TABLE** Cliente

(

IDCliente INTEGER **primary** **key** **not** **null**,

IDProducto INTEGER **not** **null**, *-- Definido el campo con anterioridad para la llave foranea (llave primaria en Producto)*

Fecha\_suscripcion char(8),

Pedido CHAR(100)

);

**CREATE** **TABLE** Producto

(

IDProducto INTEGER **primary** **key** **not** **null**,

IDCliente INTEGER **not** **null**, *-- Definido el campo con anterioridad para la llave foranea (llave primaria en Cliente)*

Nombre varchar(10),

Descripcion varchar(100)

);

*-- script: Adicionar las llaves foraneas*

*-- sintaxis: ALTER TABLE tablaB ADD CONSTRAINT FK\_nomre\_cualquiera\_unico FOREIGN KEY (campo\_tablaB)*

*--- REFERENCES tablaA(campo\_nombre\_tablaA);*

*-- LLave foranea para Cliente*

**ALTER** **TABLE** cliente **ADD** **CONSTRAINT** FK\_IDproducto\_clt **FOREIGN** **KEY** (IDproducto)

**REFERENCES** producto(IDproducto);

*-- LLave foranea para Producto*

**ALTER** **TABLE** producto **ADD** **CONSTRAINT** FK\_IDCliente\_pro **FOREIGN** **KEY** (IDCliente)

**REFERENCES** Cliente(IDCliente);

*-- Script eliminación de tablas\*\**

*-- \*\*se recomienda borrar en orden inverso al que fuero creadas.*

**DROP** **TABLE** Producto **CASCADE** **CONSTRAINTS**;

**DROP** **TABLE** Cliente **CASCADE** **CONSTRAINTS**;

# CONCEPTO DE FK RESTRINGIDA

Una clave externa (FK) es una columna o combinación de columnas que se utiliza para establecer y exigir un vínculo entre los datos de dos tablas. Puede crear una clave externa mediante la definición de una restricción FOREIGN KEY cuando cree o modifique una tabla.

En una referencia de clave externa, se crea un vínculo entre dos tablas cuando las columnas de una de ellas hacen referencia a las columnas de la otra que contienen el valor de clave principal. Esta columna se convierte en una clave externa para la segunda tabla.

No es necesario que una restricción FOREIGN KEY esté vinculada únicamente a una restricción PRIMARY KEY de otra tabla; también puede definirse para que haga referencia a las columnas de una restricción UNIQUE de otra tabla. Una restricción FOREIGN KEY puede contener valores NULL, pero si alguna columna de una restricción FOREIGN KEY compuesta contiene valores NULL, se omitirá la comprobación de los valores que componen la restricción FOREIGN KEY. Para asegurarse de que todos los valores de la restricción FOREIGN KEY compuesta se comprueben, especifique NOT NULL en todas las columnas que participant.

# CONCEPTO DE FK CASCADA

Una clave externa con la eliminación en cascada significa que si se elimina un registro de la tabla primaria, a continuación, se eliminarán automáticamente los registros correspondientes en la tabla secundaria. Esto se llama una eliminación en cascada en SQL Server.

Una clave externa con la eliminación en cascada puede ser creado utilizando una sentencia CREATE TABLE o una instrucción ALTER TABLE.

La sintaxis para crear una clave externa con la eliminación en cascada utilizando una instrucción tabla en SQL Server (Transact-SQL) CREATE es:

CREATE TABLE child\_table

(

column1 datatype [ NULL | NOT NULL ],

column2 datatype [ NULL | NOT NULL ],

...

CONSTRAINT fk\_name

FOREIGN KEY (child\_col1, child\_col2, ... child\_col\_n)

REFERENCES parent\_table (parent\_col1, parent\_col2, ... parent\_col\_n)

ON DELETE CASCADE

[ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]

);

Se utiliza en conjunción con ON DELETE u ON UPDATE. Esto significa que los datos secundarios se elimina o actualiza ya sea cuando se elimina o actualiza los datos de los padres.

# CONCEPTO DE FK NULL

El nulabilidad de una columna determina si las filas de la tabla puede contener un valor nulo para esa columna. Un valor nulo, o NULL, es diferente de cero (0), o una cadena de caracteres en blanco, de longitud cero, tales como "". NULL significa que no se ha hecho una entrada. La presencia de NULL típicamente implica que el valor es desconocido o indefinido. Por ejemplo, un valor nulo en el SellEndDate la columna de la Production.Product tabla de la base de datos AdventureWorks2008R2 no significa que el elemento no tiene fecha de finalización de venta. NULL significa que la fecha se desconoce o no se ha establecido.

Si se inserta una fila, pero sin valor se incluye para una columna que permite valores nulos, el motor de base proporciona el valor NULL, a menos que exista una definición DEFAULT u objeto. Una columna definida con la palabra clave NULL, también acepta una entrada explícita de NULL por parte del usuario, independientemente de qué tipo de datos que es o si tiene un defecto asociado con él. El valor NULL no se debe añadir entre comillas, ya que se puede interpretar como la cadena de caracteres '' NULL '', en lugar del valor nulo.

Especificación de una columna de que no permite valores nulos puede ayudar a mantener la integridad de datos mediante la garantía de que una columna en una fila siempre contiene datos. Si los valores nulos no están permitidos, un usuario que introduce datos en la tabla debe introducir un valor en la columna o bien la fila de la tabla no puede ser aceptado en la base de datos.

Un valor de NULL indica que el valor es desconocido. Un valor de NULL es diferente de un valor vacío o cero. No hay dos valores nulos son iguales. Las comparaciones entre dos valores nulos, o entre un NULL y cualquier otro valor, regresan desconocido debido a que el valor de cada NULL es desconocida.

# CONCEPTO DE FK DAFAULT

# CONSEPTO DE DOMINIO VARIABLES

| **Tipo de datos de dominio DQS** | **Tipo de dato de SSIS admitido** |
| --- | --- |
| Date | DT\_DATE |
| Decimal | DT\_DECIMAL  DT\_NUMERIC  DT\_R4  DT\_R8 |
| Integer | DT\_I1  DT\_I2  DT\_I4  DT\_I8  DT\_U1  DT\_U2  DT\_U4  DT\_U8 |
| String | DT\_STR  DT\_WSTR |

Hay muchos tipos de datos en SQL Server y SQL Server Integration Services (SSIS), pero solo cuatro tipos de datos para dominios DQS: Date, Decimal, Integer y String. No todos los tipos de datos en SQL Server y SSIS se admiten en DQS. Solo puede asignar los datos de origen para un dominio DQS a fin de realizar actividades de calidad de datos únicamente si el tipo de datos de origen se admiten en DQS y coincide con el tipo de datos de dominio DQS. Este tema proporciona información acerca de los tipos de datos de SSIS y SQL Server que se admiten y están disponibles para asignar a cada uno de los cuatro tipos de dominio en DQS.

Tipos de datos de SQL Server admitidos

En la tabla siguiente se proporciona información acerca de los tipos de datos de SQL Server admitidos para cada tipo de datos de dominio DQS:

Tipo de datos de dominio DQSTipo de datos de SQL Server admitidoDateDateDecimalDecimal  
  
En la tabla siguiente se proporciona información acerca de los tipos de datos de SSIS admitidos para cada tipo de datos de dominio DQS:

# CONCEPTO DE AUTOINCREMENTO

Generalmente el AUTO INCREMENT / AUTO INCREMENTAL se usa para tener una PRIMARY KEY de una tabla mediante la generación automática de un numero secuencial único en la tabla.

La sintaxis del AUTO INCREMENT es muy fácil y solo varia de acuerdo al sistema de base de datos:

MySQL:

CREATE TABLE Usuarios

(

ID int NOT NULL AUTO\_INCREMENT,

NOMBRE varchar(255),

PRIMARY KEY (ID)

);

Por defecto el AUTO INCREMENTAL inicia en 1 y va aumentando de a 1 por cada nuevo registro que se inserta en la tabla.

Si queremos que nuestra tabla el auto incrementar comience con otro valor tenemos que modificar la tabla mediante nuestro comando amigo ALTER TABLE.

# REFERENCIAS

http://java-white-box.blogspot.mx/2014/07/sql-auto-increment-sintaxis-auto.html

<http://java-white-box.blogspot.mx/2014/02/queeselsql.html>

<http://java-white-box.blogspot.mx/2014/07/sql-foreign-key-llave-secundaria.html>

<http://ingsystemas.webnode.es/bases-de-datos/llaves-primarias-y-foraneas/>

<https://technet.microsoft.com/es-es/library/ms175464(v=sql.105).aspx>

https://msdn.microsoft.com/es-mx/library/ms190024.aspx