Teórico 5

Data Definition Language (DDL) (SQL - Segunda Parte)

Tipo de Datos

Los tipos de datos SQL se clasifican en 13 categorías primarias, las cuales pueden tener sinónimos reconocidos por dichos tipos de datos.

Cabe aclarar que dependiendo del manejador de bases de datos, puede haber algunas variaciones en los tipos de datos de los atributos o campos.

En SQL-92, además de los tipos de datos del SQL-89 (INTEGER, SMALLINT, CHARACTER, DECIMAL, NUMERIC, REAL, FLOAT y DOUBLE PRECISION) también se admiten los siguientes: CHARACTER VARYING, DATE, TIME, BIT, TIMESTAMP y BIT.

SQL 99, extiende SQL con características de bases de datos objeto-relacionales.

Tipos de datos de Mysql

TinyInt	Es un número entero con o sin signo. Con signo el rango de valores válidos va desde -128 a 127. Sin signo, el rango de valores es de 0 a 255.
Bit ó Bool	Un número entero que puede ser 0 ó 1.
SmallInt	Número entero con o sin signo. Con signo el rango de valores va desde -32768 a 32767. Sin signo, el rango de valores es de 0 a 65535.
MediumInt	Número entero con o sin signo. Con signo el rango de valores va desde -8.388.608 a 8.388.607. Sin signo el rango va desde 0 a 16777215.
Integer, Int:	Número entero con o sin signo. Con signo el rango de valores va desde -2147483648 a 2147483647. Sin signo el rango va desde 0 a 429.4967.295.
BigInt	Número entero con o sin signo. Con signo el rango de valores va desde -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807. Sin signo el rango va desde 0 a 18.446.744.073.709.551.615.
Float	Número pequeño en coma flotante de precisión simple. Los valores válidos van desde -3.402823466E+38 a -1.175494351E-38, 0 y desde 1.175494351E-38 a 3.402823466E+38 .
xReal, Double	Número en coma flotante de precisión doble. Los valores permitidos van desde -1.7976931348623157E+308 a -2.2250738585072014E-308, 0 y desde 2.2250738585072014E-308 a 1.7976931348623157E+308 .
Decimal, Dec, Numeric	Número en coma flotante desempaquetado. El número se almacena como una cadena.

Date	Tipo fecha, almacena una fecha. El rango de valores va desde el 1 de enero del 1001 al 31 de diciembre de 9999. El formato de almacenamiento es de año-mes-dia.
DateTime	Combinación de fecha y hora. El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. El formato de almacenamiento es de año-mes-dia horas:minutos:segundos.
TimeStamp	Combinación de fecha y hora. El rango va desde el 1 de enero de 1970 al año 2037. El formato de almacenamiento depende del tamaño del campo.
Char(n)	Almacena una cadena de longitud fija. La cadena podrá contener desde 0 a 255 caracteres.
VarChar(n)	Almacena una cadena de longitud variable. La cadena podrá contener desde 0 a 255 caracteres.
TinyText y TinyBlob	Columna con una longitud máxima de 255 bytes (caracteres en text)..
Blob y Text	Columna con un máximo de 65535 bytes (caracteres en text)..
MediumBlob y MediumText	Columna con un máximo de 16.777.215 bytes (caracteres en text).
LongBlob y LongText	Columna con un máximo de caracteres 4.294.967.295 bytes (caracteres en text).

Tipos de datos de **PostgreSQL**

Tipo	**Descripción**
bool	valor lógico o booleano (true/false)
char(n)	Cadena de caracteres de longitud fina n
date	fecha (sin hora)
float4	número de punto flotante
float8	número de punto flotante de doble precisión
int2	entero de dos bytes con signo
int4	entero de cuatro bytes con signo
time	hora en horas, minutos, segundos y centésimas
timestamp	fecha y hora con zonificación
varchar(n)	cadena de caracteres de tamaño máximo n
point	punto geométrico en el plano
polygon	trayectoria geométrica cerrada en el plano
serial	identificador numérico único (autoincrementado)

Create Table

Es el comando fundamental para definir datos, crea una nueva relación (una nueva tabla). La sintaxis del comando CREATE TABLE es:

CREATE TABLE *tabla* (*campo1 tipo* [(*tamaño*)] [NOT NULL] [*índice1*]
[, *campo2 tipo* [(*tamaño*)] [NOT NULL] [*índice2*] [, ...]]
[, *campoN tipo* [(*tamaño*)] [NOT NULL] [*índiceN*] [, ...]]
[, *ligaduraIntegridad1* [, ...]]
[, *ligaduraIntegridadN* [, …]])

Argumento	Descripción
Tabla	Nombre de la tabla a crear.
CampoN	Nombre del campo o de los campos componentes de la nueva tabla. La nueva tabla debe contener, al menos, un campo.
Tipo	Es el tipo de datos del campo en la nueva tabla. Anteriormente se vieron los tipos permitidos de un campo. También puede ser un dominio definido por el usuario.
Tamaño	Tamaño del campo, sólo se aplica para algunos tipos, ej.: varchar.
ÍndiceN	Es una cláusula CONSTRAINT que define el tipo de restricción a crear. Mas adelante se describe esta cláusula.
LigaduraIntegridadN	Es una cláusula CONSTRAINT que define el tipo de restricción a crear.
NOT NULL	Indica que el campo no puede recibir valores nulos.

Ejemplo

```
CREATE TABLE Proveedores
(Nprov  INTEGER NOT NULL CONSTRAINT pkproveedores PRIMARY KEY,
 Nombre VARCHAR(50) NOT NULL CONSTRAINT uniconombre UNIQUE,
 Direccion VARCHAR(50));
```

La ejecución de este comando produce como resultado la creación de una tabla Proveedores con tres campos; Nprov: de tipo entero y clave primaria de la tabla, Nombre: de tipo cadena de caracteres de 50 e índice único o clave secundaria, y por último el campo Dirección de tipo cadena de caracteres de 50.

Ligaduras de Integridad

- Claves primarias (Primary key)
- Claves secundarias o candidatas (Unique)
- Cardinalidades de relaciones (en E/R)
- Ligaduras de dominios (check).
- Integridad referencial (Foreing key)
- Aserción (Assertion)
- Disparadores (Trigger)
- Dependencia funcionales (Tema teórico posterior)

Cláusula CONSTRAINT

• Se utiliza la cláusula CONSTRAINT en las instrucciones ALTER TABLE y CREATE TABLE para crear o eliminar restricciones.

• Existen dos sintaxis para esta cláusula dependiendo si se quiere crear un índice de un único campo o si se trata de un índice multi-campo.

Para los índices de campos únicos:

CONSTRAINT *nombre*
{PRIMARY KEY | UNIQUE | NOT NULL | REFERENCES
tablaexterna [(*campoexterno1*]}

Para los índices de múltiples campos:

CONSTRAINT *nombre*
{
PRIMARY KEY (*primario1*[,*primario2* [, ...]])
| UNIQUE (*único1*[,*único2* [, ...]])
| NOT NULL (*nonulo1* [, *nonulo2* [, ...]])
| CHECK *condicion*
| FOREIGN KEY (*referencia1* [, *referencia2* [, ...]])
REFERENCES *tablaexterna [*(*campoexterno1*
[,*campoexterno2* [, ...]])]
[ON DELETE accion][ON UPDATE accion]
}

Tipo de Índice	Descripción
PRIMARY KEY	Genera un índice primario el campo o los campos especificados. Todos los campos de la clave primaria deben ser únicos y no nulos, cada tabla sólo puede contener una única clave primaria.
UNIQUE	Genera un índice de clave única, de esta forma se generan las claves secundarias o candidatas.
CHECK	Es para especificar una condición (condición) que deben cumplir los datos de una tabla.
FOREIGN KEY	Genera un índice externo (toma como valor del índice campos contenidos en otras tablas), de esta forma se incorporan restricciones de **integridad referencial**.

Argumento	Descripción
Nombre	Es el nombre del índice que se va a crear.
PrimarioN	Es el nombre del campo o de los campos que forman el índice primario.
UnicoN	Es el nombre del campo o de los campos que forman el índice de clave única.
NonuloN	Es el nombre del campo o de los campos que no admiten valores nulos.
ReferenciaN	Es el nombre del campo o de los campos que forman el índice externo (hacen referencia a campos de otra tabla). Se utiliza para introducir restricciones de integridad referencial.
Tabla externa	Es el nombre de la tabla que contiene el campo o los campos referenciados en refN.
CampoexternoN	Es el nombre del campo o de los campos de la tabla externa especificados por ref1, ref2, ..., refN.
Accion	Puede ser: Cascade, set null, set Default y restricted o no action

Ejemplos

1)

```
CREATE TABLE articulos (
Nart INTEGER NOT NULL PRIMARY KEY,
Descr varchar(100), Precio FLOAT, Cant INTEGER,
Stock_min INTEGER, Stock_max INTEGER,
CONSTRAINT control_min_max CHECK (stock_min<stock_max),
CONSTRAINT precio_positivo CHECK (precio>0));
```

La ejecución de este comando produce como resultado la creación de la tabla artículos con seis campos; Nart de tipo entero y clave primaria de la tabla, descr: de tipo cadena de caracteres de 100, precio de tipo FLOAT y por último los campos cant, stock_min y stock_max de tipo INTEGER. Además se agregan dos restricciones CHECK, una para controlar que el stock_min sea menor que el stock_max y la otra para controlar que el precio sea positivo.

Ejemplos (cont.)

2)

```
CREATE TABLE PROVEE
(Nprov  INTEGER NOT NULL,
 Nart  INTEGER NOT NULL,
 Precio_venta FLOAT,
 Constraint pkprovee primary key (Nart,Nprov),
 Constraint fkproveedores foreign key (Nprov) references proveedores,
 Constraint fkarticulos foreign key (Nart)
                references articulos on delete cascade)
```

Este comando crea la tabla PROVEE con los campos Nart, Nprov y precio_venta, además tenemos como clave primaria a Nart y Nprov. La tabla PROVEE posee además dos claves Foráneas (integridad referencial), una, el atributo Nprov que hace referencia a la tabla PROVEEDORES y la otra, el atributo Nart que hace referencia a la tabla ARTICULOS y el borrado será en cascada.

Alter Table

Comando utilizado para modificar la Estructura de una tabla existente, su sintaxis es:

ALTER TABLE tabla
{
 ADD {COLUMN *campo* *tipo* [(*tamaño*)] [NOT NULL]
     | *clausulaConstraint*}
     | DROP {COLUMN *campo* | CONSTRAINT *clausulaEliminar*}
}

Argumento	**Descripción**
Tabla	Es el nombre de la tabla que se desea modificar.
Campo	Es el nombre del campo que se va a agregar o eliminar.
Tipo	Tipo del campo que se va a agregar.
Tamaño	Tamaño del campo que se va a agregar, depende del tipo de dato.
clausulaConstraint	Es una cláusula CONSTRAINT que define el tipo de restricción a crear.
clausulaEliminar	Nombre de la cláusula constraint a eliminar.

Ejemplos

1) ALTER TABLE Proveedores ADD Telefono VARCHAR (20) NOT NULL;

La ejecución de este comando provoca que la tabla Proveedores, ya existente, incorpore una nueva columna llamada “Telefono” de tipo cadena de caracteres de 20 de longitud máxima.

2) ALTER TABLE Proveedores DROP Direccion;

Elimina la columna Dirección de la tabla Proveedores

3) ALTER TABLE Proveedores ADD CONSTRAINT telefonounico UNIQUE Telefono;

Agrega una restricción de Unique para el campo telefono de la tabla Proveedores

4) ALTER TABLE Proveedores DROP CONSTRAINT telefonounico;

Elimina la restricción con nombre telefonounico sobre el campo telefono de la tabla Proveedores

Dominios

- Un dominio es un tipo de datos definido por el usuario en función de un tipo existente. Su sintaxis es:

CREATE DOMAIN nombreDominio [AS] *tipoDatos*

[DEFAULT valorPorDefecto]

[NOT NULL] [CHECK (condición)];

Donde :

Argumento	**Descripción**
nombreDominio	Nombre del nuevo dominio, el cual se hará referencia en la definición.
tipoDatos	Es un tipo de datos de SQL.
valorPorDefecto	Es el valor por defecto que tendrán las columna de este dominio
condición	Condición que deben cumplir los valores del dominio.

Ejemplo

```
CREATE DOMAIN enteroPositivo AS INTEGER
        DEFAULT 2
        CHECK (value>1)
                NOT NULL;
```

Esta instrucción crea un dominio de valores enteros mayores que 1, no acepta valores nulos y tiene por defecto el valor 2.

Ejemplo de Utilización

```
CREATE TABLE PROVEE
( Nprov enteroPositivo,
  Nart enteroPositivo,
 Precio_venta FLOAT,
 Constraint pkprovee primary key (Nart,Nprov),
 Constraint fkproveedores foreign key (Nprov) references proveedores,
 Constraint fkarticulos foreign key (Nart) references articulos)
```

Este comando crea la tabla PROVEE utilizando el dominio definido para los campos Nart, Nprov.

Índices

- Un índice esta basado en una o más columnas de una tabla, su función es ordenar el contenido de las columnas especificadas y almacenar esta información ordenada en disco, para tener acceso rápido y eficiente a estas columnas.

- Conforme los registros se anexan, cambian o eliminan, el sistema de administración de base de datos actualiza automáticamente el índice para reflejar los cambios.

Creación de Índices

CREATE [UNIQUE] INDEX *índice* ON *tabla* (*campo1* [ASC|DESC] [, *campo*2[ASC|DESC], ...])
  [WITH { PRIMARY | DISALLOW NULL | IGNORE NULL }]

Argumento	Descripción
Índice	Nombre del índice a crear.
Tabla	Nombre de una tabla existente en la que se creará el índice.
CampoN	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	Prohíbe 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 Primaria.

Ejemplo

CREATE INDEX IndiceTelefono ON Proveedores
(Telefono);

Crea un índice llamado IndiceTelefono en la tabla proveedores con el campo Telefono.

Vistas

- Son relaciones que no forman parte del modelo lógico de la base de datos que son visibles a los usuarios como tablas virtuales.
- Son utilizadas por razones de seguridad, para ocultar cierta información a los usuarios.

Sintaxis

Create view <nombre_vista> as <expresion_de_consulta>

Ejemplo:

CREATE VEW nombre_proveedores AS

SELECT nombre FROM proveedores;

Borrado de Estructuras

El comando DROP elimina una tabla existente de una base de datos o elimina un índice existente de una tabla. Su sintaxis es:

DROP {TABLE *tabla* | VIEW *vista*| INDEX *índice* ON *tabla*}

Donde:

Argumento	**Descripción**
Tabla/Vista	Nombre de la tabla/vista que se va a eliminar o la tabla de la cual se va a eliminar un índice.
Índice	Nombre del índice que se va a eliminar de tabla.

EJEMPLOS

1) DROP TABLE provee;

Elimina de la base de datos la tabla provee.

2)DROP TABLE proveedores;

Elimina de la base de datos la tabla proveedores.

3) DROP INDEX IndiceTelefono ON proveedores;

Elimina el índice IndiceTelefono de la tabla proveedores.

Aserciones(Assert)

- Es un Predicado que define una condición que la base de datos debe cumplir siempre.
- Como el chequeo de estas condiciones puede ser muy costoso, la mayoría de los RDBMS no implementan las aserciones, ej.: Firdbird, MySql y Posgres aún no las implementan.

Ejemplo

- La base de datos no debe aceptar que un artículo tenga como precio de venta, un precio menor que el máximo precio de venta de los proveedores del articulo.

- La aserción que define esta restricción sería:

```
CREATE ASSERTION asercion CHECK (NOT EXISTS
(SELECT * FROM articulos a WHERE
(SELECT MAX(precio_venta) FROM provee p
  WHERE  a.nart = p.nart)
<=
(SELECT precio FROM articulos ar
  WHERE a.nart = ar.nart)))
```

Disparadores(Trigger)

Un disparador es una orden que ejecuta el DBMS automáticamente ante alguna modificación en la base de datos.

Para diseñar un trigger es necesario :

1)Especificar la condición de disparo.

2)Especificar la acciones que se van a ejecutar cuando la condición de disparo de cumpla.

Condiciones de disparo

Un trigger se puede disparar antes o después de, insertar, modificar o eliminar un registro en una tabla.

Sintaxis

•En SQL92 no incluye a los trigger, la definición de un trigger no esta normalizado, por esto es que cada RDBMS tiene su forma no normalizada de definición de trigger.

•La Norma SQL1999 incorpora la sintaxis para los trigger, pero de todas formas los RDBMS mantienen su forma no normalizada de definición de trigger.

Nota: La variable contextual OLD almacena temporariamente los viejos valores del registro en la eliminación o actualización. La variable contextual NEW almacena temporariamente los nuevos valores del registro en la inserción o actualización.

Ejemplo en mysql

A continuación se muestra la definición de un trigger que se dispara para despues de la eliminación de un registro para la tabla articulos, y lo que realiza es la inserción de un nuevo registro en la tabla articulos_bajas con los valores de nart, descr y precio del articulo eliminado de la tabla articulo, además inserta el usuario que realizo la operación y la fecha y hora de la eliminación:

```
delimiter $$
CREATE TRIGGER trigger_baja_articulos
     AFTER  DELETE ON articulos
     FOR EACH ROW
      BEGIN
        INSERT INTO articulos_bajas VALUES (OLD.nart,
            OLD.descr,OLD.precio, CURRENT_USER(), NOW() );
      END;
$$
delimiter ;
```

Ejemplo de trigger en PostgreSQL

// crear la función que se ejecutará en el trigger

```
create function funcion_auditoria () returns trigger as 'begin insert into
    auditoria values(1,old.nart,now(),old.cant-new.cant);
    return new;
    end;'
    LANGUAGE 'plpgsql';
```

// crear el trigger para generar información de auditoria

```
create trigger trigger_auditoria after update on articulos for each row
    execute procedure funcion_auditoria();
```

Algunas Características Objeto-Relacionales de SQL 99

- Tipos complejos.
- Tipos estructurados.
- Herencia de tablas.

Tipos Complejos

Colecciones

- setOf
  - Ej,: create table libro (...,palabra_clave setof(varchar(30)),...)
- Array
  - Ej,: create table libro (...,autores varchar(30) array[10]),...)

Tipos estructurados.

SQL99 declara tipos y su uso de la siguiente forma:

- Declaración:

```
Create type Editor as
  (nombre varchar(30),
   sucursal varchar(30))
```

- Uso:

```
Create table libro (
 titulo varchar(40),
 autores varchar(30) array[5],
 editor Editor)
```

Herencia

Las herencias pueden ser a nivel tipos o nivel tablas.

- Herencias de tipos:

  Create type Persona as (nombre varchar(30),direccion varchar(50));

  Create type Estudiante under Persona as (grado varchar(30));

- Herencia de tablas:

  Create table Persona (nombre varchar(30),direccion varchar(50));

  Create table Estudiante under Persona (grado varchar(30));

  En postgres:

  Create table Estudiante (grado varchar(30)) inherits (Persona) ;