Fundamentos de Bases de Datos.
Práctica 11.

Profesora: Dra. Amparo López Gaona
alg@ciencias.unam.mx
Laboratorio: Lic. Carlos Augusto Escalona Navarro
caen@ciencias.unam.mx

9 de diciembre de 2021

Se dan a conocer especificaciones de entrega para la práctica 11.

1. Restricciones de Integridad

1.1. Restricciones

Las restricciones le permiten definir la manera en que el SMBD define la integridad de los datos en una base de datos. Las restricciones definen reglas relativas a los valores permitidos en las columnas (dependencias funcionales) y como es que se relacionan la tablas entre si, lo cual contribuye al mecanismo estándar para la integridad. El uso de restricciones es preferible por medio de funciones, procedimientos almacenados y disparadores [1]. El optimizador de consultas también utiliza definiciones de restricciones para generar planes de ejecución de consultas con un mejor rendimiento, esto se realiza por medio de la generacion de índices dentro de cada tabla.

2. Integridad en base de datos

Existen 3 tipos de integridad dentro de bases de datos, los cuales se listan a continuación:

1. **Integridad de entidad**.- Establece que la *llave primaria* de una tabla debe tener un valor único para cada registro; en caso contrario, la BD perderá su integridad. Se especifica con la instrucción *CREATE TABLE*. El SMBD comprueba automáticamente la unicidad del valor de la llave primaria con cada instrucción *INSERT* Y *UPDATE* que recibe.

[1] Se tratará con mas detalle en la siguientes prácticas.

PostgreSQL admite las siguientes clases de restricciones:

- *Unique.*- Se usa para definir que el valor de una o varias columnas no se encuentre duplicado. En general, una restricción se viola si hay más de una fila donde los valores de todas las columnas incluidas en la restricción son iguales. Sin embargo, dos valores nulos no se consideran iguales en esta comparación. Eso significa que incluso en presencia de una restricción única, es posible almacenar las filas duplicadas que contienen un valor nulo en al menos una de las columnas limitadas.

- *Primary key.*- Se usa para definir que el valor de una o varias columnas determinen toda la tupla o registro haciéndolos únicos en la tabla. Como se mencionó anteriormente, puede existir cualquier número de restricciones unique y not null, pero sólo se puede definir una llave primaria dentro de la tabla, como lo establece la teoría de BDs. Esta regla no es obligatoria en PostgreSQL, pero es parte de las buenas prácticas de las BDs.

2. **Integridad de referencial**.- asegura la integridad entre las llaves foráneas y primarias. La *llave foránea*, también conocida como llave externa, se refiere a uno o más atributos de un tabla que hacen referencia a los atributos de la llave primaria de otra tabla. La llave foránea, *indica cómo están relacionadas las tablas*. Los datos en los atributos de la llave foránea y la primaria deben coincidir, aunque los nombres de los atributos no sean los mismos.

   PostgreSQL admite las siguientes clases de restricciones:

   - *Foreign Key* .-Se usa para definir el valor de una o varias columnas para hacerlo coincidir con los valores que aparecen en la columna de otra tabla. Una llave foránea debe hacer referencia a las columnas, o bien son una llave primaria o forman una restricción única. Un *DELETE* del registro de la tabla o un **UPDATE** de una columna requerirá un análisis de la tabla de referencia para las columnas que coinciden con el valor antiguo, a menudo es una buena idea para indexar las columnas de referencia también.
     Existen dos parámetros para una llave foránea que son importantes y definen cómo se comporta la BD para salvaguardar la integridad de nuestros datos. Estos parámetros son:

     ON DELETE acción
     ON UPDATE acción
     En donde acción puede tener estos valores:
     NO ACTION: Produce un error por violación de la llave foránea definida.

     RESTRICT: Produce un error por una violación de la llave foránea definida.

CASCADE: Borra o actualiza automáticamente todas las referencias activas.

SET NULL: Define las referencias activas como NULL.
SET DEFAULT: Define las referencias activas con el valor por defecto (si está definido) de las mismas.

3. **Integridad de dominio**.- es especificar para cada atributo de una tabla un dominio de valores posibles. Una definición adecuada de las restricciones de los dominios es que no sólo permite verificar los valores introducidos en la base de datos sino también examinar consultas para asegurarse de que tengan sentido las comparaciones que hagan.

   PostgreSQL admite las siguientes clases de restricciones:

   - NOT NULL especifica que la columna no acepta valores NULL. Un valor NULL no es lo mismo que cero (0), en blanco o que una cadena de caracteres de longitud cero, como . NULL significa que no hay ninguna entrada. La presencia de un valor NULL suele implicar que el valor es desconocido o no está definido. Se recomienda evitar la aceptación de valores NULL, dado que pueden implicar una mayor complejidad en las consultas y actualizaciones. Por otra parte, hay otras opciones para las columnas, como las restricciones PRIMARY KEY, que no pueden utilizarse con columnas que aceptan valores NULL.
   - Las restricciones CHECK exigen la integridad del dominio mediante la limitación de los valores que se pueden asignar a una columna. Una restricción CHECK especifica una condición de búsqueda booleana (se evalúa como TRUE, FALSE o desconocido) que se aplica a todos los valores que se indican en la columna. Se rechazan todos los valores que se evalúan como FALSE. En una misma columna se pueden especificar varias restricciones CHECK.
   - Las restricciones DEFAULT es utilizada para definir un valor por defecto cuando el usuario no pone un valor o pone null en una columna de la tabla.

2.1. Incrementadores

2.1.1. Secuencias

Una secuencia es un objeto enlazado a un esquema definido por el usuario que genera una secuencia de valores numéricos según la especificación con la que se creó la secuencia. La secuencia de valores numéricos se genera en orden ascendente o descendente en un intervalo definido y se puede configurar para reiniciarse (en un ciclo) cuando se agota. Las secuencias, a diferencia de las columnas de identidad, no se asocian a tablas concretas. Las aplicaciones hacen referencia a un objeto de secuencia para recuperar su valor siguiente. La aplicación controla la relación entre las secuencias y tablas. Las aplicaciones de usuario pueden hacer referencia un objeto de secuencia y

coordinar los valores a través de varias filas y tablas.

A diferencia de los valores de las columnas de identidad que se generan cuando se insertan filas, una aplicación puede obtener el número de secuencia siguiente sin insertar la fila mediante una llamada a la función NEXT VALUE FOR.

```
CREATE SEQUENCE sequence_name
  MINVALUE value
  MAXVALUE value
  START WITH value
  INCREMENT BY value
  CACHE value;
```

Los números de secuencia se generan fuera del ámbito de la transacción actual. Se utilizan tanto si la transacción que usa el número de secuencia se confirma como si se revierte.

2.1.2. Serial

En PostgreSQL, una secuencia es un tipo especial de objeto de base de datos que genera una secuencia de enteros. Una secuencia se usa a menudo como la columna de llave primaria en una tabla.

Al crear una nueva tabla, la secuencia se puede crear a través del pseudotipo SERIAL de la siguiente manera:

Al asignar el pseudotipo SERIAL a la columna id, PostgreSQL realiza lo siguiente:

- Primero, se crea un objeto de secuencia y se establece el siguiente valor generado por la secuencia como el valor predeterminado para la columna.
- En segundo lugar, se agrega una restricción NOT NULL a la columna id porque una secuencia siempre genera un número entero, que es un valor no nulo.
- Tercero, se asigna el propietario de la secuencia a la columna de identificación; como resultado, el objeto de secuencia se elimina cuando se cae la columna o tabla de identificación

```
CREATE TABLE post (
    id  SERIAL NOT NULL,
    title VARCHAR(255),
    PRIMARY KEY (id) );
```

3. pgAdmin PostgreSQL Tools

pgAdmin es la herramienta de gestión de código abierto líder para Postgres, la base de datos de código abierto más avanzada del mundo. pgAdmin 4 está diseñado para satisfacer las necesidades de usuarios principiantes y experimentados de Postgres por igual, proporcionando una interfaz gráfica poderosa que simplifica la creación, mantenimiento y uso de objetos de bases de datos.

4. Diccionario de datos

Contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización. Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información, se desarrolla durante el análisis de flujo de datos y auxilia a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño.

Cada entrada en el diccionario consiste de un conjunto de detalles que describen los datos utilizados o producidos por el sistema. Cada uno esta identificado con:

- Un nombre: para distinguir un dato de otro.
- Descripción: indica lo que representa en el sistema.
- Alias: porque un dato puede recibir varios nombres, dependiendo de quien uso este dato.
- Longitud: porque es de importancia de saber la cantidad de espacio necesario para cada dato.
- Valores de los datos: porque en algunos procesos solo son permitidos valores muy específicos para los datos. Si los valores de los datos están restringidos a un intervalo especifico, esto debe estar en la entrada del diccionario.

PostgreSQL proporciona de manera sencilla toda la información de los objetos que se encuentran en una base de datos a través de su INFORMATION_SCHEMA.

Al consultar el INFORMATION_SCHEMA de PostgreSQL (que no es más que una serie de vistas que nos muestran la estructura de nuestras bases de datos) nos encontraremos toda la información referente tablas, vistas y campos contenidos dentro de las mismas.

Las siguientes tablas son las que usaremos para generar nuestro diccionario de datos:

- PG_CLASS.
- TABLES.
- COLUMNS.

- TABLE_CONSTRAINTS.
- KEY_COLUMN_USAGE.
- REFERENTIAL_CONSTRAINTS.
- CONSTRAINT_COLUMN_USAGE.

5. Actividad

Para esta práctica vamos a agregar integridad a nuestras tablas de la base de datos, para esto vamos a hacer uso de los **ALTER TABLE** para alterar la estructura de nuestras tablas para poder agregar agregar los tipos de restricción en cada uno de nuestras tablas.

Por ultimo, deberán generar un diccionario de datos de la base de datos, el cual deberá contener contener el nombre de las tablas, las columnas de cada una de las tablas, todos los constraints de la tabla.

6. Entregables

Se debe crear un archivo **DDL.sql** el cual contendrá las instrucciones para la creación de la base de datos, sus tablas y sus restricciones.

Se debe crear un archivo **diccionario.sql** el cual contendrá las instrucciones para la creación de los metadatos.

El diccionario de datos deberá ser generado en un documento llamado **diccionario.pdf**, el cual lo deberán guardar dentro de la carpeta DOC.