Fundamentos de Bases de Datos.
Práctica 5.

Profesor: M.I. Gerardo Avilés Rosas
gar@ciencias.unam.mx
Laboratorio: Luis Eduardo Castro Omaña
lalo_ castro@ciencias.unam.mx

30 de marzo de 2020

Se dan a conocer especificaciones de entrega para la practica 5.

1. DDL

Un esquema de base de datos se especifica mediante un conjunto de definiciones expresadas mediante un lenguaje especial llamado lenguaje de definición de datos (DDL)[1].

Especificamos el almacenamiento y los métodos de acceso usados por el sistema de bases de datos por un conjunto de instrucciones en un tipo especial de DDL denominado lenguaje de almacenamiento y definición de datos. Estas instrucciones definen los detalles de implementación de los esquemas de base de datos, que se ocultan usualmente a los usuarios.

2. Tablas

Una tabla consiste de renglones y columnas. Las columnas definen los datos que queremos almacenar y las filas contienen los valores para esas columnas.

Existen los siguientes tipos de tablas en PostgreSQL:

- Tabla Ordinaria: Una tabla permanente. La vida útil de la tabla comienza con la creación de la tabla y termina con la eliminación de la tabla. Tablas ordinarias deben pertenecer a un esquema, si no se especifica un schema a la tabla, la tabla se asigna el esquema actual.

[1] Siglas en inglés de Data Definition Language

- Tabla Temporal: Tablas no permanentes. La vida útil de la tabla es la sesión del usuario. Tablas temporales se eliminan al finalizar la sesión del usuario o al termino de una transacción. Tablas permanentes pertenecen a un schema especial, por lo que no se debe especificar el esquema al momento de crear la tabla. El nombre de la tabla debe ser diferente a cualquier objeto de la base de datos.

- Unlogged table: Tienen un tiempo de escritura menor que las tablas ordinarias o temporales, esto se debe a que los datos no se escriben en los archivos WAL. Unlogged tables no son crash-safe, por lo que si la base de datos tiene algún fallo, los datos almacenados en este tipo de tablas no se pueden recuperar.

- Tabla hija: Son tablas que heredan de una o mas tablas ordinarias.Modificaciones a los podres de propagan a las tablas hijas. La herencia a menudo se usa con exclusión de restricciones para particionar físicamente los datos en el disco duro y mejorar el rendimiento al recuperar un subconjunto de datos que tiene un cierto valor.

2.1. Crear Tablas

Crea una nueva tabla en PostgreSQL

```
--Tipo de base de datos
CREATE [ [ GLOBAL | LOCAL ] { TEMPORARY | TEMP } | UNLOGGED ]
--Nombre de base de datos y definición de columnas
TABLE [ IF NOT EXISTS ] [database_name].[schema_name].table_name ( [
  { column_name data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
    | table_constraint
    | LIKE source_table [ like_option ... ] }
    [, ... ]
] )
--Referencia a una tabla padre
[ INHERITS ( parent_table [, ... ] ) ]
--Tablespace a almacenar la tabla
[ TABLESPACE tablespace_name ]
```

database_name: Es el nombre de la base de datos en la que se crea la tabla. database_name debe especificar el nombre de una base de datos existente. Si no se especifica, database_name el valor predeterminado es la base de datos actual.

schema_name: Es el nombre del esquema al que pertenece la nueva tabla. Si no se específica un esquema, los objetos creados se agregan a el esquema `public`.

table_name: Es el nombre de la nueva tabla. Los nombres de tabla deben seguir las convenciones de nombrado para identificadores.

Sintaxis completa para crear una tabla se puede encontrar en: `https://www.postgresql.org/docs/current/sql-createtable.html`.

2.2. Tipos de datos nativos.

En PostgreSQL, cada columna, variable local, expresión y parámetro tiene un tipo de datos relacionado. Un tipo de datos es un atributo que especifica el tipo de datos que el objeto puede contener.

Se debe de considerar el tipo de dato de cada columna antes de crear una tabla en la base de datos. Cambiar un tipo de dato en un ambiente productivo es una operación costosa, la cual puede bloquear la tabla y en algunos casos, es necesario reescribir la tabla. Por lo que escoger el tipo de dato correcto puede resultar complejo, aquí algunas aspectos a considerar al momento de escoger los tipos de datos para tu tabla:

- Extensibilidad: Considerar si es posible cambiar la longitud del tipo de dato sin necesidad de reconstruir la tabla.
- Tamaño del tipo de dato: Escoger un tamaño que se ajuste a la necesidad del dato. Asignar tipos de datos muy grandes pueden consumir más recursos de los necesarios.
- Soporte: Considerar que las tablas pueden ser consumidas por aplicaciones escritas en diferentes lenguajes de programación. Por lo que se recomienda utilizar tipos de datos soportados por estos lenguajes, de otra forma se tendrá que trabajar extra para serializar y desearizar los datos.

Tipos de datos de PostgreSQL se organizan en las siguientes categorías:

2.2.1. Numericos

- Enteros: Tipos de datos numéricos exactos que utilizan datos enteros.

Tipo de datos	Intervalo	Storage
`bigint`	De −9,223,372,036,854,775,808 a 9,223,372,036,854,775,807.	8 bytes
`integer`	De $-2,147,483,648$ a $+2,147,483,647$.	4 bytes
`smallint`	De $-32,768$ a $+32,767$.	2 bytes

- Decimal o numéricos: Tipos de datos numéricos que tienen precisión y escala fijas. Decimal y numeric son sinónimos y se pueden usar indistintamente.

  `decimal`[(p[,s])] y `numeric`[(p[,s])] Números de precisión y escala fijas. Cuando se utiliza la precisión máxima, los valores válidos se sitúan entre

-10^38+1 y 10^38-1. numérico es funcionalmente equivalente a decimal.

p (precisión): El número total máximo de dígitos decimales que almacenará, tanto a la izquierda como a la derecha del separador decimal. La precisión debe ser un valor comprendido entre 1 y la precisión máxima de 38. La precisión predeterminada es 18.

s (escala): El número de dígitos decimales que se almacenará a la derecha del separador decimal. Este número se resta de p para determinar el número máximo de dígitos a la izquierda del separador decimal. El número máximo de dígitos decimales que se puede almacenar a la derecha del separador decimal. Escala debe ser un valor comprendido entre 0 y p. Solo es posible especificar la escala si se ha especificado la precisión. La escala predeterminada es 0.

- `real`: Al menos 6 dígitos de precisión y puede variar dependiendo de la de la plataforma. 4 bytes.
- `double precision` Al menos 15 dígitos de precisión y puede variar dependiendo de la de la plataforma. 8 bytes.

2.2.2. Caracteres

Son tipos de datos de cadena de longitud fija o de longitud variable.

- `char`: Equivalente a `char[1]`. Longitud máxima 1.
- `name`: Equivalente a `varchar[64]`. Longitud máxima 64.
- `char [ ( n ) ]`: Datos de cadena no Unicode de longitud fija `n`. `n` define la longitud de cadena y debe ser un valor entre 1 y 10485760. El tamaño de almacenamiento es n bytes.
- `varchar [ ( n | max ) ]`: Datos de cadena no Unicode de longitud variable. `n` define la longitud de cadena y puede ser un valor entre 1 y 10485760. `max` indica que el tamaño máximo de almacenamiento es 1 GB. El tamaño de almacenamiento es la longitud real de los datos especificados + 2 bytes.
- `text`: Longitud imitada.

2.2.3. Fechas

Tipo de datos que almacenan datos de fechas.

- `time`: Define un tiempo de un día. La hora es sin conocimiento de zona horaria y se basa en un reloj de 24 horas.

- `date`: Define una fecha con el formato YYYY-MM-DD

Para completa descripción de tipos de datos, referirse a la siguiente documentación `https://www.postgresql.org/docs/11/datatype.html`.

3. Integridad de datos

3.1. Restricciones

Las restricciones le permiten definir la manera en que motor de base de datos exigirá automáticamente la integridad de una base de datos. Las restricciones definen reglas relativas a los valores permitidos en las columnas y constituyen el mecanismo estándar para exigir la integridad. El uso de restricciones es preferible al uso de desencadenadores DML[2], reglas y valores predeterminados. El optimizador de consultas también utiliza definiciones de restricciones para generar planes de ejecución de consultas de alto rendimiento.

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. 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 UNIQUE exigen la unicidad de los valores de un conjunto de columnas. En una restricción UNIQUE, dos filas de la tabla no pueden tener el mismo valor en las columnas. Las claves principales también exigen exclusividad, pero no aceptan NULL como uno de los valores exclusivos.

- Las restricciones PRIMARY KEY identifican la columna o el conjunto de columnas cuyos valores identifican de forma exclusiva cada una de las filas de una tabla. Dos filas de la tabla no pueden tener el mismo valor de clave principal. No se pueden asignar valores NULL a ninguna de las columnas

[2]Se tratará con mas detalle en la siguiente práctica.

de una clave principal. Se recomienda utilizar una columna pequeña de tipo entero como clave principal. Todas las tablas tienen que tener una clave principal. Una columna o combinación de columnas certificada como valor de clave principal se denomina clave candidata.

- Las restricciones FOREIGN KEY identifican y exigen las relaciones entre las tablas. Una clave externa de una tabla apunta a una clave candidata de otra tabla. No se puede insertar una fila que tenga un valor de clave externa, excepto NULL, si no hay una clave candidata con dicho valor. La cláusula ON DELETE controla las acciones que se llevarán a cabo si intenta eliminar una fila a la que apuntan las claves externas existentes. La cláusula ON DELETE tiene las siguientes opciones:
  - NO ACTION especifica que la eliminación produce un error.
  - CASCADE especifica que también se eliminan todas las filas con claves externas que apuntan a la fila eliminada.
  - SET NULL especifica que todas las filas con claves externas que apuntan a la fila eliminada se establecen en NULL.
  - SET DEFAULT especifica que todas las filas con claves externas que apuntan a la fila eliminada se establecen en sus valores predeterminados.

  La cláusula ON UPDATE define las acciones que se llevarán a cabo si intenta actualizar un valor de clave candidata a la que apuntan las claves externas existentes. Esta cláusula también admite las opciones NO ACTION, CASCADE, SET NULL y SET DEFAULT.

3.2. Incrementadores

3.2.1. IDENTITY

Cada tabla debe contener una columna como llave primaría. Es ampliamente recomendable utilizar una columna que se incremente por cada registro en la tabla, dicha columna podría ser la llave primaria o parte de ella de la tabla que la contenga. En PostgreSQL se crea una columna de identidad en una tabla; la cual, genera un campo que se incrementara en i cada vez que se realice un registro en la tabla que la contiene. Esta propiedad se usa con las instrucciones CREATE TABLE y ALTER TABLE, se declara como sigue:

```
column_name type GENERATED { ALWAYS | BY DEFAULT }
AS IDENTITY [ ( sequence_options ) ]
```

- El tipo de la columna puede ser `SMALLINT`, `INTEGER` o `BIGINT`.
- Cuando se especifica `GENERATED ALWAYS` el valor siempre se va a incrementar automáticamente. Si al momento de hacer un `INSERT` o `UPDATE` y se especifica un valor a la columna, PostgreSQL creara un error y no se podrá realizar la inserción/actualización.

- Cuando se especifica `GENERATED ALWAYS`, el valor se actualizará solo si no se especifica un valor al momento de realizar inserción/actualización. Si se especifica un valor en `INSERT` o `UPDATE` se guardará el valor asignado sin error, siempre y cuando cumpla con las restricciones asignadas a la columna.

3.2.2. 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 `SELECT nextval('sequence_name');`.

```
CREATE [ TEMPORARY | TEMP ] SEQUENCE [ IF NOT EXISTS ] name
    [ AS data_type ]
    [ INCREMENT [ BY ] increment ]
    [ MINVALUE minvalue | NO MINVALUE ] [ MAXVALUE maxvalue | NO MAXVALUE ]
    [ START [ WITH ] start ] [ CACHE cache ] [ [ NO ] CYCLE ]
    [ OWNED BY { table_name.column_name | NONE } ]
```

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.

4. Actividad

Deberán crear las instrucciones DDL para crear el esquema de bases de datos utilizando como base el diagrama relacional que realizaron en la práctica pasada.

Los nombres de las tablas y columnas debe cumplir con las convenciones de codificación establecidas en prácticas pasadas. Se debe elegir el correcto tipo de dato para cada columna de acuerdo a las necesidades del caso de uso. Además, se deben definir las restricciones necesarias para cada una de las columnas definidas en la tabla.

Por último, se deben especificar las llaves foráneas, compuestas y primarias de cada tabla.

5. Entregables

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

La fecha de entrega es el día lunes 13 de Abril de 2020.