Universidad Simón Bolívar
Departamento de Computación y T.I.

Laboratorio semana 3

Para el primer taller nos ocuparemos de los objetos básicos para almacenar datos (tablas) y de las restricciones definibles a través de objetos CONSTRAINT. Los comandos que provee **PostgreSQL** para realizar estas operaciones son:

- **CREATE TABLE**
- **DROP TABLE**
- **ALTER TABLE**

Dado que los "constraints" son objetos que dependen estrictamente de la existencia de una tabla, no existe un comando específico que crear o eliminar un "constraint", sino que estos comandos deben formar parte de lo que se indica en un comando CREATE TABLE o ALTER TABLE. A continuación, se explicará brevemente el comando CREATE TABLE, DROP TABLE y la forma de agregar "constraints" durante la creación de una tabla o posterior a la creación de la misma utilizando el comando ALTER TABLE. Ud. deberá consultar el manual de referencia de SQL para estudiar todas las potencialidades del comando ALTER TABLE.

EL COMANDO CREATE TABLE

Este comando permite crear una tabla en la base de datos. Una forma particular del comando CREATE TABLE para una tabla basada en el modelo relacional, y que se crea sin requerir de datos ya existentes en otras tablas es:

CREATE TABLE table <table_name>(

<column_name> <datatype> [DEFAULT <expr>] [NULL | NOT NULL]

{, <column_name> <datatype> [DEFAULT <expr>] [NULL | NOT NULL] }

{, <table_constraint>});

Donde:

- <table_name> es el nombre con el que se designará la tabla.
- <column_name> es el nombre con el que se designará a cada una de las columnas
- <datatype> representa uno de los tipos de datos admitidos por **PostgreSQL**.
- <expr> es una expresión que debe evaluar
- <table_constraint> es la especificación de un "constraint" a ser definido sobre las columnas de la tabla.

La forma presentada es una de las posibles formas de ordenar la declaración de columnas y restricciones en la que:

- Primero se describe cada una de las columnas de la tabla con su tipo de datos, su valor por defecto (utilizando la cláusula DEFAULT) y si la columna admite o no el valor NULL.
- Luego se indican las restricciones de integridad asociada a la tabla utilizando las estructuras de "constraints" que se verán posteriormente.

La forma del comando CREATE TABLE permite diferenciar las estructuras de las restricciones implícitas y hacer más sencilla la lectura del comando. Sin embargo, en el manual de referencia de SQL Ud. podrá encontrar todas las variaciones sintácticas de este comando.

EL COMANDO DROP TABLE

Este comando permite eliminar una tabla (borrar las filas e inclusive eliminar la estructura) de la base de datos. La forma general de este comando es:

DROP TABLE [<table_name>] [CASCADE | RESTRICT];

Si se utiliza la opción CASCADE, se eliminarán todas aquellas restricciones de integridad referencial definidas en otras tablas de la base de datos que referencien a la tabla que se está eliminando.

DEFINICIÓN DE CONSTRAINTS PostgreSQL

PostgreSQL permite definir los siguientes tipos de "constraints":

• **NOT NULL**. Cuando esta declaración se agrega en la especificación de una columna se indica que ninguna fila de la tabla puede ser tal que el valor para dicha columna sea NULL. La opción por defecto en **PostgreSQL** es la de suponer que toda columna admite el valor NULL, sin embargo puede indicarse explícitamente (para hacer más legible un "script") el hecho de que la columna admita el valor NULL.

• **UNIQUE**. Cuando esta declaración se agrega a una o más columnas de una tabla se indica que, en ningún momento, podrán existir dos filas en la tabla que tengan el mismo valor para las columnas que se están indicando como únicas en el "constraint".

• **PRIMARY KEY**. Cuando esta declaración se agrega a una o más columnas de una tabla se indica que estas columnas constituyen la clave primaria de la tabla.

• **FOREIGN KEY**. Cuando esta declaración se agrega a una o más columnas de una tabla se indica que estas columnas constituyen una clave foránea de la tabla hacia alguna tabla de la base de datos.

• **CHECK**. Cuando esta declaración se agrega a una tabla se indica que no se admitirá ninguna fila en la tabla que no cumpla con una condición lógica establecida por el "constraint".

De acuerdo con la forma de especificación del comando CREATE TABLE dada anteriormente, la cláusula puede entonces tener las siguientes formas:

• CONSTRAINT <constraint_name> PRIMARY KEY (<column_name>[{, <column_name>}])

• CONSTRAINT <constraint_name> UNIQUE <constraint_name> (<column_name>[{, <column_name>}])

• CONSTRAINT <constraint_name> FOREIGN KEY (<column_name>[{, <column_name>}]) REFERENCES <table_name>

• CONSTRAINT <constraint_name> CHECK (<condition>)

Donde:

• <constraint_name> es el nombre con el que se designará al "constraint" en el esquema donde se crea la tabla que lo incluye.

• <column_name> es el nombre de una columna de la tabla en la que se define el "constraint"

• <table_name> es el nombre de una tabla definida en el esquema donde existe la tabla que incluye el "constraint".

• <condition> es una expresión lógica de SQL.

En el manual de referencia de SQL Ud. podrá encontrar información sobre las reglas de formación de expresiones lógicas de SQL. Un punto importantísimo a destacar es el de garantizar que los nombres de los "constraints" sean nemónicos. Toda vez que un "constraint" sea violado, el DBMS generará un mensaje

de error indicando el “constraint” que ha fallado. Asignar nombres nemónicos permitirá hacer la depuración de programas y la carga de datos mucho más sencilla, además de garantizar una perfecta cohesión entre el esquema de implantación y la documentación del esquema lógico. Es así como se sugiere utilizar el siguiente estándar de nomenclatura:

• Para restricciones de no-nulidad se recomienda no crear “constraints” sino declarar la no-nulidad al momento de creación de la tabla y para cada columna. Esto se debe al hecho de que el módulo de chequeo de integridad de **PostgreSQL** no utiliza el nombre de la restricción de integridad para indicar que el valor de una columna en una fila deba ser nulo si se indica lo contrario.

• Para asociar un “constraint” de clave primaria a una tabla se recomienda designar el “constraint” con el nombre PK_<table_name>.

• Para asociar un “constraint” de unicidad relacionado con una clave alterna de la tabla, se recomienda designar el “constraint” con el nombre AK_<table_name>. Si existen varias claves alternas podrá agregársele al nombre del “constraint” un indicador que permita diferenciar cada una de estas claves alternas.

• Para asociar un “constraint” de clave foránea a una tabla que referencia a una tabla se recomienda designar el “constraint” con el nombre FK_<table_name$_1$>_<table_name$_2$>. Si existen múltiples claves foráneas en la tabla que referencia a la tabla entonces será necesario colocar algún indicador que permita diferenciar cada uno de los “constraints”. Un posible nombre a utilizar será el nombre de la interrelación (y en rol en caso de interrelaciones recursivas) del esquema conceptual a partir de la cual se generó la clave foránea que define el “constraint”.

• Para asociar un “constraint” que represente una restricción de dominio sobre una columna <column_name> de una tabla <table_name>, se sugiere utilizar DOM_<table_name>_<column_name> como nombre del “constraint”.

• Para asociar un “constraint” que represente una restricción explícita, que puede ser descrita a través de una restricción de tipo CHECK, de una tabla se sugiere utilizar EXP_<table_name>_R_<constraint_number> como nombre del “constraint”. En este nombre representará el número de restricción explícita asociado en la documentación del esquema relacional que se implanta.

Los “constraints” pueden ser agregados a una tabla previamente creada, o eliminados de una tabla existente. Para tal fin se pueden utilizar dos variaciones del comando ALTER TABLE, cuya sintaxis a continuación se indica:

• ALTER TABLE <table_name> ADD (<table_constraint> [,<table_constraint>]);

Permitirá agregar una o más “constraints” a la tabla existente en la base de datos. Cada uno de los “constraints” que se añaden a la tabla seguirán las convenciones sintácticas de la cláusula <table_constraint>.

• ALTER TABLE <table_name> DROP CONSTRAINT <constraint_name> [CASCADE];

Eliminará de la tabla la restricción <constraint_name>. Si se utiliza la cláusula CASCADE, la eliminación del “constraint” tendrá como efecto eliminar cualquier otro “constraint” que esté relacionado con el “constraint” que se elimina. Por ejemplo, si se elimina un “constraint” de clave primaria de una tabla *A* bajo modalidad “constraint” entonces se eliminarán los “constraints” de clave foránea que referencien a *A*.

UN EJEMPLO DE CREACIÓN DE TABLAS

A continuación, se presenta un ejemplo de un comando de creación de tablas

```
CREATE TABLE EMPLEADO (
        NUMERO NUMERIC(3) NOT NULL,
        CI NUMERIC(8) NOT NULL,
        NOMBRE VARCHAR2(20) NOT NULL,
```

```
    CARGO VARCHAR2(9) NOT NULL,
    JEFE NUMERIC(3) NULL,
    INGRESO DATE NULL,
    SUELDO NUMERIC(10,2) NOT NULL,
    NUMERO_DEPTO NUMERIC(2) NOT NULL,
    CONSTRAINT PK_EMPLEADO PRIMARY KEY (NUMERO),
    CONSTRAINT AK_EMPLEADO UNIQUE (CI),
    CONSTRAINT FK_EMPLEADO_EMPLEADO FOREIGN KEY (JEFE) REFERENCES EMPLEADO,
    CONSTRAINT FK_EMPLEADO_DEPTO FOREIGN KEY (NUMERO_DEPTO) REFERENCES DEPTO,
    CONSTRAINT DOM_EMPLEADO__SUELDO CHECK (SUELDO > 0),
    CONSTRAINT DOM_EMPLEADO__NOMBRE CHECK (NOMBRE= NLS_UPPER(NOMBRE)) );
```