Fundamentos de Bases de Datos.
Práctica 12.

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

16 de diciembre de 2021

Se dan a conocer especificaciones de entrega para la práctica 12.

1. PL/pgSQL

1.1. Structured Query Language Procedural Language (PL/pgSQL)

Structured Query Language Procedural Language (PL/pgSQL) es el lenguaje de programación incrustado en el SMBD **postgresql**. Soporta todas las consultas de SQL, incluyendo características tales como:

- El manejo de variables
- Estructuras modulares
- Estructuras de control de flujo y toma de decisiones
- Control de excepciones

Este lenguaje es utilizado generalmente para la definición de objetos dentro de la BD, tales como disparadores y funciones.

A diferencia de PL/SQL o Transact-SQL, PL/pgSQL no cuenta con una sintaxis específica para definir procedimientos almacenados, debido a que utiliza la misma sintaxis de creación de funciones.También cabe mencionar que PL/pgSQL se encuentra instalado por defecto en postgresql a partir de la versión 9.0 [**?**].

1.2. Funciones

Una función definida por el usuario es un conjunto de instrucciones SQL y PL/pgSQL, las cuales contienen declaraciones, asignaciones e instrucciones para control de flujo dentro de la BD. Las funciones se almacenan en el servidor de la BD y

pueden ser invocadas mediante la sintaxis de SQL[**?**, **?**](Capítulo 9 pp. 251-275, Capítulo 2 pp. 33-40).

Sintaxis de funciones

```
CREATE [ OR REPLACE ] FUNCTION
 name ( [  [ argname ] argtype ] [, ...] ] )
 [ RETURNS rettype  | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE lang_name
 | AS 'definition'
}
```

Para mayor información sobre la sintaxis y el uso de las funciones, se puede consultar la documentación oficial de *postgresql https://www.postgresql.org/docs/9.5/sql-createfunction.html*. [**?**]

Las funciones se comportan como se muestran en la siguiente tabla:

Funcionalidad	**Cumple**
Utilizada en una expresión	Sí
Devuelve un valor	Sí
Devuelve valores como parámetros de salida	No
Devuelve un registro de un dominio específico	Sí
Devuelve varios registros de un dominio específico	No

Cuadro 1: Comportamiento de las funciones en PostgreSQL.

A continuación, se ilustra cada uno de los casos de la tabla anterior:

- Utilizada en una expresión

  Una función se puede utilizar dentro de una llamada de SQL, esto quiere decir que al realizar una consulta es posible invocar una función o enviar como argumento el resultado de la consulta. Por ejemplo, se muestra la creación de una función encargada de duplicar el salario de los empleados.

  Para ello, primero se crea una tabla de *Empleados*, la cual contendrá información de ellos.

```
CREATE TABLE Empleados(
nombre VARCHAR(70) PRIMARY KEY,
salario INTEGER,
cubiculo INTEGER
);

COMMENT ON TABLE Empleados  IS
```

```
'Captura|Tabla que contiene registros de empleados de prueba.';
COMMENT ON COLUMN Empleados.nombre IS
'|Identificador de la tabla de prueba.';
COMMENT ON COLUMN Empleados.salario IS
    '|Este campo hace referencia al salario asociado al empleado.';
COMMENT ON COLUMN Empleados.cubiculo IS
     '|Este campo hace referencia al número del cubículo del empleado.';

     INSERT INTO Empleados VALUES ('Carlos Escalona', 150 , 0);
INSERT INTO Empleados VALUES ('Alejandra del Angel', 130 , 1);
INSERT INTO Empleados VALUES ('Jean Pierre', 125 , 2);

CREATE OR REPLACE FUNCTION duplica_salario(Empleados) RETURNS INTEGER
    AS 'SELECT $1.salario * 2 AS salario_deseado;' LANGUAGE 'sql';

     SELECT nombre, duplica_salario(Empleados) salario_deseado  FROM
Empleados WHERE Empleados.cubiculo in (1,0);
```

- Devuelve un valor

  Los valores de salida se utilizan para enviar el resultado de un procedimiento almacenado o una función. Por ejemplo, se muestra la creación de una función encargada de sumar dos números, los cuales son sumados por medio de operaciones básicas de SQL.

```
CREATE FUNCTION sumaNumeros(INTEGER, INTEGER) RETURNS INTEGER
     AS 'SELECT $1 + $2;' LANGUAGE 'sql';
    SELECT sumaNumeros(1, 2) AS resultado;
```

- Devuelve un registro de un dominio específico

  Regresa un solo registro o valor según sea especificado en el return de este, el cual debe cumplir con el cuerpo de procedimiento almacenado o de la función. Ejemplo:

```
CREATE TABLE Empleados (
id INT PRIMARY KEY,
subid INT,
nombre TEXT);


INSERT INTO Empleados VALUES (0, 1, 'Jose Carlos');
```

```
INSERT INTO Empleados VALUES (1, 2, 'Jean Pierre');
INSERT INTO Empleados VALUES (2, 1, 'Carlos Escalona');

  CREATE FUNCTION obten_un_empleado(INT) RETURNS Empleados AS $$
    SELECT * FROM Empleados WHERE id = $1;
$$ LANGUAGE SQL;


 SELECT obten_un_empleado(1);
```

1.3. Procedimientos almacenados

El procedimiento almacenado (o SP por sus siglas en inglés) es un conjunto de instrucciones SQL y PL/pgSQL, las cuales contienen declaraciones, asignaciones e instrucciones para control de flujo dentro de la BD. Los procedimientos almacenados son guardados dentro de la BD y pueden ser invocados mediante sintaxis de SQL.

NOTA: A diferencia de SMBD como ORACLE o SQL SERVER, en PostgreSQL no existe la palabra reservada PROCEDURE para los procedimientos almacenados, para estos se utiliza la palabra reservada FUNCTION, la cual puede o no devolver un valor, esto depende de la forma en que se implemente y se invoque.

1.3.1. Paso de parámetros a procedimientos almacenados en PL/pgSQL

En PL/pgSQL se pueden pasar tres tipos parámetros a procedimientos:

1. IN: se utilizan para enviar valores a procedimientos almacenados.
2. OUT: es utilizado para obtener valores de procedimientos almacenados. Esto es similar a un tipo de devolución en funciones.
3. IN OUT: son utilizados para enviar y obtener valores de procedimientos almacenados.

NOTA: Si el tipo de parámetro no está definido explícitamente, de manera predeterminada es de tipo IN.

Sintaxis de procedimientos almacenados

```
CREATE [ OR REPLACE ] FUNCTION
 name ( [  [ argname ] argtype ] [, ...] ] )
 [ RETURNS rettype  | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE lang_name
 | AS 'definition'
}
```

Para mayor información sobre la sintaxis y el uso de las funciones, se puede consultar la documentación oficial de *postgresql https://www.postgresql.org/docs/9.5/sql-createfunction.html*.

En general los procedimientos almacenados se comportan como se muestra en la siguiente tabla:

Funcionalidad	**Cumple**
Utilizada en una expresión	No
Devuelve un valor	No
Devuelve valores como parámetros de salida	Sí
Devuelve un registro de un dominio específico	Sí
Devuelve varios registros de un dominio específico	Sí

Cuadro 2: Comportamiento de los procedimientos almacenados en PostgreSQL.

- Devuelve valores como parámetros de salida

  Los parámetros de salida (OUT) se utilizan para enviar el resultado de un procedimiento de almacenado. Este es un parámetro de **solo escritura**, es decir, no se pueden pasar valores a los parámetros OUT mientras se ejecuta el procedimiento almacenado. Sin embargo, es posible asignar valores al parámetro OUT dentro del procedimiento almacenado de tal manera que el programa que lo ejecuta reciba el valor de salida para utilizarlo dentro de la ejecución de este.

  Ejemplo:

```
CREATE FUNCTION genera_suma_producto (primera_entrada INT,
segunda_entrada INT, OUT resultado_suma INT,  OUT
resultado_producto INT) AS
'SELECT primera_entrada + segunda_entrada,
primera_entrada * segunda_entrada'
LANGUAGE SQL;
```

- Devuelve un registro de un dominio específico

  Regresa un solo registro o valor según sea especificado en el return de este, el cual debe cumplir con el cuerpo de procedimiento almacenado o de la función. Ejemplo:

```
CREATE TABLE Empleados (
id INT PRIMARY KEY,
subid INT,
nombre TEXT);
```

```
INSERT INTO Empleados VALUES (0, 1, 'Jose Carlos');
INSERT INTO Empleados VALUES (1, 2, 'Jean Pierre');
INSERT INTO Empleados VALUES (2, 1, 'Carlos Escalona');

  CREATE FUNCTION obten_un_empleado(INT) RETURNS Empleados AS $$
    SELECT * FROM Empleados WHERE id = $1;
$$ LANGUAGE SQL;


 SELECT *, upper(nombre) FROM obten_un_empleado(1) AS t1;
```

- Devuelve varios registros de un dominio específico

  Regresa un conjunto de registros o valores según sea especificado en el return del procedimiento de almacenado. Se debe tener cuidado con cómo se define la salida de este, debido a que se podría ocupar como función o procedimiento almacenado. Ejemplo:

```
CREATE TABLE Empleados (
id INT PRIMARY KEY,
subid INT,
nombre TEXT);

  INSERT INTO Empleados VALUES (0, 1, 'Jose Carlos');
INSERT INTO Empleados VALUES (1, 2, 'Jean Pierre');
INSERT INTO Empleados VALUES (2, 1, 'Carlos Escalona');

CREATE FUNCTION obten_conjunto_empleados(INT) RETURNS SETOF
Empleados AS $$
 SELECT * FROM Empleados WHERE id >= $1;
$$ LANGUAGE SQL;

SELECT * FROM obten_conjunto_empleados(1) AS t1;
```

2. Actividad

Haciendo uso de tu base de datos desarrollada a lo largo del semestre, se deben crear un archivo llamado **funciones.sql** con las siguientes funciones :

1. Una función que reciba el RFC y nos regrese la edad de los clientes y empleados de los viveros.

2. Una función que reciba el ID del empleado y nos regrese el número de ventas que a realizado.

3. Una función que reciba el RFC y te regrese el número de días faltantes para el cumpleaños del empleado o del cliente.

4. Una función que reciba el nombre del empleado y su fecha de nacimiento y te regrese su RFC , ejemplo: SELECT regresaCURP('Jean Pierre Jaramillo Avila' , '02/08/1988') ; debe regresar ->JAJP080288

5. Una función que reciba el id del cliente y nos regrese 3 valores; cada una de ellas con el total de dinero gastado en compras. Los valores deben ser de la siguiente manera: Efectivo, Tarjeta débito y Tarjeta de crédito.

Adicional a lo anterior sera necesario crear un archivo llamado **procedimientos_almacenados.sql** con los siguientes procedimientos almacenados:

1. Un procedimiento el cual se encarga de registrar un empleado, en este procedimiento debe introducir la información del empleado y se debe encargar de insertar en las tablas correspondientes, es importante que no permitan la inserción de numero o símbolos cuando sean campos relacionados a nombres y apellidos.

2. Un procedimiento el cual se encargue de eliminar un empleado, en este procedimiento debes introducir SÓLO el id del empleado y cuando elimines este empleado debes eliminar todas sus referencias de sus tablas hijas, a excepción de las tablas de ventas, ahí deberás cambiar el id del empleado por otro valor (NULL / DEFAULT).

3. Un procedimiento el cual se encargue de registrar las ventas físicas realizadas por un cliente.

4. Un procedimiento el cual se encargue de registrar las ventas online realizadas por un cliente.

3. Entregables

Se debe crear un archivo **funciones.sql** el cual contendrá las funciones solicitadas y se debe crear un archivo **procedimientos_almacenados.sql** el cual contendrá los procedimientos almacenados solicitados anteriormente.