UTEC
Departamento de Ciencias de la Computación
CS2701 - Base de Datos I
Laboratorio 9.1 **Fecha:09/06/2021**

Trigger - Procedimientos Almacenados

Profesor: Teófilo Chambilla
ACL: Alexandra Shulca Romero

Conectando postgres con python

1. Crear la base de datos **tienda** y restuar el script de `https://gist.github.com/tchambil/d7364de6d544d8665434f34e61accf86`

2. Dentro del servidor crear un usuario propio con el que se conectará a la base de datos:

   - Usando el terminal de linux acceda al cliente de PostgresSQL.

     ```
     # psql -h 204.2.195.90 -p 30794 -U <UserName> -d <dbname>

     Donde:
     <UserName> su usuario es usr_group seguido de su número de grupo.
     Ejem: usr_group5

     <Password> se le compartirá en clases

     <dbname> es dbgroup seguido de su número de grupo.
     Ejem: dbgroup5
     ```

   - Crear un esquema donde el nombre será formada por el nombre del sistema (OpenDay) y su código de UTEC.

     ```
     CREATE SCHEMA <Nombre><Codigo>;
     Ejemplo:

     CREATE SCHEMA Tienda{Grupo};
     ```

   - Crear usuario para acceder a la base de datos y esquema

     ```
     sudo -u postgres createuser <username>
     sudo -u postgres psql
     postgres=# alter user <username> with encrypted password '<password>';
     postgres=# grant all privileges on database <databasename> to <username> ;
     postgres=# \c tienda (Ejemplo tienda es la base de datos)
     tienda=# grant all privileges on all tables in schema <schemaname>  TO <username>
     tienda=# \q
     ```

- Pruebe su usuario ejecutando las siguientes consultas:

```
sudo -u postgres psql -U <username> -d tienda
tienda=> select * from clientes;
tienda=> select * from empleados;
```

3. Pruebe el siguiente codigo en Python para conectarse a la base de datos (python3)

   Puede obtener la estructura desde github:`https://github.com/tchambil/CS2701_2021-1`

```python
import psycopg2
conn = psycopg2.connect(database='tienda', user='teofilo',
   password='159753')
cur = conn.cursor()
cur.execute('select * from clientes;')
for row in cur.fetchall():
  print(row[0], row[1], row[2])
cur.close()
conn.close()
```

Resultado esperado

```
(1, '1890786576', 'SUPERMERCADO ESTRELLA         ')
(2, '1298765477', 'EL ROSADO                     ')
(3, '1009876567', 'DISTRIBUIDORA PRENSA          ')
(4, '1876090006', 'SU TIENDA                     ')
(5, '1893456776', 'SUPERMERCADO DORADO           ')
(6, '1678999891', 'MI COMISARIATO                ')
(7, '1244567888', 'SUPERMERCADO DESCUENTO        ')
(8, '1456799022', 'EL DESCUENTO                  ')
(9, '1845677777', 'DE LUISE                      ')
(10, '183445667 ', 'YARBANTRELLA                  ')
```

4. Crear un procedimiento almacenado para responder ¿En que fechas el cliente identificado con RUC '1298765477' registró ordenes de pedido?

```
CREATE OR REPLACE FUNCTION get_cliente_by_ruc(p_ruc varchar)
RETURNS TABLE(clienteid INTEGER, fechaorden date) AS
$$
BEGIN
 RETURN QUERY
    SELECT O.clienteid, O.fechaorden
    FROM clientes C, ordenes O
    where C.clienteid =O.clienteid AND C.cedula_ruc = p_ruc;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE FUNCTION SP_CLIENTE_INS(
p_id int,
p_ruc varchar,
p_nombre varchar,
p_contacto varchar,
p_direccion varchar
)
RETURNS void AS $$
BEGIN
  INSERT INTO clientes(clienteid, cedula_ruc, nombrecia, nombrecontacto, direccioncli)
  VALUES (p_id, p_ruc, p_nombre,p_contacto,p_direccion);
END;
$$ LANGUAGE plpgsql;

 -- ejecutando en postgresql
   SELECT SP_CLIENTE_INS(1001, '1044182707', 'Teofilo Chambilla', 'Teofilo','');
```

Código en python para leer el procedimiento almacenado

```
#!/usr/bin/python
import psycopg2
def get_clientes():
    conn = None
    try:
        # conectarse a la base de datos
        conn = psycopg2.connect(database='tienda', user='
   postgres', password='teo1421')
        # crear un cursor object para la ejecucion
        cur = conn.cursor()
        # llamar al procedimiento almacenado
        cur.callproc('get_cliente_by_ruc', ('1298765477',))
        # procesar el result set
        row = cur.fetchone()
        while row is not None:
            print(row)
            row = cur.fetchone()
        # cerrar la comunicacion con el servidor de
   PostgreSQL
        cur.close()
    except (Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if conn is not None:
            conn.close()

print(get_clientes())
```

Resultado esperado

```
(2, datetime.date(2007, 6, 13))
(2, datetime.date(2007, 6, 17))
(2, datetime.date(2007, 6, 18))
```

P1. En base de datos generalemente usamos cuatro funciones básicas de manipulación de datos: Create, Read, Update and Delete (CRUD). Se le pide crear cuatro procedimientos almacenados CRUD sobre la tabla clientes:

- SP_CLIENTE_INS: Procedimiento para insertar un nuevo cliente.
- SP_CLIENTE_SEL: Procedimiento para lee los datos de clientes, recibe como condición de búsqueda el RUC.

- SP_CLIENTE_UPD: Procedimiento para cambiar los datos de un cliente, recibe como condición de busqueda el RUC.
- SP_CLIENTE_DEL: Procedimiento para eliminar a un cliente, recibe como condición de busqueda el RUC.

Entregable El entregable son los scripts de los procedimientos almacenados con sentencias de prueba de funcionalidad.