PL/SQL – Práctica Final II

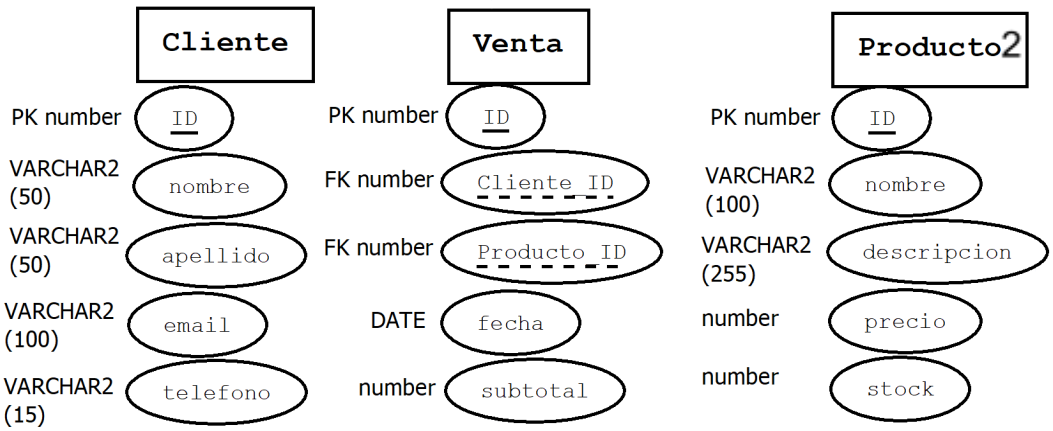
Ampliación de paquetes; uso de procedures y trigger compuesto

---

En este caso, trabajaremos con tres entidades: **Producto2, Cliente** y **Venta.**
A continuación el código SQL para crearlas:

```
CREATE TABLE Cliente (
        ID NUMBER PRIMARY KEY,
        Nombre VARCHAR2(50),
        Apellido VARCHAR2(50),
        Email VARCHAR2(100),
        Telefono VARCHAR2(15)
);
        – No hay columna cantidad; cada venta será sólo de un producto
CREATE TABLE Venta (
        ID NUMBER PRIMARY KEY,
        Cliente_ID NUMBER,
        Producto_ID NUMBER,
        Fecha DATE,
        Subtotal NUMBER,
        FOREIGN KEY (Cliente_ID) REFERENCES Cliente(ID),
        FOREIGN KEY (Producto_ID) REFERENCES Producto(ID)
);

CREATE TABLE Producto2 (
        ID NUMBER PRIMARY KEY,
        Nombre VARCHAR2(100),
        Descripcion VARCHAR2(255),
        Precio NUMBER,
        Stock NUMBER
);
```



Paquete para Gestión de Ventas

Implementa un paquete **PKG_VENTA** para trabajar con la tabla Ventas, el paquete tendrá los siguientes procedimientos y triggers:

1. Procedimiento para agregar una nueva venta con todos sus datos.
   ** Antes de insertar, comprobar que el ID de venta indicado no está en la base de datos, y que las FK (producto y cliente) existan en el sistema.*
   *** [Se recomienda usar funciones locales para estas comprobaciones]*

2. Procedimiento para mostrar por consola (DBMS_OUTPUT) aquellas ventas realizadas en un día y mes específicos (pasados por parámetros IN).
   ** PISTA: Usa la función EXTRACT* de Oracle SQL

3. Hacer un procedimiento similar al anterior, pero que inserte cada venta en una tabla aparte. El nombre de la tabla será libre, pero deberá incluir el día y el mes pasados por parámetro. P.Ej: ***ventas_d8m5*** para el día 8 y el mes 5)
   * Las columnas para la nueva tabla son ID_venta y su fecha asociada.
   *** Se deberán usar sentencias DDL (CREATE TABLE …).*
   **** Investiga antes sobre las consultas dinámicas: crear objetos SQL (como tablas o vistas) usando variables dentro de bloques PL/SQL para facilitarte el trabajo ;D*

*[ALTERNATIVA EJERCICIO 3: Si es muy complejo esto o es difícil de investigar para la fase 1 de formación, insertar aquí las funciones a usar para crear la tabla en vez de las pistas ** y ***]:*

```
C/C++
CREATE OR REPLACE FUNCTION crear_Tabla(nombre_tab IN VARCHAR2) RETURN BOOLEAN is
BEGIN
       EXECUTE IMMEDIATE 'CREATE TABLE ' || nombre_tab || ' (id_Venta NUMBER, fecha
DATE)';

       --Si todo va bien, devolverá TRUE
       RETURN TRUE;
       EXCEPTION
       WHEN OTHERS THEN
       DBMS_OUTPUT.PUT_LINE(SQLCODE || ': ' || SQLERRM); RETURN FALSE;
END;
/
```

```
C/C++
CREATE OR REPLACE FUNCTION INSERTAR_EN_TABLA(nom_tabla IN VARCHAR2, valores IN
VARCHAR2) RETURN BOOLEAN IS
```

```
  sql_stmt VARCHAR2(500); --Un varchar2 de longitud 500 nos permitirá usar cualquier
sentencia SQL por larga que sea
BEGIN
  -- Construyendo la sentencia SQL dinámicamente; pasara los valores del parametro,
que deben estar separados por coma
  sql_stmt := 'INSERT INTO ' || nom_tabla || ' VALUES (' || valores || ')';

  -- Ejecutando la sentencia SQL
  EXECUTE IMMEDIATE sql_stmt;
  --Si todo va bien, devolver TRUE
  RETURN TRUE;
EXCEPTION
  WHEN OTHERS THEN
        -- Manejo de errores
        DBMS_OUTPUT.PUT_LINE('Error: ' || SQLERRM); RETURN FALSE;
END;
/
```

4. Trigger COMPUESTO de tipo FILA que compruebe el stock actual en la tabla Producto ***antes*** de que se realice cualquier venta, y que ACTUALICE dicho stock en caso de haber suficiente ***después*** de realizar cualquier venta.
   ** Si el stock es menor o igual a 0, lanzar un error/excepción.*