# TRABAJO PRÁCTICO MODELO DE PUNTO DE VENTA

**AÑO 2021**

# Integrantes del grupo

|  |  |  |  |
| --- | --- | --- | --- |
| **Cédula** | **Apellido** | **Nombre** | **Sección** |
| **6129422** | **Amarilla Leguizamón** | **Oscar Alexander** | **NA** |
| **4659580** | **Gómez Cardenas** | **Eduardo David** | **NA** |
| **4662520** | **Medina Recalde** | **Oliver Isaías** | **NA** |

**PARTE 4**

# TRIGGERS, OBJETOS Y DINÁMICO

1. Implemente con TRIGGERS las siguientes reglas de negocio:
   1. Al **INSERTAR** sobre la tabla **D\_PAGO\_OPERACION**
      1. Verifica si ya se ha aplicado algún pago con el mismo ID de operación. Si ya se ha aplicado un pago, se muestra el mensaje “La operación ya ha sido pagada”, y se aborta la operación.
      2. Recorre la tabla **D\_DETALLE\_OPERACIONES** correspondiente a la operación que se paga.
      3. Por cada registro:
         1. Accede a la tabla **D\_PRODUCTOS\_MEDIDA\_PRECIO** considerando el producto, la unidad de medida, la sucursal y la forma de pago y obtiene el PRECIO.
         2. Si no existe un registro, mantiene los valores actuales. Si existe un registro, reemplaza los valores de la tabla **D\_DETALLE\_OPERACIONES** de la siguiente manera:
            * El precio\_operacion: con el correspondiente precio de la tabla **D\_PRODUCTO\_MEDIDA\_PRECIO**
            * Importe\_descuento: ( precio\_operacion X cantidad) X el porcentaje de descuento correspondiente a la forma de pago/100
            * Importe\_recargo : ( precio\_operacion X cantidad) X porcentaje de recargo correspondiente a la forma de pago

/100

* + - * + Importe\_operacion: (precio\_operacion X cantidad\_operacion)
        + Importe\_iva: (importe\_operacion- importe\_descuento+ importe\_recargo ) / divisor\_iva\_incluido
      1. Una vez completado el recorrido, asigna el importe\_pago con la sumatoria del importe\_operacion + importe\_recargo – importe\_descuento.
      2. Si la forma de pago requiere comprobante, debe estar asignado el nro de referencia\_pago
      3. Si la forma de pago es “CHEQUE”, deberá tener asignado el código de banco con un banco existente, el nro\_Cuenta y el nro \_cheque
  1. No se permite BORRAR O MODIFICAR sobre la tabla **D\_PAGO \_OPERACIÓN**
  2. Sobre la tabla **D\_DETALLE\_OPERACIONES** no se permite modificar o borrar si la operación ya tiene un registro de pago.

-------------------------------------------

CREATE OR REPLACE TRIGGER item1\_a\_i

AFTER INSERT ON D\_PAGO\_OPERACION

FOR EACH ROW

DECLARE

TYPE pago\_operacion IS TABLE OF d\_pago\_operacion%ROWTYPE;

v\_pago\_operacion pago\_operacion;

BEGIN

SELECT \*

BULK COLLECT INTO v\_pago\_operacion

FROM d\_pago\_operacion

WHERE id\_operacion = :new.id\_operacion;

IF v\_pago\_operacion.COUNT > 0 THEN

dbms\_output.put\_line('Movimiento ya pagado');

END IF;

END item1\_a;

---------------------------------------------------

CREATE OR REPLACE TRIGGER item1\_a\_iii

AFTER INSERT ON D\_PAGO\_OPERACION

FOR EACH ROW

DECLARE

-- Tablas a recorrer

TYPE detalle\_operaciones IS TABLE OF D\_DETALLE\_OPERACIONES%ROWTYPE;

v\_detalle\_operaciones detalle\_operaciones;

TYPE productos\_medida\_precio IS TABLE OF D\_PRODUCTOS\_MEDIDA\_PRECIO%ROWTYPE;

v\_productos\_medida\_precio productos\_medida\_precio;

-- Porcentajes

TYPE FORMA\_PAGO IS TABLE OF D\_FORMA\_PAGO%ROWTYPE;

porcentaje FORMA\_PAGO;

TYPE TIPO\_IVA IS TABLE OF D\_TIPO\_IVA%ROWTYPE;

divisor TIPO\_IVA;

-- Consulta interna

consulta\_productos VARCHAR(500);

consulta\_detalles VARCHAR(400);

consulta\_porcentajes VARCHAR(400);

consulta\_update VARCHAR(400);

-- Input variables

v\_id\_pago\_operacion NUMBER;

v\_id\_forma\_pago NUMBER;

-- Indices

j NUMBER;

i NUMBER;

-- Variables de proceso

precio\_operacion NUMBER;

importe\_descuento NUMBER;

importe\_recargo NUMBER;

importe\_operacion NUMBER;

importe\_iva NUMBER;

-- Sumatoria

sum\_importe\_operacion NUMBER;

sum\_importe\_descuento NUMBER;

sum\_importe\_recargo NUMBER;

BEGIN

-- test

v\_id\_pago\_operacion := :NEW.id\_operacion;

v\_id\_forma\_pago := :NEW.cod\_forma\_pago;

---------

--

sum\_importe\_operacion := 0;

sum\_importe\_descuento := 0;

sum\_importe\_recargo := 0;

--

-- Obtener detalle de operaciones

-- parametros:

-- id\_operacion

consulta\_detalles := 'SELECT \*

FROM D\_DETALLE\_OPERACIONES

WHERE id\_operacion = :1';

EXECUTE IMMEDIATE consulta\_detalles

BULK COLLECT INTO v\_detalle\_operaciones

USING v\_id\_pago\_operacion;

i := v\_detalle\_operaciones.FIRST;

WHILE i <= v\_detalle\_operaciones.LAST

LOOP -- Se recorre el cursor de detalle\_operaciones

-- Obtener precio de producto

-- parametros:

-- id\_producto

-- cod\_medida

-- cod\_forma\_pago

-- cod\_sucursal

consulta\_productos := 'SELECT \*

FROM d\_productos\_medida\_precio m

WHERE m.id\_producto = :1

AND m.cod\_medida = :2

AND m.cod\_forma\_pago = :3

AND m.cod\_sucursal = (

SELECT COD\_SUCURSAL FROM D\_MOVIMIENTO\_OPERACIONES

WHERE ID\_OPERACION = :4

)';

EXECUTE IMMEDIATE consulta\_productos

BULK COLLECT INTO v\_productos\_medida\_precio

USING v\_detalle\_operaciones(i).id\_producto,

v\_detalle\_operaciones(i).cod\_medida,

v\_id\_forma\_pago,

v\_detalle\_operaciones(i).id\_operacion;

-- Obtener porcentajes de forma de pago

consulta\_porcentajes := 'SELECT \*

FROM d\_forma\_pago WHERE cod\_forma\_pago = :1';

EXECUTE IMMEDIATE consulta\_porcentajes

BULK COLLECT INTO porcentaje

USING v\_id\_forma\_pago;

i := v\_detalle\_operaciones.FIRST;

WHILE i <= v\_detalle\_operaciones.LAST

LOOP -- Se recorre el cursor de detalle\_operaciones

-- Obtener precio de producto

-- parametros:

-- id\_producto

-- cod\_medida

-- cod\_forma\_pago

-- cod\_sucursal

consulta\_productos := 'SELECT \*

FROM d\_productos\_medida\_precio m

WHERE m.id\_producto = :1

AND m.cod\_medida = :2

AND m.cod\_forma\_pago = :3

AND m.cod\_sucursal = (

SELECT COD\_SUCURSAL FROM D\_MOVIMIENTO\_OPERACIONES

WHERE ID\_OPERACION = :4

)';

EXECUTE IMMEDIATE consulta\_productos

BULK COLLECT INTO v\_productos\_medida\_precio

USING v\_detalle\_operaciones(i).id\_producto,

v\_detalle\_operaciones(i).cod\_medida,

v\_id\_forma\_pago,

v\_detalle\_operaciones(i).id\_operacion;

-- Obtener porcentajes de forma de pago

consulta\_porcentajes := 'SELECT \*

FROM d\_forma\_pago WHERE cod\_forma\_pago = :1';

EXECUTE IMMEDIATE consulta\_porcentajes

BULK COLLECT INTO porcentaje

USING v\_id\_forma\_pago;

-- Obtener divisor de iva

consulta\_porcentajes := 'SELECT \*

FROM d\_tipo\_iva WHERE cod\_tipo\_iva = :1';

EXECUTE IMMEDIATE consulta\_porcentajes

BULK COLLECT INTO divisor

USING v\_detalle\_operaciones(i).cod\_tipo\_iva;

j := v\_productos\_medida\_precio.FIRST;

WHILE j <= v\_productos\_medida\_precio.LAST LOOP -- Se recorre el cursor de productos\_medida\_precio

DBMS\_OUTPUT.PUT\_LINE('- Detalle operacion ------- id: ' || v\_detalle\_operaciones(i).id\_operacion);

DBMS\_OUTPUT.PUT\_LINE('ID producto: '||v\_detalle\_operaciones(i).id\_producto);

DBMS\_OUTPUT.PUT\_LINE('Cantidad: '||v\_detalle\_operaciones(i).cantidad\_operacion);

DBMS\_OUTPUT.PUT\_LINE('iva: '||divisor(divisor.first).divisor\_iva\_incluido);

--b.1

precio\_operacion := v\_productos\_medida\_precio(j).precio\_venta;

--b.2

importe\_descuento := (precio\_operacion \* v\_detalle\_operaciones(i).cantidad\_operacion) \* (porcentaje(porcentaje.FIRST).porcentaje\_descuento / 100);

--b.3

importe\_recargo := (precio\_operacion \* v\_detalle\_operaciones(i).cantidad\_operacion) \* (porcentaje(porcentaje.FIRST).porcentaje\_recargo / 100);

--b.4

importe\_operacion := precio\_operacion \* v\_detalle\_operaciones(i).cantidad\_operacion;

--b.5

importe\_iva := (importe\_operacion - importe\_descuento + importe\_recargo) / (divisor(divisor.FIRST).divisor\_iva\_incluido);

DBMS\_OUTPUT.PUT\_LINE('Producto\_medida\_precio------');

DBMS\_OUTPUT.PUT\_LINE('--- Precio: '||v\_productos\_medida\_precio(j).precio\_venta);

DBMS\_OUTPUT.PUT\_LINE('--- Importe descuento: '||importe\_descuento);

DBMS\_OUTPUT.PUT\_LINE('--- Importe recargo: '||importe\_recargo);

DBMS\_OUTPUT.PUT\_LINE('--- Importe operacion: '||importe\_operacion);

DBMS\_OUTPUT.PUT\_LINE('--- Importe iva: '||importe\_iva);

DBMS\_OUTPUT.PUT\_LINE('---------------------------');

-------------------------------------------

CREATE OR REPLACE TRIGGER item1\_a\_iii\_d

BEFORE INSERT ON D\_PAGO\_OPERACION

FOR EACH ROW WHEN (NEW.NRO\_REFERENCIA\_PAGO IS NULL)

DECLARE

v\_require\_nro\_comprobante NUMBER;

BEGIN

SELECT requiere\_nro\_comprobante INTO v\_require\_nro\_comprobante

FROM D\_FORMA\_PAGO

WHERE COD\_FORMA\_PAGO = :NEW.COD\_FORMA\_PAGO;

if v\_require\_nro\_comprobante = 1 then

dbms\_output.put\_line('El tipo de forma de pago seleccionado requiere un número de comprobante.');

end if;

END;

-------------------------------------------

--b.5

importe\_iva := (importe\_operacion - importe\_descuento + importe\_recargo) / (divisor(divisor.FIRST).divisor\_iva\_incluido);

DBMS\_OUTPUT.PUT\_LINE('Producto\_medida\_precio------');

DBMS\_OUTPUT.PUT\_LINE('--- Precio: '||v\_productos\_medida\_precio(j).precio\_venta);

DBMS\_OUTPUT.PUT\_LINE('--- Importe descuento: '||importe\_descuento);

DBMS\_OUTPUT.PUT\_LINE('--- Importe recargo: '||importe\_recargo);

DBMS\_OUTPUT.PUT\_LINE('--- Importe operacion: '||importe\_operacion);

DBMS\_OUTPUT.PUT\_LINE('--- Importe iva: '||importe\_iva);

DBMS\_OUTPUT.PUT\_LINE('---------------------------');

-- Sumatoria

sum\_importe\_operacion := sum\_importe\_operacion + importe\_operacion;

sum\_importe\_descuento := sum\_importe\_descuento + importe\_descuento;

sum\_importe\_recargo := sum\_importe\_recargo + importe\_recargo;

j:= v\_productos\_medida\_precio.NEXT(j);

END LOOP;

IF sum\_importe\_operacion <> 0 THEN

-- mostrar sumatoria

DBMS\_OUTPUT.PUT\_LINE('Sumatoria');

DBMS\_OUTPUT.PUT\_LINE('--- Importe recargo: '||sum\_importe\_recargo);

DBMS\_OUTPUT.PUT\_LINE('--- Importe operacion: '||sum\_importe\_operacion);

DBMS\_OUTPUT.PUT\_LINE('--- Importe descuento: '||sum\_importe\_descuento);

-- reset sumatoria

sum\_importe\_operacion := 0;

sum\_importe\_descuento := 0;

sum\_importe\_recargo := 0;

DBMS\_OUTPUT.PUT\_LINE('---------------------------');

DBMS\_OUTPUT.PUT\_LINE(' ');

-- update importe\_operacion en la tabla d\_pago\_operacion

consulta\_update := 'UPDATE D\_PAGO\_OPERACION

SET IMPORTE\_PAGO = :1

WHERE ID\_OPERACION = :2';

EXECUTE IMMEDIATE consulta\_update

USING sum\_importe\_operacion, v\_detalle\_operaciones(i).id\_operacion;

END IF;

i:= v\_detalle\_operaciones.NEXT(i);

END LOOP;

END;

CREATE OR REPLACE TRIGGER item1\_a\_iii\_e

BEFORE INSERT ON D\_PAGO\_OPERACION

FOR EACH ROW WHEN (NEW.NRO\_CUENTA IS NULL OR NEW.NRO\_CHEQUE IS NULL OR COD\_BANCO IS NULL)

DECLARE

forma\_pago NUMBER;

BEGIN

SELECT desc\_forma\_pago INTO forma\_pago

FROM D\_FORMA\_PAGO

WHERE COD\_FORMA\_PAGO = :NEW.COD\_FORMA\_PAGO;

if forma\_pago = 'CHEQUE' then

dbms\_output.put\_line('El tipo de forma de pago seleccionado requiere un número NRO\_CUENTA y NRO\_CHEQUE.');

end if;

END;

------------------------------------------

CREATE OR REPLACE TRIGGER T\_PAGOS

BEFORE DELETE OR UPDATE ON D\_PAGO\_OPERACION

FOR EACH ROW

BEGIN

IF DELETING OR UPDATING THEN

RAISE\_APPLICATION\_ERROR(-20011,'NO SE PERMITE ELIMINAR DATOS DE ESTA TABLA');

END IF;

END;

/

-------------------------------------------

CREATE OR REPLACE TRIGGER NO\_MODIFICAR\_DETALLE\_OPERACIONES

BEFORE DELETE OR UPDATE ON D\_DETALLE\_OPERACIONES

FOR EACH ROW

DECLARE

V\_ID\_OPERACION NUMBER(12) := 0;

BEGIN

SELECT ID\_OPERACION INTO V\_ID\_OPERACION FROM D\_PAGO\_OPERACION

WHERE ID\_OPERACION = :NEW.ID\_OPERACION;

IF V\_ID\_OPERACION > 0 THEN

RAISE\_APPLICATION\_ERROR(-2009,'NO SE PERMITE MODIFICAR O ELIMINAR DATOS DE ESTA TABLA');

END IF;

EXCEPTION WHEN NO\_DATA\_FOUND THEN

NULL;

END;

## SQL Dinámico:

* 1. El siguiente script crea una tabla para auditar las sentencias DML sobre las tablas:

CREATE TABLE LOG\_TABLAS

(FECHA\_HORA TIMESTAMP,

OPERACION VARCHAR2(10),

NOMBRE\_TABLA VARCHAR2(30),

CLAVE VARCHAR2(2000),

USUARIO VARCHAR2(30))

* + 1. Cree la tabla con el script proporcionado.
    2. Cree el procedimiento **P\_GENERAR\_TRIGGERS** que recorrerá las tablas del esquema y creará dinámicamente un trigger con la denominación T\_<nombre\_tabla> que deberá dispararse para todas las tablas después de la inserción, borrado o modificación de datos. EL trigger deberá grabar la fecha de operación, la operación que dispara el DML, el nombre de la tabla, el usuario que generó la operación, y en el campo CLAVE irá la columna que conforma la PK. Si la PK está conformada por varias columnas, las mismas irán entre comas.

CREATE OR REPLACE PROCEDURE "P\_GENERAR\_TRIGGERS"

AUTHID CURRENT\_USER AS

TYPE T\_CUR IS REF CURSOR;

TYPE R\_TAB IS RECORD(

NOMBRE\_TABLA VARCHAR2(200),

NOMBRE\_PK VARCHAR2(200)

);

TYPE T\_TAB IS TABLE OF

R\_TAB INDEX BY BINARY\_INTEGER;

l\_sql VARCHAR2(4000);

l\_dummy NUMBER;

l\_trigger\_name VARCHAR2(30);

V\_TAB T\_TAB;

V\_CUR T\_CUR;

ind NUMBER;

v\_tablas VARCHAR2(2000);

ind\_ante NUMBER;

CURSOR C\_TABLAS IS

SELECT cols.table\_name, LISTAGG( cols.column\_name, ',')

WITHIN GROUP(

ORDER BY cols.column\_name) AS COLUMN\_NAME

FROM all\_constraints cons, all\_cons\_columns cols

WHERE cols.table\_name in (SELECT T.TABLE\_NAME

FROM ALL\_TABLES T

WHERE T.OWNER = 'BASEDATOS2'

AND T.TABLE\_NAME LIKE 'D\_%')

AND cons.constraint\_type = 'P'

AND cons.constraint\_name = cols.constraint\_name

AND cons.owner = cols.owner

GROUP BY COLS.TABLE\_NAME

order by table\_name;

BEGIN

FOR REG IN C\_TABLAS LOOP

l\_trigger\_name := 'T\_' || REG.TABLE\_NAME;

l\_sql :=

'CREATE OR replace TRIGGER ' || l\_trigger\_name ||

' BEFORE INSERT OR UPDATE OR DELETE ON ' || TO\_CHAR(REG.TABLE\_NAME) ||

' FOR EACH ROW

DECLARE

OPERACION VARCHAR2(100);

BEGIN

IF INSERTING THEN

OPERACION := ' || q'['INSERTAR']' || ';

ELSIF UPDATING THEN

OPERACION := ' || q'['ACTUALIZAR']' || ';

ELSIF DELETING THEN

OPERACION := ' || q'['BORRADO']' || ';

END IF;

INSERT INTO LOG\_TABLAS(FECHA\_HORA, OPERACION, NOMBRE\_TABLA, CLAVE, USUARIO)

VALUES(sysdate, OPERACION, ''' || TO\_CHAR(REG.TABLE\_NAME) ||''', ''' || TO\_CHAR(REG.COLUMN\_NAME) ||''', user );

END;';

EXECUTE IMMEDIATE l\_sql;

END LOOP;

END;

/

## Objetos.

* 1. Cree el directorio DIR\_VENTA. Copie las imágenes proveídas en el presente ejercicio.
  2. Cree el tipo tabla anidada T\_STOCK compuesto de los siguientes elementos:
* COD\_SUCURSAL NUMBER(2)
* EXISTENCIA NUMBER(10,2)
  1. Cree el objeto T\_PROD con los siguientes elementos:
     1. Los atributos:
* ID\_PRODUCTO NUMBER(12)
* DESC\_ABREVIADO VARCHAR2(20)
* PORCENTAJE\_IVA NUMBER(3)
* PRECIO NUMBER(10)
* IMAGEN\_PRODUCTO BLOB
* EXISTENCIA T\_STOCK
  + 1. El método estático INSTANCIAR\_PRODUCTO que recibe como parámetro ID de producto y devuelve un objeto del tipo T\_PROD. El método deberá obtener los datos del producto, el porcentaje de iva, el precio (precio de última compra adicionando el resultado de aplicar el porcentaje de beneficio). Así mismo, asignará la imagen a partir del directorio DIR\_VENTA. La imagen tiene como nombre el código del producto rellenado con ceros a la izquierda y la extensión jpeg.
    2. El método miembro ASIGNAR\_EXISTENCIA que permitirá asignar la existencia en todas las sucursales.
  1. Cree la tabla de objetos D\_PRODUCTOS2 constituida de elementos T\_PROD.
  2. En un PL/SQL anónimo, recorra los productos, instancie un objeto del tipo T\_PROD con su constructor INSTANCIAR\_PRODUCTO pasando como parámetro el id\_producto, e inserte el objeto en la tabla D\_PRODUCTOS2.

**Observación:** En el directorio DIR\_VENTA existen imágenes para los productos con ID 1 al 20. Para probar, restrinja su recorrido de productos hasta 20.

/\*

3. Objetos.

A. Cree el directorio DIR\_VENTA. Copie las imágenes proveídas en el presente

ejercicio.

\*/

CREATE DIRECTORY DIR\_VENTA AS 'C:\Users\olome\Documents\Facultad\quintoSemestre\BDII\sql\TPs\TP - PARTE 4-20211021\DIR\_VENTA' ;

GRANT READ, WRITE ON DIRECTORY DIR\_VENTA TO IMP\_FULL\_DATABASE;

GRANT READ,WRITE ON DIRECTORY DIR\_VENTA TO EXP\_FULL\_DATABASE;

/\*

B. Cree el tipo tabla anidada T\_STOCK compuesto de los siguientes elementos:

o COD\_SUCURSAL NUMBER(2)

o EXISTENCIA NUMBER(10,2)

\*/

CREATE OR REPLACE TYPE L\_EXIST AS OBJECT (

COD\_SUCURSAL NUMBER(2),

EXISTENCIA NUMBER(10,2)

);

CREATE TYPE T\_STOCK AS TABLE OF L\_EXIST;

/\*

C. Cree el objeto T\_PROD con los siguientes elementos:

i. Los atributos:

o ID\_PRODUCTO NUMBER(12)

o DESC\_ABREVIADO VARCHAR2(20)

o PORCENTAJE\_IVA NUMBER(3)

o PRECIO NUMBER(10)

o IMAGEN\_PRODUCTO BLOB

o EXISTENCIA T\_STOCK

\*/

CREATE OR REPLACE TYPE T\_PROD IS OBJECT

(

ID\_PRODUCTO NUMBER(12),

DESC\_ABREVIADO VARCHAR2(20),

PORCENTAJE\_IVA NUMBER(3),

PRECIO NUMBER(10),

IMAGEN\_PRODUCTO BLOB,

EXISTENCIA T\_STOCK,

MEMBER FUNCTION ASIGNAR\_EXISTENCIA RETURN T\_STOCK,

STATIC FUNCTION INSTANCIAR\_PRODUCTO(v\_id IN NUMBER) RETURN T\_PROD

);

/\*

ii. El método estático INSTANCIAR\_PRODUCTO que recibe como parámetro ID de

producto y devuelve un objeto del tipo T\_PROD. El método deberá obtener los

datos del producto, el porcentaje de iva, el precio (precio de última compra

adicionando el resultado de aplicar el porcentaje de beneficio). Así mismo,

asignará la imagen a partir del directorio DIR\_VENTA. La imagen tiene como

nombre el código del producto rellenado con ceros a la izquierda y la extensión

jpeg.

\*/

/\*creamos una tabla que contedra las imagenes de los productos para luego insertar dentro del metodo estatico y

a su vez seleccionar dicha imagen para pasar como parametro del atributoto IMAGEN\_PRODUCTO\*/

CREATE TABLE imagenes (

id NUMBER,

blob\_data BLOB

);

CREATE OR REPLACE TYPE BODY T\_PROD IS

STATIC FUNCTION INSTANCIAR\_PRODUCTO(v\_id IN NUMBER) RETURN T\_PROD IS

retorno T\_PROD;

nom\_img varchar2(100);

v\_id\_prod NUMBER;

v\_cod\_sucursal NUMBER(2);

v\_cant\_existencia NUMBER(10,2);

v\_desc\_abrv D\_PRODUCTOS.DESC\_ABREVIADO%TYPE;

v\_porcentaje\_iva D\_TIPO\_IVA.PORCENTAJE\_IVA%TYPE;

v\_precio NUMBER(10,2);

l\_existencia T\_STOCK:= T\_STOCK();

indice NUMBER(8);

v\_bfile BFILE;

v\_blob BLOB;

r\_blob BLOB;

CURSOR c\_cod\_suc is

SELECT COD\_SUCURSAL, CANTIDAD\_EXISTENCIA FROM D\_STOCK\_SUCURSAL WHERE ID\_PRODUCTO = 1 AND ID\_PRODUCTO <= 20;

BEGIN

SELECT ID\_PRODUCTO INTO v\_id\_prod FROM D\_PRODUCTOS WHERE ID\_PRODUCTO = v\_id AND ID\_PRODUCTO <= 20;

-- SELECT COD\_SUCURSAL INTO v\_cod\_sucursal FROM D\_STOCK\_SUCURSAL WHERE ID\_PRODUCTO = v\_id AND ID\_PRODUCTO <= 20;

-- SELECT CANTIDAD\_EXISTENCIA INTO v\_cant\_existencia FROM D\_STOCK\_SUCURSAL WHERE ID\_PRODUCTO = v\_id AND ID\_PRODUCTO <= 20;

SELECT P.DESC\_ABREVIADO INTO v\_desc\_abrv

FROM D\_PRODUCTOS P WHERE P.ID\_PRODUCTO = v\_id AND P.ID\_PRODUCTO <= 20;

SELECT I.PORCENTAJE\_IVA INTO v\_porcentaje\_iva

FROM D\_TIPO\_IVA I

JOIN D\_PRODUCTOS P

ON P.COD\_TIPO\_IVA = I.COD\_TIPO\_IVA

WHERE P.ID\_PRODUCTO= v\_id AND P.ID\_PRODUCTO <= 20;

SELECT (P.PRECIO\_ULTIMA\_COMPRA + P.PRECIO\_ULTIMA\_COMPRA \* P.PORCENTAJE\_BENEFICIO) PRECIO\_COMPRA INTO v\_precio

FROM D\_PRODUCTOS P WHERE P.ID\_PRODUCTO = v\_id AND P.ID\_PRODUCTO <= 20;

SELECT LPAD(TO\_CHAR(P.ID\_PRODUCTO), 2, '0') ID INTO nom\_img FROM D\_PRODUCTOS P WHERE P.ID\_PRODUCTO <= 20 AND P.ID\_PRODUCTO = v\_id;

nom\_img := nom\_img || '.jpg';

indice := 0;

for i in c\_cod\_suc loop

if(indice <= l\_existencia.LAST) THEN

l\_existencia.EXTEND;

l\_existencia(indice) := L\_EXIST(i.COD\_SUCURSAL, i.CANTIDAD\_EXISTENCIA);

indice := indice + 1;

end if;

end loop;

INSERT INTO imagenes (id, blob\_data)

VALUES (v\_id, empty\_blob())

RETURNING blob\_data INTO v\_blob;

v\_bfile := BFILENAME('DIR\_VENTA', nom\_img);

DBMS\_LOB.FILEOPEN(v\_bfile, DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADFROMFILE(v\_blob, v\_bfile, DBMS\_LOB.GETLENGTH(v\_bfile) );

DBMS\_LOB.FILECLOSE(v\_bfile);

SELECT blob\_data INTO r\_blob FROM imagenes WHERE id = v\_id;

retorno := T\_PROD( v\_id\_prod, v\_desc\_abrv, v\_porcentaje\_iva, v\_precio, r\_blob , l\_existencia );

RETURN retorno;

END INSTANCIAR\_PRODUCTO;

/\*

iii. El método miembro ASIGNAR\_EXISTENCIA que permitirá asignar la existencia

en todas las sucursales.

\*/

MEMBER FUNCTION ASIGNAR\_EXISTENCIA RETURN T\_STOCK IS

CURSOR C\_SUCURSAL IS

SELECT \* FROM D\_STOCK\_SUCURSAL;

TYPE existencias IS TABLE OF T\_STOCK;

l\_existencia T\_STOCK:= T\_STOCK();

BEGIN

FOR SUC IN C\_SUCURSAL LOOP

l\_existencia := T\_STOCK(L\_EXIST(SUC.COD\_SUCURSAL, SUC.CANTIDAD\_EXISTENCIA) );

RETURN l\_existencia;

END LOOP;

END ASIGNAR\_EXISTENCIA;

END;

--Cree la tabla de objetos D\_PRODUCTOS2 constituida de elementos T\_PROD.

CREATE TABLE D\_PRODUCTOS2 OF T\_PROD

NESTED TABLE EXISTENCIA STORE AS PRODUCTOS\_TAB;

/\*

En un PL/SQL anónimo, recorra los productos, instancie un objeto del tipo T\_PROD

con su constructor INSTANCIAR\_PRODUCTO pasando como parámetro el

id\_producto, e inserte el objeto en la tabla D\_PRODUCTOS2.

\*/

DECLARE

CURSOR C\_PRODUCTOS IS

SELECT ID\_PRODUCTO FROM D\_PRODUCTOS

WHERE ID\_PRODUCTO <= 20;

valores T\_PROD;

BEGIN

FOR I IN C\_PRODUCTOS LOOP

valores := T\_PROD.INSTANCIAR\_PRODUCTO(i.ID\_PRODUCTO);

INSERT INTO D\_PRODUCTOS2 VALUES( valores );

END LOOP;

END;

/

CREATE TABLE ordenes\_tab OF T\_PROD

NESTED TABLE productos STORE AS productos\_tab ;

CREATE OR REPLACE TYPE T\_PROD IS OBJECT

(

ID\_PRODUCTO NUMBER(12),

DESC\_ABREVIADO VARCHAR2(20),

PORCENTAJE\_IVA NUMBER(3),

PRECIO NUMBER(10),

IMAGEN\_PRODUCTO BLOB,

EXISTENCIA T\_STOCK,

MEMBER FUNCTION ASIGNAR\_EXISTENCIA RETURN T\_STOCK,

STATIC FUNCTION INSTANCIAR\_PRODUCTO(v\_id IN NUMBER) RETURN T\_PROD

);

/\*

ii. El método estático INSTANCIAR\_PRODUCTO que recibe como parámetro ID de

producto y devuelve un objeto del tipo T\_PROD. El método deberá obtener los

datos del producto, el porcentaje de iva, el precio (precio de última compra

adicionando el resultado de aplicar el porcentaje de beneficio). Así mismo,

asignará la imagen a partir del directorio DIR\_VENTA. La imagen tiene como

nombre el código del producto rellenado con ceros a la izquierda y la extensión

jpeg.

\*/

/\*creamos una tabla que contedra las imagenes de los productos para luego insertar dentro del metodo estatico y

a su vez seleccionar dicha imagen para pasar como parametro del atributoto IMAGEN\_PRODUCTO\*/

CREATE TABLE imagenes (

id NUMBER,

blob\_data BLOB

);

CREATE OR REPLACE TYPE BODY T\_PROD IS

STATIC FUNCTION INSTANCIAR\_PRODUCTO(v\_id IN NUMBER) RETURN T\_PROD IS

retorno T\_PROD;

nom\_img varchar2(100);

v\_id\_prod NUMBER;

v\_cod\_sucursal NUMBER(2);

v\_cant\_existencia NUMBER(10,2);

v\_desc\_abrv D\_PRODUCTOS.DESC\_ABREVIADO%TYPE;

v\_porcentaje\_iva D\_TIPO\_IVA.PORCENTAJE\_IVA%TYPE;

v\_precio NUMBER(10,2);

l\_existencia T\_STOCK:= T\_STOCK();

indice NUMBER(8);

v\_bfile BFILE;

v\_blob BLOB;

r\_blob BLOB;

CURSOR c\_cod\_suc is

SELECT COD\_SUCURSAL, CANTIDAD\_EXISTENCIA FROM D\_STOCK\_SUCURSAL WHERE ID\_PRODUCTO = 1 AND ID\_PRODUCTO <= 20;

BEGIN

SELECT ID\_PRODUCTO INTO v\_id\_prod FROM D\_PRODUCTOS WHERE ID\_PRODUCTO = v\_id AND ID\_PRODUCTO <= 20;

-- SELECT COD\_SUCURSAL INTO v\_cod\_sucursal FROM D\_STOCK\_SUCURSAL WHERE ID\_PRODUCTO = v\_id AND ID\_PRODUCTO <= 20;

-- SELECT CANTIDAD\_EXISTENCIA INTO v\_cant\_existencia FROM D\_STOCK\_SUCURSAL WHERE ID\_PRODUCTO = v\_id AND ID\_PRODUCTO <= 20;

SELECT P.DESC\_ABREVIADO INTO v\_desc\_abrv

FROM D\_PRODUCTOS P WHERE P.ID\_PRODUCTO = v\_id AND P.ID\_PRODUCTO <= 20;

SELECT I.PORCENTAJE\_IVA INTO v\_porcentaje\_iva

FROM D\_TIPO\_IVA I

JOIN D\_PRODUCTOS P

ON P.COD\_TIPO\_IVA = I.COD\_TIPO\_IVA

WHERE P.ID\_PRODUCTO= v\_id AND P.ID\_PRODUCTO <= 20;

SELECT (P.PRECIO\_ULTIMA\_COMPRA + P.PRECIO\_ULTIMA\_COMPRA \* P.PORCENTAJE\_BENEFICIO) PRECIO\_COMPRA INTO v\_precio

FROM D\_PRODUCTOS P WHERE P.ID\_PRODUCTO = v\_id AND P.ID\_PRODUCTO <= 20;

SELECT LPAD(TO\_CHAR(P.ID\_PRODUCTO), 2, '0') ID INTO nom\_img FROM D\_PRODUCTOS P WHERE P.ID\_PRODUCTO <= 20 AND P.ID\_PRODUCTO = v\_id;

nom\_img := nom\_img || '.jpg';

indice := 0;

for i in c\_cod\_suc loop

if(indice <= l\_existencia.LAST) THEN

l\_existencia.EXTEND;

l\_existencia(indice) := L\_EXIST(i.COD\_SUCURSAL, i.CANTIDAD\_EXISTENCIA);

indice := indice + 1;

end if;

end loop;

INSERT INTO imagenes (id, blob\_data)

VALUES (v\_id, empty\_blob())

RETURNING blob\_data INTO v\_blob;

v\_bfile := BFILENAME('DIR\_VENTA', nom\_img);

DBMS\_LOB.FILEOPEN(v\_bfile, DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADFROMFILE(v\_blob, v\_bfile, DBMS\_LOB.GETLENGTH(v\_bfile) );

DBMS\_LOB.FILECLOSE(v\_bfile);

SELECT blob\_data INTO r\_blob FROM imagenes WHERE id = v\_id;

retorno := T\_PROD( v\_id\_prod, v\_desc\_abrv, v\_porcentaje\_iva, v\_precio, r\_blob , l\_existencia );

RETURN retorno;

END INSTANCIAR\_PRODUCTO;

/\*

iii. El método miembro ASIGNAR\_EXISTENCIA que permitirá asignar la existencia

en todas las sucursales.

\*/

MEMBER FUNCTION ASIGNAR\_EXISTENCIA RETURN T\_STOCK IS

CURSOR C\_SUCURSAL IS

SELECT \* FROM D\_STOCK\_SUCURSAL;

TYPE existencias IS TABLE OF T\_STOCK;

l\_existencia T\_STOCK:= T\_STOCK();

BEGIN

FOR SUC IN C\_SUCURSAL LOOP

l\_existencia := T\_STOCK(L\_EXIST(SUC.COD\_SUCURSAL, SUC.CANTIDAD\_EXISTENCIA) );

RETURN l\_existencia;

END LOOP;

END ASIGNAR\_EXISTENCIA;

END;

--Cree la tabla de objetos D\_PRODUCTOS2 constituida de elementos T\_PROD.

CREATE TABLE D\_PRODUCTOS2 OF T\_PROD

NESTED TABLE EXISTENCIA STORE AS PRODUCTOS\_TAB;

/\*

En un PL/SQL anónimo, recorra los productos, instancie un objeto del tipo T\_PROD

con su constructor INSTANCIAR\_PRODUCTO pasando como parámetro el

id\_producto, e inserte el objeto en la tabla D\_PRODUCTOS2.

\*/

DECLARE

CURSOR C\_PRODUCTOS IS

SELECT ID\_PRODUCTO FROM D\_PRODUCTOS

WHERE ID\_PRODUCTO <= 20;

valores T\_PROD;

BEGIN

FOR I IN C\_PRODUCTOS LOOP

valores := T\_PROD.INSTANCIAR\_PRODUCTO(i.ID\_PRODUCTO);

INSERT INTO D\_PRODUCTOS2 VALUES( valores );

END LOOP;

END;

/

CREATE TABLE ordenes\_tab OF T\_PROD

NESTED TABLE productos STORE AS productos\_tab ;

SELECT (P.PRECIO\_ULTIMA\_COMPRA + P.PRECIO\_ULTIMA\_COMPRA \* P.PORCENTAJE\_BENEFICIO) PRECIO\_COMPRA INTO v\_precio

FROM D\_PRODUCTOS P WHERE P.ID\_PRODUCTO = v\_id AND P.ID\_PRODUCTO <= 20;

SELECT LPAD(TO\_CHAR(P.ID\_PRODUCTO), 2, '0') ID INTO nom\_img FROM D\_PRODUCTOS P WHERE P.ID\_PRODUCTO <= 20 AND P.ID\_PRODUCTO = v\_id;

nom\_img := nom\_img || '.jpeg';

indice := 0;

for i in c\_cod\_suc loop

if(indice <= l\_existencia.LAST) THEN

l\_existencia.EXTEND;

l\_existencia(indice) := L\_EXIST(i.COD\_SUCURSAL, i.CANTIDAD\_EXISTENCIA);

indice := indice + 1;

end if;

end loop;

INSERT INTO imagenes (id, blob\_data)

VALUES (v\_id, empty\_blob())

RETURNING blob\_data INTO v\_blob;

v\_bfile := BFILENAME('DIR\_VENTA', nom\_img);

DBMS\_LOB.FILEOPEN(v\_bfile, DBMS\_LOB.FILE\_READONLY);

DBMS\_LOB.LOADFROMFILE(v\_blob, v\_bfile, DBMS\_LOB.GETLENGTH(v\_bfile) );

DBMS\_LOB.FILECLOSE(v\_bfile);

SELECT blob\_data INTO r\_blob FROM imagenes WHERE id = v\_id;

retorno := T\_PROD( v\_id\_prod, v\_desc\_abrv, v\_porcentaje\_iva, v\_precio, r\_blob , l\_existencia );

RETURN retorno;

END INSTANCIAR\_PRODUCTO;

/\*

iii. El método miembro ASIGNAR\_EXISTENCIA que permitirá asignar la existencia

en todas las sucursales.

\*/

MEMBER FUNCTION ASIGNAR\_EXISTENCIA RETURN T\_STOCK IS

CURSOR C\_SUCURSAL IS

SELECT \* FROM D\_STOCK\_SUCURSAL;

TYPE existencias IS TABLE OF T\_STOCK;

l\_existencia T\_STOCK:= T\_STOCK();

BEGIN

FOR SUC IN C\_SUCURSAL LOOP

l\_existencia := T\_STOCK(L\_EXIST(SUC.COD\_SUCURSAL, SUC.CANTIDAD\_EXISTENCIA) );

RETURN l\_existencia;

END LOOP;

END ASIGNAR\_EXISTENCIA;

END;

--Cree la tabla de objetos D\_PRODUCTOS2 constituida de elementos T\_PROD.

CREATE TABLE D\_PRODUCTOS2 OF T\_PROD

NESTED TABLE EXISTENCIA STORE AS PRODUCTOS\_TAB;

/\*

En un PL/SQL anónimo, recorra los productos, instancie un objeto del tipo T\_PROD

con su constructor INSTANCIAR\_PRODUCTO pasando como parámetro el

id\_producto, e inserte el objeto en la tabla D\_PRODUCTOS2.

\*/

DECLARE

CURSOR C\_PRODUCTOS IS

SELECT ID\_PRODUCTO FROM D\_PRODUCTOS

WHERE ID\_PRODUCTO <= 20;

valores T\_PROD;

BEGIN

FOR I IN C\_PRODUCTOS LOOP

valores := T\_PROD.INSTANCIAR\_PRODUCTO(i.ID\_PRODUCTO);

INSERT INTO D\_PRODUCTOS2 VALUES( valores );

END LOOP;

END;

/

|  |  |  |  |
| --- | --- | --- | --- |
| **CRITERIOS** | | | |
| **1 Triggers** | A Trigger INSERT s/ pagos | Sintaxis Gral | 1 |
| i Verificación | 2 |
| ii Recálculo | 1 |
| iii a Recuperación | 0,5 |
| iii b Cálculo | 1,5 |
| iii c Asignación | 0,5 |
| iii d Requiere comprobante | 1 |
| iii e Si es cheque | 1 |
| B Trigger DELETE/UPDATE s/ pagos | | 1 |
| C Trigger DELETE/UPDATE s/ detalles | | 1 |
| **2 SQL**  **Dinámico** | A Generación de script | Lógica | 2,5 |
| Sintaxis | 2,5 |
| **3 Objetos** | A Directory | | 0,5 |
| B Tabla anidada | | 0,5 |
| C Tipo Producto | i Especificación | 1 |
| ii Método estático | 2 |
| iii Método Miembro | 1 |
| D Tabla de objetos | | 0,5 |
| E Inserción | | 1 |
| **TOTAL** | | | **22** |