Ejercicios PL/SQL – Funciones y Procedimientos

Dada la siguiente BD:

```
CREATE TABLE CLIENTE (
        cod NUMBER,
        nombre varchar2(100),
        direccion varchar2(100),
        país varchar2(100) DEFAULT 'ESPAÑA',
        CONSTRAINT PK_client PRIMARY KEY(cod)
);
```

```
CREATE TABLE PEDIDO  (
        cod NUMBER,
        cliente NUMBER,
        fecha2 DATE,
        total NUMBER(6,2),
        CONSTRAINT PK_order PRIMARY KEY(cod),
        CONSTRAINT FK_ord_cli FOREIGN KEY (cliente) REFERENCES CLIENTE
) ;
```

```
CREATE TABLE LINPEDIDO    (
        codPedido NUMBER,
        numPed NUMBER,
        cantidad NUMBER,
        descripcion VARCHAR2(20),
        precioUnidad NUMBER(6,2),
        CONSTRAINT PK_li_order PRIMARY KEY(LINECOD, numPed),
        CONSTRAINT FK_ord_lineord FOREIGN KEY(numPed) REFERENCES
PEDIDO
) ;
```

```
INSERT INTO CLIENT (cod, nombre, direccion, pais)
VALUES (1, 'Manolo', 'C/Saigon 5','ESPAÑA');
INSERT INTO CLIENT (cod, nombre, direccion, pais)
VALUES (2, 'Carlos', 'C/Silvestre Stalone 3','ANDORRA');
INSERT INTO CLIENT (cod, nombre, direccion, pais)
VALUES (3, 'Jose', 'C/Rambo 4','ESPAÑA');
INSERT INTO CLIENT (cod, nombre, direccion, pais)
VALUES (4, 'Armando', 'C/Acorralado 2','ANDORRA');
```

```
INSERT INTO C_ORDER (cod, cliente, fecha2, total)
VALUES (1,1,TO_DATE('08/02/2016', 'dd/mm/yyyy')+dbms_random.value(1, 50),50);
INSERT INTO C_ORDER (cod, cliente, fecha2, total)
VALUES (2,1,TO_DATE('08/02/2016', 'dd/mm/yyyy')+dbms_random.value(1, 50),60);
INSERT INTO C_ORDER (cod, cliente, fecha2, total)
VALUES (3,2,TO_DATE('08/02/2016', 'dd/mm/yyyy')+dbms_random.value(1, 50),70);
```

```
INSERT INTO ORDER_LINE (LINECOD, C_ORDER, cantidad, descripcion, precioUnidad)
VALUES (1,1,2,'ZAPATILLAS CORRER',125);
INSERT INTO ORDER_LINE
(LINECOD,C_ORDER,QUANTITY,DESCRIPTION,UNIT_PRICE)
VALUES (2,1,3,'CALCETINES RUNNER',53);
INSERT INTO ORDER_LINE
(LINECOD,C_ORDER,QUANTITY,DESCRIPTION,UNIT_PRICE)
VALUES (3,1,1,'BOTAS DE AGUA',34);
Insert into ORDER_LINE (LINECOD,C_ORDER,QUANTITY,DESCRIPTION,UNIT_PRICE)
```

```
VALUES (1,2,1,'TELEFONO NOQUIA',55);
INSERT INTO ORDER_LINE
(LINECOD,C_ORDER,QUANTITY,DESCRIPTION,UNIT_PRICE)
VALUES (2,2,5,'GALLETAS PRINCIPE',5);
INSERT INTO ORDER_LINE
(LINECOD,C_ORDER,QUANTITY,DESCRIPTION,UNIT_PRICE)
VALUES (1,3,3,'LIBROS DE ESTUDIO',3.5);
INSERT INTO ORDER_LINE
(LINECOD,C_ORDER,QUANTITY,DESCRIPTION,UNIT_PRICE)
VALUES (2,3,5,'RULOS PARA EL PELO',13.5);
```

Ejercicios Procedimientos

1. Dado el siguiente procedimiento:

```
CREATE OR REPLACE PROCEDURE crear_depart (
    v_num_dept depart.dept_no%TYPE,
    v_dnombre depart.dnombre%TYPE DEFAULT 'PROVISIONAL',
    v_loc depart.loc2%TYPE DEFAULT 'PROVISIONAL')
AS
BEGIN
    INSERT INTO depart VALUES (v_num_dept, v_dnombre, v_loc);
END crear_depart;
```

Indicar cuáles de las siguientes llamadas son correctas y cuáles incorrectas. En el caso de que sean incorrectas, escribir la llamada correcta usando la notación posicional, siempre que sea posible:

crear_aeparc ;

crear_depart(50) ;

crear_depart('COMPRAS') ;

crear_depart(50,'COMPRAS') ;

crear_depart('COMPRAS', 50);

crear_depart('COMPRAS', 'VALENCIA') ;

crear_depart(50, 'COMPRAS', 'VALENCIA') ;

crear_depart('COMPRAS', 50, 'VALENCIA');

crear_depart('VALENCIA', 'COMPRAS') ;

crear_depart('VALENCIA', 50);

2. Crea un procedimiento que recibiendo un numero de pedido muestre el código y nombre del comprador, la fecha de pedido y el importe del pedido.

3. Declara una función que devuelva el número de líneas de pedido, para ello habrá que pasar por parámetro el pedido.

4. Declara una función que dado un número de pedido devuelva el importe del pedido, para ello tiene que sumar las líneas de dicho pedido sin usar ningún cursor.

5. Usando la función anterior, crea un procedimiento que corrija el importe de todos los pedidos. NOTA: SABEMOS QUE LOS PEDIDOS SON UNA SECUENCIA DESDE 1 HASTA EL NUMERO DE PEDIDOS. NO HAY QUE USAR CURSORES.

Ejercicios Cursores

1. Crea un procedimiento almacenado que usando un cursor realice el cálculo del importe de los pedidos a partir de las líneas de pedido, usando la DB anterior.

2. A partir de la BD anterior, crea dos nuevas tablas:

FACTURAS -> Esta tabla contendrá las facturas creadas a partir de los pedidos. Esta tabla será igual que la de los pedidos pero vendrá incluido del IVA, la columna se llamará importe_iva (NUMBER(6,2)) y la forma de pago (VARCHAR2(20)).

LINEA_FACTURA -> Esta tabla contendrá las líneas de cada factura. Será idéntica a la de Linea_Pedidos, salvo que añadirá un nuevo campo llamado importe_iva (NUMBER(6,2))

3. Crea un procedimiento almacenado que copie los datos de todos los pedidos a la tabla de facturas.

Además copiará la tabla de líneas de pedido sobre la de líneas de factura. Y hazlo usando cursores.

4. Modifica el procedimiento anterior añadiendo una excepción, que si detecta una clave primaria repetida, muestre un mensaje por pantalla de error y realice un ROLLBACK.

5. Crea un procedimiento almacenado que usando un cursor calcule de cada línea de factura el importe del IVA.

Esto lo hará pasando el código de una factura por parámetro. Para hacer este procedimiento almacenado usa el bucle WHILE – END LOOP.

Además, crea un cursor FOR UPDATE, que será el que utilices para actualizar la información.

6. Crea un procedimiento almacenado que usando un cursor calcule de cada factura el importe del IVA de esa factura, para ello habrá que sumar el importe de las líneas de esa factura usando el procedimiento anterior.

Este procedimiento tiene que recorrer todas las líneas de cada factura, calculando su importe total de IVA.

7. Crea un procedimiento anterior que haga todo lo anterior del siguiente modo:

- A este procedimiento se le pasará un rango de fechas sobre las que hay que generar las facturas a partir de los pedidos. Se pasarán 2 fechas la primera de inicio y la segunda de fin.
- Para generar las facturas, usar un cursor que recorra los pedidos entre el rango de fechas y otro cursor para que de cada pedido recorra sus líneas calculando el importe del IVA de cada línea, y al final calcule el IVA total de la factura. Con estos dos cursores se tienen que ir copiando los datos del pedido a la factura al mismo tiempo que se van calculando los importes del iva, y el importe de la factura sin iva.
- Si el importe de la factura sin IVA es inferior a 500€ en el campo forma_pago se le indicará ‘Contado’ y si es mayor o igual de 500€ se indicará ‘Transferencia’.

8. Modifica el procedimiento almacenado anterior para que se añada el control de Transacciones, de tal forma que cuando se estén copiando las líneas de un pedido a una factura se confirme al finalizar con éxito la copia de toda la factura completa y sino que se deshaga el trabajo para ese pedido en concreto, y que continúe con el siguiente pedido.

Ejercicios Triggers

1. A partir de las tablas LINEA_PEDIDO y PEDIDO de las prácticas anteriores.

Escribir un disparador de base de datos que haga fallar cualquier operación de modificación o inserción de la tabla PEDIDOS cuya fecha sea inferior a la fecha actual.

2. Crea un Trigger para que cuando se modifique, se inserte o se borre una línea de la tabla de LINEA_PEDIDO se actualice el importe del PEDIDO.

NOTA: Un Trigger no puede contener instrucciones que consulten o modifiquen tablas mutantes.

Una tabla mutante es aquella que está siendo modificada por una sentencia UPDATE, DELETE o INSERT.

3. Crea una tabla llamada LOG_SISTEMA con los campos: ID, USUARIO, FECHA, EVENTO.

A partir de esta tabla crea 2 Triggers que almacene la siguiente información de eventos en el esquema:

- Que quede registrado el usuario y fecha a la que el usuario accede al sistema
- Que quede registrado cualquier acción de creación de algún objeto o tabla, de este se tiene que registrar qué objeto es el que se es involucrado almacenándose esta información en el campo EVENTO.