

Mario Rubio Avila BD Tarea 6.1

Junto a este fichero se entrega cada uno de los ficheros para la ejecución de estas. Todo se distribuye en funciones y con una función main(bloque anonimo) que ayuda a comprobar los métodos y a probarlos.

APARTADO A

Ejercicio 1 (Fichero Tarea6.1_1.1.sql)

Funcion main

```
--Funcion Main
DECLARE
  V_pedidosCodigo pedidos.num%type := &NumeroDePedido;
BEGIN
  DBMS_OUTPUT.PUT_LINE('Actividad 1.1');
  --PedidoExiste(V_pedidosCodigo);
  if PedidoExiste(V_pedidosCodigo) then
    DBMS_OUTPUT.PUT_LINE('El pedido ' || V_pedidosCodigo || ' existe');
  else
    DBMS_OUTPUT.PUT_LINE('El pedido ' || V_pedidosCodigo || ' no existe');
  end if;
END;
/
```

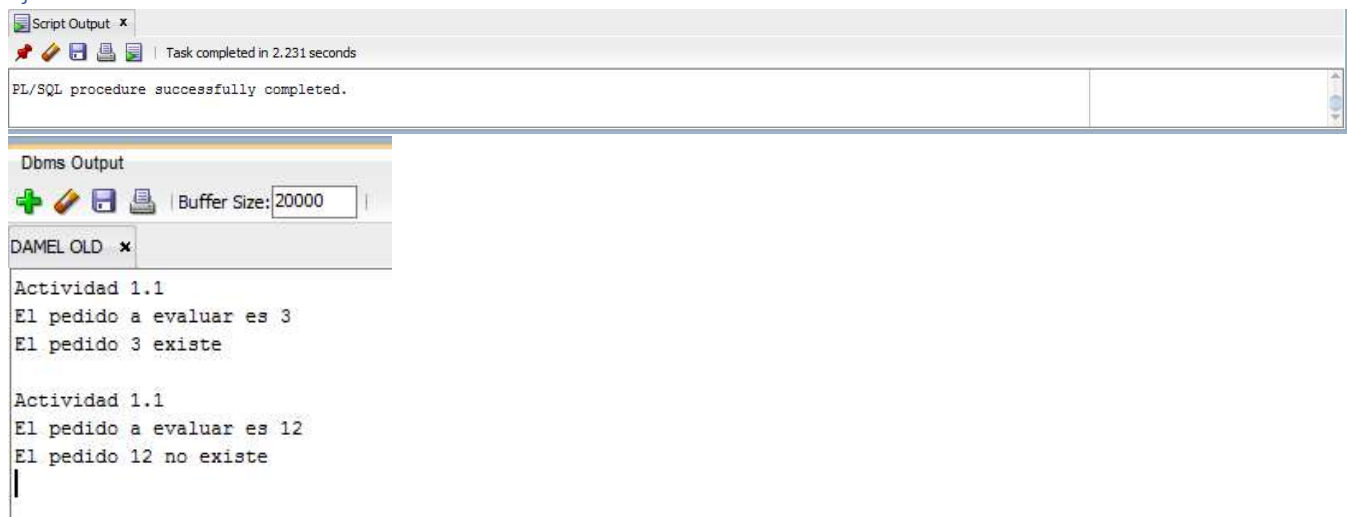
Funcion ExistePedido

```
--Funcion Pedido Existe
Create or replace function PedidoExiste (V_Codigo pedidos.num%type)
return boolean
as
  V_codigoCapturado pedidos.num%type;
  --V_resultado boolean :=true ;
BEGIN
  DBMS_OUTPUT.PUT_LINE('El pedido a evaluar es ' || V_Codigo);
  select num into V_codigoCapturado
  from pedidos
  where num=V_Codigo;
  return true;
exception
  when no_data_found then
    return false;
END;
/
```

Compilación



Ejecucion



Ejercicio 2 (Fichero Tarea6.1_1.2.sql)

Funcion main

```
--SELECT * FROM PEDIDOS; --Descomentar para ver la tabla Pedidos
--Funcion Main
DECLARE
    V_pedidosCodigo pedidos.num%type := &NumeroDePedido;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Actividad 2.1');
    if PedidoExiste(V_pedidosCodigo) then --Si existe el pedido funcion Ejercicio 1.1 hace lo siguiente
        mostrarPedido(getPedido(V_pedidosCodigo)); --Cojo el pedido llamado a la funcion y se lo paso al procedimiento mostrar
    else --Si no existe el pedido muestra el siguiente mensaje
        DBMS_OUTPUT.PUT_LINE('El pedido ' || V_pedidosCodigo || ' no existe');
    end if;
END;
/
```

Funcion PedidoExiste

```
--Funcion Ejercicio 1.1 (Solo comento la linea de texto)
Create or replace function PedidoExiste (V_Codigo pedidos.num%type)
return boolean
as
    V_codigoCapturado pedidos.num%type;
BEGIN
    --DBMS_OUTPUT.PUT_LINE('El pedido a evaluar es ' || V_Codigo);
    select num into V_codigoCapturado
        from pedidos
        where num=V_Codigo;
    return true;
exception
    when no_data_found then
        return false;
END;
/
```

Funcion getPedido

```
--FUNCION Ejercicio 1.2
--Coge el row del pedido
--NUM  FECHA  GASTOS_ENVIO  FECHA_PREVISTA  TOTAL  CLIENTE
Create or replace function getPedido (V_Codigo pedidos.num%type)
return pedidos%rowtype
as
    V_pedido pedidos%rowtype;
begin
    select * into V_pedido
        from pedidos
        where num=V_Codigo;
    return V_pedido;
exception --Nunca se ejecutara porque de esto ya esta evaluado en Pedido Existe
    when no_data_found then
        DBMS_OUTPUT.PUT_LINE('ERROR DATO NO ENCONTRADO');
    return null;
END;
/
```

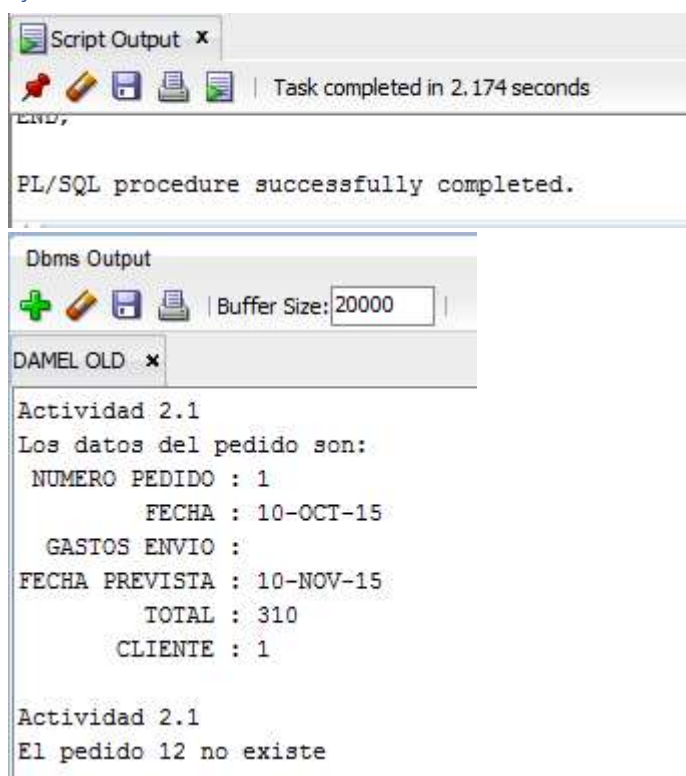
Procedimiento mostrarPedido

```
--PROCEDIMIENTO Ejercicio 1.2
--Mostrara toda la informacion pedido
--NUM  FECHA  GASTOS_ENVIO  FECHA_PREVISTA  TOTAL  CLIENTE
Create or replace procedure mostrarPedido (V_pedido pedidos%rowtype)
as
begin
    DBMS_OUTPUT.PUT_LINE('Los datos del pedido son: ');
    DBMS_OUTPUT.PUT_LINE(' NUMERO PEDIDO : ' || v_pedido.NUM);
    DBMS_OUTPUT.PUT_LINE('          FECHA : ' || v_pedido.FECHA);
    DBMS_OUTPUT.PUT_LINE(' GASTOS ENVIO : ' || v_pedido.GASTOS_ENVIO);
    DBMS_OUTPUT.PUT_LINE(' FECHA PREVISTA : ' || v_pedido.FECHA_PREVISTA);
    DBMS_OUTPUT.PUT_LINE('          TOTAL : ' || v_pedido.TOTAL);
    DBMS_OUTPUT.PUT_LINE('          CLIENTE : ' || v_pedido.CLIENTE);
exception --Nunca se ejecutara porque de esto ya esta evaluado en Pedido Existe
when no_data_found then
    DBMS_OUTPUT.PUT_LINE('ERROR DATO NO ENCONTRADO');
END;
/
```

Compilacion



Ejecución



Ejercicio 3 (Fichero Tarea6.1_1.3.sql)

Bloque anonimo de test del procedimiento

```
--SELECT * FROM CLIENTES; --Descomentar para ver la tabla Clientes
--Funcion Main
DECLARE
    V_codigoCliente clientes.codigo%type := &CodigoCliente;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Actividad 1.3');
    if clienteExiste(V_codigoCliente) then
        mostrarCliente(V_codigoCliente);--Vamos a mostrar el cliente
    else --Si no existe el pedido muestra el siguiente mensaje
        DBMS_OUTPUT.PUT_LINE('El pedido ' || V_codigoCliente || ' no existe');
    end if;
END;
/
```

Función clienteExiste

Esta función no se pide, pero ayuda a hacer un programa de pruebas que no intente mostrar algo que no existe.

```
--Funcion Ejercicio 1.3
--Comprueba si existe el cliente
Create or replace function clienteExiste (V_Codigo clientes.codigo%type)
return boolean
as
    V_codigoCapturado clientes.codigo%type;
BEGIN
    select codigo into V_codigoCapturado
        from clientes
        where codigo=V_Codigo;
    return true;
exception
    when no_data_found then
        return false;
END;
/
```

Procedimiento mostrarClientes

```
--PROCEDIMIENTO Ejercicio 1.3
--Mostrara toda la informacion del cliente
--CODIGO NOMBRE APELLIDOS EDAD
Create or replace procedure mostrarCliente (V_Codigo clientes.codigo%type)
as
    V_cliente clientes%rowtype;
begin
    select * into V_cliente
        from clientes
        where codigo=V_Codigo;

    DBMS_OUTPUT.PUT_LINE('Los datos del cliente son: ');
    DBMS_OUTPUT.PUT_LINE('    CODIGO : ' || V_cliente.codigo);
    DBMS_OUTPUT.PUT_LINE('    NOMBRE : ' || V_cliente.nombre);
    DBMS_OUTPUT.PUT_LINE('    APELLIDOS : ' || V_cliente.apellidos);
    DBMS_OUTPUT.PUT_LINE('    EDAD : ' || V_cliente.edad);
exception --Nunca se ejecutara porque de esto ya esta evaluado en Pedido Existe
    when no_data_found then
        DBMS_OUTPUT.PUT_LINE('ERROR DATO NO ENCONTRADO');
END;
/
```

Compilacion


```
Script Output x
Task completed in 0.72 seconds

Function CLIENTEEXISTE compiled
```

```
Script Output x
Task completed in 20.131 seconds

Procedure MOSTRARCLIENTE compiled
```

Ejecución

```
Script Output x
Task completed in 2.97 seconds

PL/SQL procedure successfully completed.
```

Actividad 1.3
Los datos del cliente son:
CODIGO : 1
NOMBRE : Luis
APELLIDOS : Garcia Perez
EDAD : 30

Actividad 1.3
El pedido 9 no existe

Ejercicio 4 (Fichero Tarea6.1_1.4.sql)

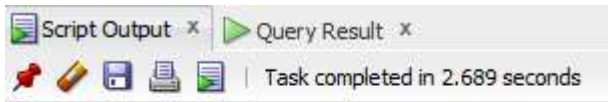
Bloque anónimo Main para la comprobación de la función

```
--Bloque anonimo Main
DECLARE
    V_pedidosCodigo pedidos.num$type := sNumeroDePedido;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Actividad 1.4');
    mostrarPedidoDetallado(V_pedidosCodigo);
END;
/
```

Procedimiento de mostrar pedido

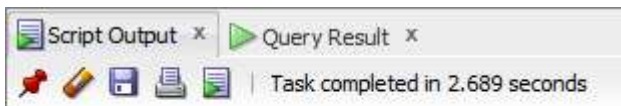
```
Create or replace procedure mostrarPedidoDetallado (V_pedidosCodigo pedidos.num$type)
as
--V_pedidosCodigo pedidos.num$type := sNumeroDePedido;
cursor lineaPedidoCursor is
select LINEAS.NUM, LINEAS.CANTIDAD, LINEAS.IMPORTE, PRODUCTOS.NOMBRE, PRODUCTOS.PRECIO
from LINEAS , PRODUCTOS
where NUM_PEDIDO = V_pedidosCodigo AND PRODUCTOS.CODIGO = lineas.producto;
V_pedidoLinea lineaPedidoCursor%rowtype;
BEGIN
if PedidoExiste(V_pedidosCodigo) then --Si existe el pedido buscamos las lineas que lo componen
    DBMS_OUTPUT.PUT_LINE('NºLinea ' || 'NombreProducto' || ' Precio ' || ' Cantidad ' || ' Importe ');
    for registro in lineaPedidoCursor loop
        DBMS_OUTPUT.PUT_LINE(registro.NUM || ' ' || registro.NOMBRE || ' ' || registro.PRECIO || ' ' || registro.CANTIDAD || ' ' || registro.IMPORTE);
    end loop;
else
    DBMS_OUTPUT.PUT_LINE('El pedido ' || V_pedidosCodigo || ' no existe');
end if;
END;
/
```

Compilación



Procedure MOSTRARPEDIDODETALLADO compiled

Ejecución



PL/SQL procedure successfully completed.

Actividad 1.4

NºLinea	NombreProducto	Precio	Cantidad	Importe
1	PANTALON	50	2	100
2	VESTIDO C	80	1	80
3	CAMISA M/L	65	2	130

Actividad 1.4

El pedido 15 no existe

Ejercicio 5 (Fichero Tarea6.1_1.5.sql)

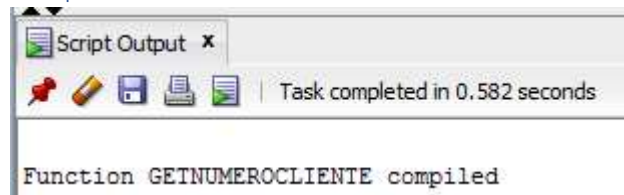
Bloque anonimo de test del procedimiento

```
--Bloque anonimo
DECLARE
    V_pedidosCodigo pedidos.num%type := sNumeroDePedido;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Actividad 1.5');
    if PedidoExiste(V_pedidosCodigo) then
        mostrarPedido(getPedido(V_pedidosCodigo));
        mostrarpedidodetallado(V_pedidosCodigo);
        --Cojo el numero de pedido y solicito el numero cliente que lo hizo y lo inserto en mostrarcliente
        mostrarcliente( GETNUMEROCLIENTE(V_pedidosCodigo));
    else
        DBMS_OUTPUT.PUT_LINE('El pedido ' || V_pedidosCodigo || ' no existe');
    end if;
END;
/
```

Función auxiliar creada para recuperar el código cliente dado un pedido

```
--Dado un numero pedido devuelve el numero de cliente que lo realizo
Create or replace function GETNUMEROCLIENTE (V_Codigo pedidos.num%type)
return pedidos.cliente%type
as
    V_codigoClienteCapturado pedidos.cliente%type;
BEGIN
    select pedidos.cliente into V_codigoClienteCapturado
    from pedidos
    where num=V_Codigo;
    return V_codigoClienteCapturado;
exception
    when no_data_found then
        return null;
END;
/
```

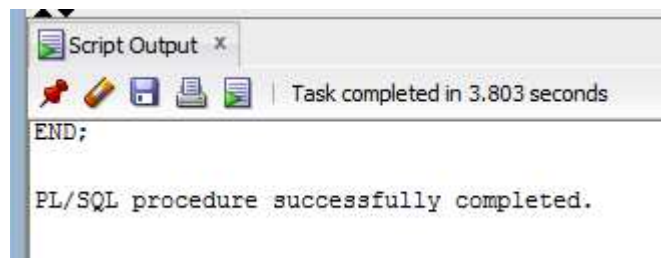
Compilación



```
Script Output x
Task completed in 0.582 seconds

Function GETNUMEROCLIENTE compiled
```

Ejecución



```
Script Output x
Task completed in 3.803 seconds

END;

PL/SQL procedure successfully completed.
```

Actividad 1.5

Los datos del pedido son:

```
NUMERO PEDIDO : 1
FECHA : 10-OCT-15
GASTOS ENVIO :
FECHA PREVISTA : 10-NOV-15
TOTAL : 310
CLIENTE : 1
```

N°Linea	NombreProducto	Precio	Cantidad	Importe
1	PANTALON	50	2	100
2	VESTIDO C	80	1	80
3	CAMISA M/L	65	2	130

Los datos del cliente son:

```
CODIGO : 1
NOMBRE : Luis
APELLIDOS : Garcia Perez
EDAD : 30
```

Actividad 1.5

El pedido 15 no existe

Ejercicio 6

Se capturaron las excepciones en cada uno de los ejercicios, Por ejemplo, en los procedimientos de mostrar cliente o pedido.

```
exception --Nunca se ejecutara porque de esto ya esta evaluado en Pedido Existe
when no_data_found then
DBMS_OUTPUT.PUT_LINE('ERROR DATO NO ENCONTRADO');
return null;
```

O esta otra que nos devuelve un falso la función si no existe el pedido o cliente.

```
exception
when no_data_found then
return false;
```

APARTADO B

1. Cada vez que se vaya a insertar o modificar una línea de un pedido debe de actualizarse correctamente el importe de la misma (cantidad X precio del producto).` (Fichero Tarea6.1_2.1.sql)

Bloque anonimo de test del procedimiento

```
DECLARE
    V_pedidosCodigo pedidos.num$type := sNumeroDePedido;
    V_numeroLinea lineas.num_pedido$type := sNumeroDeLinea;
    V_Cantidad lineas.cantidad$type := sCantidad;
    V_IDProducto lineas.producto$type := sIDdelProducto;

    --Creacion de una excepcion propia
    CHECK_CONSTRAINT_VIOLATED EXCEPTION;
    PRAGMA EXCEPTION_INIT(CHECK_CONSTRAINT_VIOLATED, -2291); --Excepcion "integrity constraint (%s.%s) violated - parent key not found"
BEGIN
    DBMS_OUTPUT.PUT_LINE('Actividad 2.1');
    if PedidoExiste(V_pedidosCodigo) then --Existe el pedido
        DBMS_OUTPUT.PUT_LINE('El pedido ' || V_pedidosCodigo || ' antes de la modificacion.');
```

mostrarpedidodetallado(V_pedidosCodigo);

```
        if LineaExiste(V_numeroLinea,V_pedidosCodigo) then --Existe la linea vamos a actualizarla
            update lineas set cantidad = V_Cantidad, producto = V_IDProducto where num_pedido=1 and num=1;
        else --No existe vamos a crearla
            INSERT INTO LINEAS VALUES (V_numeroLinea, V_pedidosCodigo, V_IDProducto, V_Cantidad, null);
        end if;
        DBMS_OUTPUT.PUT_LINE('El pedido despues de la modificacion.');
```

mostrarpedidodetallado(V_pedidosCodigo);

```
    else
        DBMS_OUTPUT.PUT_LINE('El pedido ' || V_pedidosCodigo || ' no existe');
    end if;
    EXCEPTION
        WHEN CHECK_CONSTRAINT_VIOLATED THEN --Excepcion "integrity constraint (%s.%s) violated - parent key not found"
            DBMS_OUTPUT.PUT_LINE('ERROR LA REFERENCIA DE PRODUCTO NO EXISTE. SALTA EXCEPCION DE PRODUCTO. REGISTRE ANTES DEL PRODUCTO');
END;
```

/

Funcion para comprobar si la linea existe

```
CREATE OR REPLACE FUNCTION LineaExiste (V_Codigo lineas.num$type,V_Pedido lineas.num_pedido$type)
return boolean
as
    V_codigoCapturado lineas.num$type;
BEGIN
    select num into V_codigoCapturado
    from lineas
    where num=V_Codigo and num_pedido = V_Pedido;
    return true;
exception
    when no_data_found then
        return false;
END;
```

/

Trigger que actualiza el precio si se actualiza o inserta una linea de una factura

```
-- Cada vez que se vaya a insertar o modificar una línea de un pedido debe de actualizarse correctamente el importe de la misma (cantidad X precio del producto).
CREATE OR REPLACE TRIGGER modificacionLineasPedido
BEFORE INSERT OR UPDATE ON LINEAS
FOR EACH ROW
DECLARE
    NEWPRECIO productos.precio$type;
begin
    SELECT PRECIO INTO NEWPRECIO from productos where codigo=:new.producto;
    :new.importe := :new.cantidad * NEWPRECIO;

    exception
    when no_data_found then
        DBMS_OUTPUT.PUT_LINE('ATENCION ESTAS HACIENDO REFERENCIA A UN PRODUCTO QUE NO EXISTE. REVISE ' || :NEW.producto || '. PRECIO VALOR A 0.');
```

:new.importe := 0;

```
end;
```


Ejecución

Actividad 2.1

El pedido 1 antes de la modificación.

NºLinea	NombreProducto	Precio	Cantidad	Importe
1	PANTALON	50	2	100
2	VESTIDO C	80	1	80
3	CAMISA M/L	65	2	130
4	FALDA CORTA	45	2	90

El pedido después de la modificación.

NºLinea	NombreProducto	Precio	Cantidad	Importe
1	PANTALON	50	10	500
2	VESTIDO C	80	1	80
3	CAMISA M/L	65	2	130
4	FALDA CORTA	45	2	90

2. Cada vez que se inserten, se borren o modifiquen líneas hay que actualizar el importe del pedido correspondiente (Fichero Tarea6.1_2.2.sql)

*** No funciona del todo correcto**

Bloque anonimo de test del procedimiento

Es el mismo que del ejercicio anterior

Trigger

```
CREATE OR REPLACE TRIGGER UPDATEPRECIO before INSERT OR UPDATE
ON LINEAS FOR EACH ROW
DECLARE
    importeTotal pedidos.total%TYPE;
    numPedido pedidos.num%TYPE;
    pedido pedidos%rowtype;
BEGIN
    IF INSERTING THEN --Esto no es elegante
        numPedido := :new.num_pedido;
        select sum (lineas.importe) into importeTotal from lineas where num_pedido = numPedido;
        importeTotal := importetotal + :new.importe ;
    ELSIF UPDATING THEN --Esto no es elegante
        numPedido := :old.num_pedido;
        select sum (lineas.importe) into importeTotal from lineas where num_pedido = numPedido;
        importeTotal := importetotal + :new.importe ;
    --ELSIF DELETING THEN-- Esto falla
        --numPedido := :old.num_pedido;
        --select sum(lineas.importe) into importeTotal from lineas where num_pedido = numPedido;
        --importeTotal := importeTotal - :old.importe ;
    END IF;
    update pedidos set pedidos.total = importetotal where pedidos.num = numPedido;
    dbms_output.put_line( 'El nuevo total va a ser : ' || importeTotal);
END;
```

Trigger UPDATEPRECIO compilado