CESUR
Tu Centro Oficial de FP

CASO PRÁCTICO 1

PROGRAMACIÓN EN BASE DE DATOS ORACLE

Contexto

Como desarrollador, en tu empresa has realizado distintas bases de datos para poder optimizar y mejorar el trabajo en la compañía. Hace unos días, tus jefes te han pedido que elabores, para una de esas bases de datos, una serie de procedimientos almacenados y funciones, así como el código necesario para ejecutarlos.

Cuestiones a resolver

Son las siguientes:

1. Crea un procedimiento que muestre 'Hola mundo' por pantalla.

```
CREATE OR REPLACE PROCEDURE mostrar_pantalla IS
BEGIN
   DBMS_OUTPUT.PUT_LINE('Hola mundo');
END;
/

-- Ejecutar el procedimiento:

BEGIN
   mostrar_pantalla;
END;
/
```

2. Crea un procedimiento que pasándole una variable numérica cómo parámetro, muestre un mensaje diciendo si esa variable es mayor de 10 o no. Deberá mostrar un mensaje similar a este: "El número x es mayor que 10" o "El número x es menor que 10", siendo x el número pasado como parámetro.

```
CREATE OR REPLACE PROCEDURE mayor10(valor IN NUMBER) IS
BEGIN
   IF valor <= 10 THEN
      DBMS_OUTPUT.PUT_LINE('El número ' || valor || ' es menor o igual que 10');
   ELSE
      DBMS_OUTPUT.PUT_LINE('El número ' || valor || ' es mayor que 10');
   END IF;
END;
/

-- Declaración de la variable y solicitud de entrada de valor
DECLARE
   mi_numero NUMBER := &ingresa_valor; -- Se solicita el número
BEGIN
   DBMS_OUTPUT.PUT_LINE('Evaluamos si el número es mayor o menor que 10');
   mayor10(mi_numero);
END;
/
```

3. Crea un procedimiento que pasándole una variable numérica devuelva su valor multiplicado por 2 en un parámetro de salida y muestre el valor por consola

```
-- Crear o reemplazar el procedimiento
CREATE OR REPLACE PROCEDURE multiplicar_por_2 (valor IN NUMBER, resultado OUT NUMBER) IS
BEGIN
   resultado := valor * 2;
   DBMS_OUTPUT.PUT_LINE('El resultado de multiplicar ' || valor || ' por 2 es: ' || resultado);
END;
/

-- Ejecutar el procedimiento pidiendo el valor al usuario
DECLARE
   mi_numero NUMBER := &ingresa_valor;  -- Pide un valor al usuario
   mi_resultado NUMBER;  -- Variable para almacenar el resultado
BEGIN
   multiplicar_por_2(mi_numero, mi_resultado);  -- Llamada al procedimiento
END;
/
```

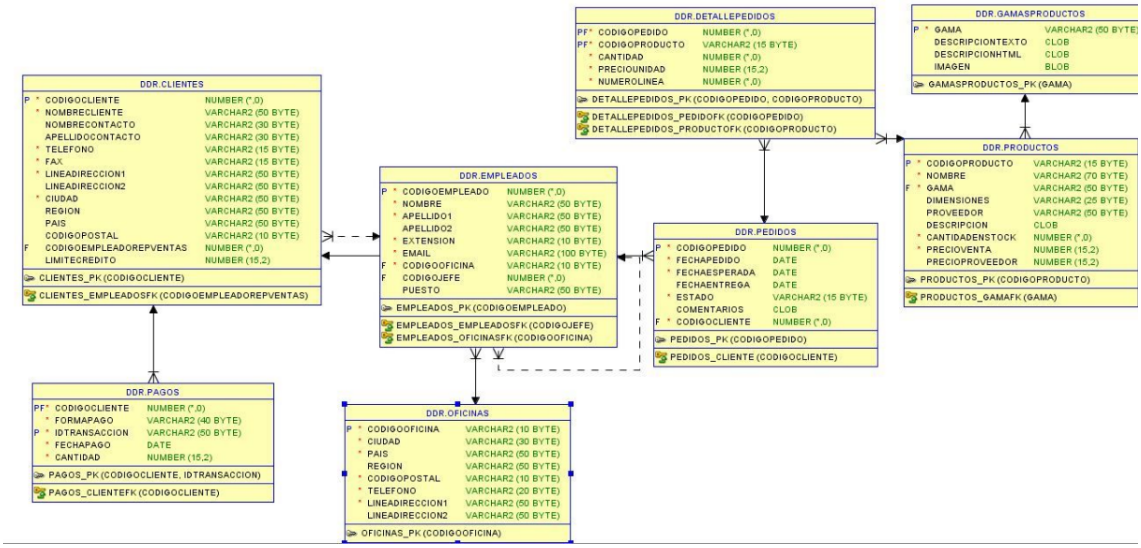
4. Crea un procedimiento que muestre los números del 1 al 100 con un WHILE.

```
-- Crear o reemplazar el procedimiento
CREATE OR REPLACE PROCEDURE mostrar_num_menor_100( valor IN NUMBER, resultado OUT NUMBER)
IS
   i NUMBER := valor; -- Iniciar 'i' con el valor proporcionado
BEGIN
   -- Comenzar el bucle mientras 'i' sea menor que 100
   WHILE (i < 100) LOOP
      resultado := i; -- Asignar el valor de 'i' al parámetro OUT 'resultado'
      DBMS_OUTPUT.PUT_LINE('Número: ' || resultado); -- Mostrar el número
      i := i + 1; -- Incrementar 'i'
   END LOOP;
END;
/

-- Ejecutar el procedimiento:
DECLARE
   resultado NUMBER; -- Variable para almacenar el resultado
BEGIN
   mostrar_num_menor_100(1, resultado); -- Iniciar el bucle desde el valor 1
END;
/
```

5. Crea un procedimiento para mostrar los números pares del 1 al 100 con un FOR

```
-- Crear o reemplazar el procedimiento
CREATE OR REPLACE PROCEDURE mostrar_numero_menor_100(
   valor IN NUMBER,
   resultado OUT NUMBER
)
IS
BEGIN
   -- Usar un bucle FOR con el rango correcto
   FOR i IN valor..99 LOOP
      resultado := i;  -- Asignar el valor de i al parámetro OUT 'resultado'
      DBMS_OUTPUT.PUT_LINE('Número: ' || resultado);  -- Mostrar el número
   END LOOP;
END;
/

-- Ejecutar el procedimiento:
DECLARE
   resultado NUMBER;  -- Variable para almacenar el resultado
BEGIN
   mostrar_numero_menor_100(1, resultado);  -- Iniciar el bucle desde el valor 1
END;
/
```

Dado el siguiente esquema de base de datos:



6. Crea un TRIGGER para actualizar la tabla T_PRODUCTOS. Cuando se haga un INSERT en la tabla T_DETALLEPEDIDOS, se debe actualizar el campo

CANTIDADENSTOCK de la tabla T_ PRODUCTOS. Por ejemplo, si hago el siguiente INSERT:

Insert into T_DETALLEPEDIDOS (CODIGOPEDIDO, CODIGOPRODUCTO, CANTIDAD, PRECIOUNIDAD, NUMEROLINEA) values (2,'FR-67',5,70,3);

al producto con código FR-67 se le debe restar 5 al campo CANTIDADENSTOCK de la tabla T_PRODUCTOS. Sin embargo, si lo que hago es un borrado de la tabla T_DETALLEPEDIDOS, lo que debe hacer el trigger es sumar esa cantidad. Por ejemplo: después de la inserción anterior, tenemos lo siguiente en la tabla T_DETALLEPEDIDOS para el producto FR-67.

La última línea es la que acabamos de insertar.

CODIGOPEDIDO	CODIGOPRODUCTO	CANTIDAD	PRECIOUNIDAD	NUMEROLINEA
1	FR-67	10	70	3
74	FR-67	15	70	1
2	FR-67	5	70	3

Si la borramos, lo que debe hacer el trigger es volver a sumar esa cantidad (5), en el campo CANTIDADENSTOCK de la tabla T_PRODUCTOS.

```
CREATE OR REPLACE TRIGGER mi_trigger_actualizador
AFTER INSERT OR DELETE ON T_DETALLEPEDIDOS
FOR EACH ROW
BEGIN
  -- Cuando se inserta un nuevo pedido, se resta la
cantidad del stock
  IF INSERTING THEN
    UPDATE T_PRODUCTOS
    SET CANTIDADENSTOCK = CANTIDADENSTOCK
- :NEW.CANTIDAD
    WHERE CODIGOPRODUCTO = :NEW.CODIGOPRODUCTO;

  -- Cuando se elimina un pedido, se vuelve a sumar la
cantidad al stock
  ELSIF DELETING THEN
    UPDATE T_PRODUCTOS
    SET CANTIDADENSTOCK = CANTIDADENSTOCK
+ :OLD.CANTIDAD
    WHERE CODIGOPRODUCTO = :OLD.CODIGOPRODUCTO;
  END IF;
END;
```

7. Crea un procedimiento que use un cursor para ver todos los productos comprados en un pedido (pasa el código de pedido como parámetro). Muestra también la cantidad de cada uno de los artículos.

```
CREATE OR REPLACE PROCEDURE ver_productos_en_pedido(p_codpedido IN
NUMBER) IS
 CURSOR productos_cursor IS
  SELECT dp.CODIGOPRODUCTO, dp.CANTIDAD
  FROM T_DETALLEPEDIDOS dp
  WHERE dp.CODIGOPEDIDO = p_codpedido;

 v_codproducto T_DETALLEPEDIDOS.CODIGOPRODUCTO%TYPE;
 v_cantidad T_DETALLEPEDIDOS.CANTIDAD%TYPE;
BEGIN
 OPEN productos_cursor;

 LOOP
  FETCH productos_cursor INTO v_codproducto, v_cantidad;
  EXIT WHEN productos_cursor%NOTFOUND;

      DBMS_OUTPUT.PUT_LINE('Producto: ' || v_codproducto || ' | Cantidad: ' ||
v_cantidad);
 END LOOP;

 CLOSE productos_cursor;
END ver_productos_en_pedido;

-- Esto siguiente lo ejecuto detras para comprobar el resultado
BEGIN
 ver_productos_en_pedido(10); -- Debajo ejecuto con el codigo 10 y veo el resultado
END;
```

8. Realiza una función que devuelva la suma de pagos que ha realizado un cliente, pasando su identificador de cliente como parámetro. Se debe mostrar un mensaje similar a este: “El total del cliente con código x es: y” siendo x el código pasado como parámetro e y el resultado de la operación.

```
CREATE OR REPLACE FUNCTION suma_pagos_cliente(p_cliente_id IN NUMBER)
RETURN NUMBER IS
 v_total NUMBER := 0;
BEGIN
 -- Cambié MONTO por CANTIDAD según la estructura de la tabla
 SELECT NVL(SUM(pago.CANTIDAD), 0)
 INTO v_total
 FROM T_PAGOS pago
 WHERE pago.CODIGOCLIENTE = p_cliente_id;

  DBMS_OUTPUT.PUT_LINE('El total del cliente con código ' || p_cliente_id || ' es: '
|| v_total);
 RETURN v_total;
END suma_pagos_cliente;

-- Y para comprobarlo hacemos esto de debajo, poniendo un codigo de cliente
DECLARE
 v_total NUMBER;
BEGIN
 v_total := suma_pagos_cliente(7);
 DBMS_OUTPUT.PUT_LINE('El total de pagos para el cliente es: ' || v_total);
END;
```

9. Crea un procedimiento almacenado para dar de alta un nuevo producto. Para ello, deberás pasar como parámetros los siguientes valores como mínimo: código, nombre, gama, cantidadstock y precioventa.

```
CREATE OR REPLACE PROCEDURE alta_producto(
  p_codproducto IN VARCHAR2,
  p_nombre IN VARCHAR2,
  p_gama IN VARCHAR2,
  p_cantidadstock IN NUMBER,
  p_precioventa IN NUMBER
) IS
  v_count NUMBER;
BEGIN
  -- Comprobar si el producto ya existe
  SELECT COUNT(*) INTO v_count
  FROM T_PRODUCTOS
  WHERE CODIGOPRODUCTO = p_codproducto;

  IF v_count > 0 THEN
     DBMS_OUTPUT.PUT_LINE('El producto con código ' || p_codproducto || ' ya está
dado de alta.');
  ELSE
    -- Insertar el producto si no existe
              INSERT INTO T_PRODUCTOS (CODIGOPRODUCTO, NOMBRE, GAMA,
CANTIDADENSTOCK, PRECIOVENTA)
    VALUES (p_codproducto, p_nombre, p_gama, p_cantidadstock, p_precioventa);
      DBMS_OUTPUT.PUT_LINE('Producto con código ' || p_codproducto || ' dado de
alta con éxito.');
  END IF;
END alta_producto;

-- Llamamos aqui para ver como lo da de alta con un ejemplo
BEGIN
  alta_producto('OR-103', 'Mimosa', 'Ornamentales', 100, 7);
END;
-- Y volvemos a llamarla para ver como reaciona
  alta_producto('OR-103', 'Mimosa', 'Ornamentales', 100, 7);
```

Antes de dar de alta el nuevo producto, el procedimiento debe asegurarse de que ese producto no existe en la tabla. Si existe, deberá mostrar un mensaje avisando de que ese producto ya está dado de alta. Por el contrario, si no existe, procederá con la inserción y mostrará un mensaje de éxito. Hazlo sin usar excepciones

10. Haz lo mismo que en el apartado anterior, pero sin usar un IF. Para este apartado debes hacer uso de las excepciones de Oracle. La excepción que tienes que usar es: DUP_VAL_ON_INDEX

```
CREATE OR REPLACE PROCEDURE alta_producto_con_excepciones(
  p_codproducto IN VARCHAR2,
  p_nombre IN VARCHAR2,
  p_gama IN VARCHAR2,
  p_cantidadstock IN NUMBER,
  p_precioventa IN NUMBER
) IS
BEGIN
  -- Intentar insertar el producto
      INSERT INTO T_PRODUCTOS (CODIGOPRODUCTO, NOMBRE, GAMA, CANTIDADENSTOCK, PRECIOVENTA)
  VALUES (p_codproducto, p_nombre, p_gama, p_cantidadstock, p_precioventa);

   DBMS_OUTPUT.PUT_LINE('Producto con código ' || p_codproducto || ' dado de alta con éxito.');

EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    DBMS_OUTPUT.PUT_LINE('El producto con código ' || p_codproducto || ' ya está dado de alta.');
END alta_producto_con_excepciones;

--Probamos el codigo aqui y como ya habiamos añadido antes este valor directamente nos mostrara la exception
BEGIN
  alta_producto('OR-103', 'Mimosa', 'Ornamentales', 100, 7);
END;
```