Bases de Datos Estructuradas

2024

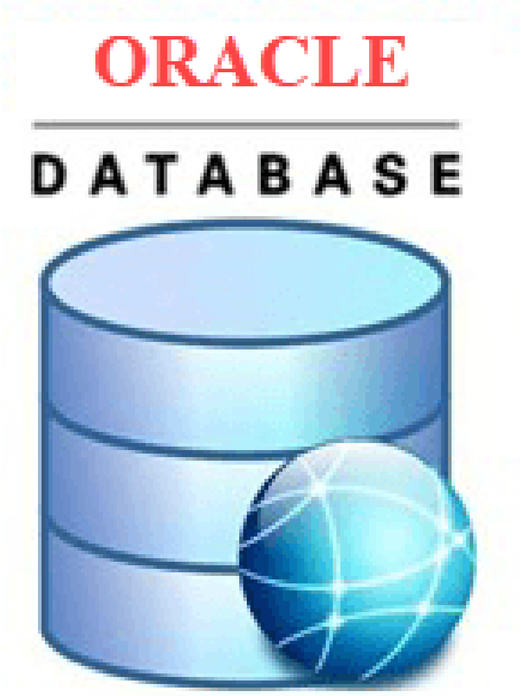
CONTENIDO

1. Unidad PL-SQL
2. Ejercicios

PL - SQL

PL/SQL es un lenguaje de procedimiento diseñado específicamente para abarcar sentencias SQL dentro de su sintaxis. El servidor de Oracle Database compila las unidades de programa PL/SQL y se almacenan dentro de la base de datos. Y en tiempo de ejecución, tanto PL/SQL como SQL se ejecutan dentro del mismo proceso de servidor, brindando una eficiencia óptima. PL/SQL hereda automáticamente la robustez, la seguridad y la portabilidad de Oracle Database.





- Procedural Language/Structured Query Language.
- Extensión de SQL para soportar programación procedural en Oracle.
- Permite crear procedimientos almacenados, funciones, triggers y más.

- `DECLARE` (optional)
  - Variables, cursors, user-defined exceptions
- `BEGIN` (mandatory)
  - SQL statements
  - PL/SQL statements
- `EXCEPTION` (optional)
  - Actions to perform when exceptions occur
- `END;` (mandatory)



Declaración de Variables:

Sintaxis:

```
DECLARE
    variable_name data_type [NOT NULL] := value;
```

Ejemplos:

```
DECLARE
    v_salario NUMBER := 5000;
    v_nombre VARCHAR2(50) := 'Juan';
```

Ejercicio 1: Cálculo de Bonificación

Descripción: Crea un procedimiento almacenado llamado sp_CalcularBonificacion que reciba el salario de un empleado y calcule su bonificación. La bonificación será del 10% si el salario es menor a 1000, y del 5% si es mayor o igual a 1000. El procedimiento debe imprimir el salario con bonificación.

Ejercicio 1: Cálculo de Bonificación

Descripción: Crea un procedimiento almacenado llamado sp_CalcularBonificacion que reciba el salario de un empleado y calcule su bonificación. La bonificación será del 10% si el salario es menor a 1000, y del 5% si es mayor o igual a 1000. El procedimiento debe imprimir el salario con bonificación.

```
CREATE OR REPLACE PROCEDURE sp_CalcularBonificacion (
    p_Salario NUMBER
) IS
    v_Bonificacion NUMBER;
    v_SalarioTotal NUMBER;
BEGIN
    IF p_Salario < 1000 THEN
        v_Bonificacion := p_Salario * 0.10;
    ELSE
        v_Bonificacion := p_Salario * 0.05;
    END IF;

    v_SalarioTotal := p_Salario + v_Bonificacion;

    DBMS_OUTPUT.PUT_LINE('El salario total con bonificación es: '
|| v_SalarioTotal);
END;
/
```

Ejercicio 2: Verificar si un Número es Par o Impar

Descripción: Crea un bloque PL/SQL que verifique si un número es par o impar e imprima el mensaje correspondiente.

Ejercicio 2: Verificar si un Número es Par o Impar

Descripción: Crea un bloque PL/SQL que verifique si un número es par o impar e imprima el mensaje correspondiente.

```
DECLARE
    v_numero NUMBER := 15;
BEGIN
    IF v_numero MOD 2 = 0 THEN
        DBMS_OUTPUT.PUT_LINE('El número ' || v_numero || ' es par.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('El número ' || v_numero || ' es impar.');
    END IF;
END;
/
```

Ejercicio 3: Contar Productos en Stock Bajo
Descripción: Crea un procedimiento almacenado llamado sp_ContarProductosBajoStock que cuente cuántos productos tienen un stock menor a 5. Imprime el total de productos con stock bajo.

Ejercicio 3: Contar Productos en Stock Bajo
Descripción: Crea un procedimiento almacenado llamado sp_ContarProductosBajoStock que cuente cuántos productos tienen un stock menor a 5. Imprime el total de productos con stock bajo.

```
CREATE OR REPLACE PROCEDURE sp_ContarProductosBajoStock IS
    v_ProductosBajoStock NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_ProductosBajoStock
    FROM Productos
    WHERE Stock < 5;

    DBMS_OUTPUT.PUT_LINE('Total de productos con stock bajo: ' ||
v_ProductosBajoStock);
END;
/
```

Ejercicio 4: Actualizar Precio de Productos
Descripción: Crea un procedimiento almacenado llamado sp_ActualizarPrecioProducto que reciba el ID del producto y un porcentaje de incremento, y actualice el precio del producto incrementándolo en ese porcentaje. Imprime el nuevo precio del producto.

Ejercicio 4: Actualizar Precio de Productos
Descripción: Crea un procedimiento almacenado llamado sp_ActualizarPrecioProducto que reciba el ID del producto y un porcentaje de incremento, y actualice el precio del producto incrementándolo en ese porcentaje. Imprime el nuevo precio del producto.

```
CREATE OR REPLACE PROCEDURE sp_ActualizarPrecioProducto (
    p_ProductoID NUMBER,
    p_PorcentajeAumento NUMBER
) IS
    v_PrecioActual NUMBER;
    v_NuevoPrecio NUMBER;
BEGIN
    -- Obtener el precio actual
    SELECT Precio INTO v_PrecioActual
    FROM Productos
    WHERE ProductoID = p_ProductoID;

    -- Calcular el nuevo precio
    v_NuevoPrecio := v_PrecioActual + (v_PrecioActual * p_PorcentajeAumento /
100);

    -- Actualizar el precio del producto
    UPDATE Productos
    SET Precio = v_NuevoPrecio
    WHERE ProductoID = p_ProductoID;

    DBMS_OUTPUT.PUT_LINE('El nuevo precio del producto con ID ' ||
p_ProductoID || ' es: ' || v_NuevoPrecio);
END;
/
```

Ejercicio 5: Calcular el Salario Promedio de Empleados

Descripción: Crea un procedimiento almacenado llamado sp_CalcularSalarioPromedio que calcule el salario promedio de los empleados de la tabla Empleados. Imprime el salario promedio.

Ejercicio 5: Calcular el Salario Promedio de Empleados
Descripción: Crea un procedimiento almacenado llamado sp_CalcularSalarioPromedio que calcule el salario promedio de los empleados de la tabla Empleados. Imprime el salario promedio.

```
CREATE OR REPLACE PROCEDURE sp_CalcularSalarioPromedio IS
   v_SalarioPromedio NUMBER;
BEGIN
   SELECT AVG(Salario)
   INTO v_SalarioPromedio
   FROM Empleados;

   DBMS_OUTPUT.PUT_LINE('El salario promedio es: ' || v_SalarioPromedio);
END;
/
```

Ejercicio 6: Insertar Nueva Venta

Descripción: Crea un procedimiento almacenado llamado sp_InsertarVenta que reciba el ID del cliente, el ID del producto, y la cantidad vendida. Inserta la nueva venta en la tabla Ventas y los detalles en la tabla DetallesVentas.

Ejercicio 6: Insertar Nueva Venta
Descripción: Crea un procedimiento almacenado llamado sp_InsertarVenta que reciba el ID del cliente, el ID del producto, y la cantidad vendida. Inserta la nueva venta en la tabla Ventas y los detalles en la tabla DetallesVentas.

```
CREATE OR REPLACE PROCEDURE sp_InsertarVenta (
    p_ClienteID NUMBER, p_ProductoID NUMBER, p_Cantidad NUMBER
) IS
    v_TotalLinea NUMBER;
    v_PrecioProducto NUMBER;
    v_VentaID NUMBER;
BEGIN
    -- Obtener el precio del producto
    SELECT Precio INTO v_PrecioProducto
    FROM Productos
    WHERE ProductoID = p_ProductoID;

    -- Calcular el total de la línea
    v_TotalLinea := v_PrecioProducto * p_Cantidad;

    -- Insertar la venta
    INSERT INTO Ventas (ClienteID, FechaVenta, TotalVenta)
    VALUES (p_ClienteID, SYSDATE, v_TotalLinea)
    RETURNING VentaID INTO v_VentaID;

    -- Insertar los detalles de la venta
    INSERT INTO DetallesVentas (VentaID, ProductoID, Cantidad, TotalLinea)
    VALUES (v_VentaID, p_ProductoID, p_Cantidad, v_TotalLinea);

    DBMS_OUTPUT.PUT_LINE('Venta insertada correctamente con ID: ' || v_VentaID);
END;
/
```

Ejercicio 7: Calcular el Total de Ventas de un ClienteDescripción: Crea un procedimiento almacenado llamado sp_CalcularTotalVentasCliente que reciba el ID de un cliente y calcule el total de ventas realizadas por ese cliente. Imprime el total de ventas.

Ejercicio 7: Calcular el Total de Ventas de un ClienteDescripción: Crea un procedimiento almacenado llamado sp_CalcularTotalVentasCliente que reciba el ID de un cliente y calcule el total de ventas realizadas por ese cliente. Imprime el total de ventas.

```
CREATE OR REPLACE PROCEDURE sp_CalcularTotalVentasCliente (
    p_ClienteID NUMBER
) IS
    v_TotalVentasCliente NUMBER;
BEGIN
    SELECT SUM(TotalVenta)
    INTO v_TotalVentasCliente
    FROM Ventas
    WHERE ClienteID = p_ClienteID;

    DBMS_OUTPUT.PUT_LINE('El total de ventas del cliente con ID ' ||
p_ClienteID || ' es: ' || v_TotalVentasCliente);
END;
/
```

Ejercicio 8: Bucle para Mostrar Empleados con Salario Mayor a un Valor

Descripción: Crea un bloque PL/SQL que use un bucle FOR para mostrar todos los empleados cuyo salario es mayor a 1500.

Ejercicio 8: Bucle para Mostrar Empleados con Salario Mayor a un Valor
Descripción: Crea un bloque PL/SQL que use un bucle FOR para mostrar todos los empleados cuyo salario es mayor a 1500.

```
DECLARE
    v_Nombre VARCHAR2(50);
    v_Salario NUMBER;
BEGIN
    FOR r IN (SELECT Nombre, Salario FROM Empleados WHERE Salario > 1500) LOOP
        v_Nombre := r.Nombre;
        v_Salario := r.Salario;
        DBMS_OUTPUT.PUT_LINE('Empleado: ' || v_Nombre || ', Salario: ' || v_Salario);
    END LOOP;
END;
/
```

Ejercicio 9: Calcular el Descuento de un ProductoDescripción: Crea una función llamada fn_CalcularDescuento que reciba el ID de un producto y un porcentaje de descuento, y devuelva el precio final con el descuento aplicado.

Ejercicio 9: Calcular el Descuento de un ProductoDescripción: Crea una función llamada fn_CalcularDescuento que reciba el ID de un producto y un porcentaje de descuento, y devuelva el precio final con el descuento aplicado.

```
CREATE OR REPLACE FUNCTION fn_CalcularDescuento (
    p_ProductoID NUMBER,
    p_PorcentajeDescuento NUMBER
) RETURN NUMBER IS
    v_PrecioProducto NUMBER;
    v_PrecioConDescuento NUMBER;
BEGIN
    -- Obtener el precio del producto
    SELECT Precio INTO v_PrecioProducto
    FROM Productos
    WHERE ProductoID = p_ProductoID;

    -- Calcular el precio con descuento
    v_PrecioConDescuento := v_PrecioProducto -
(v_PrecioProducto * p_PorcentajeDescuento / 100);

    RETURN v_PrecioConDescuento;
END;
/
```

MUCHAS GRACIAS!

inacap.cl

UNIVERSIDAD TECNOLÓGICA DE CHILE
INSTITUTO PROFESIONAL
CENTRO DE FORMACIÓN TÉCNICA

inacap.cl