ADMINISTRACIÓN DE BASE DE DATOS.

Tema Nº8:Programación PL/SQL IV.

Indicador de logro Nº8:Construye de programas, a través de funciones .

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº8:**

Programación PL/SQL IV.

**Subtema 8.1:**

¿Qué es una Función de usuario?

Una función un programa PL/SQL, cuyo código se guarda en la base de datos y tiene como objetivo realizar un cálculo y devolver un valor.

**SINTAXIS:**

**CREATE OR REPLACE FUNCTION** <NOMBRE\_FUNCTION> (PARAMETROS)

**RETURN** <TIPO\_DATO>

**IS**

<DEFINICION\_DE\_VARIABLES>

**BEGIN**

<CODIGO\_PLSQL>

**END;**

**Subtema 8.2:**

Implementa funciones, utilizando el lenguaje PL/SQL, empleo de la sentencia **Create or Replace Function** y uso de parámetros **in**, **out**, **in out**.

**Por favor descargar los Archivos:**

**Semana08\_PL\_SQL\_1.sql, Semana08\_PL\_SQL\_2.sql**

**Ejemplo 1:**

Crear una función el cual reciba 3 valores A, B y OP las variables A y B recibirán valores, los cuales serán sometidos a las operaciones básicas conocidas, la variable OP decidirá el tipo de operación a realizar, estas pueden ser: 1=Suma, 2=Resta, 3=Multiplicación y 4=División. La función deberá devolver un resultado (RETURN).

**CREATE OR REPLACE FUNCTION** OPERACION

(A NUMBER, B NUMBER, OP NUMBER)

**RETURN** NUMBER

**IS**

**C** NUMBER:=0;

**BEGIN**

**CASE**

**WHEN** OP=1 THEN **C**:=A + B;

**WHEN** OP=2 THEN **C**:=A - B;

**WHEN** OP=3 THEN **C**:=A \* B;

**WHEN** OP=4 THEN **C**:=A / B;

**ELSE**

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

**END CASE;**

**RETURN** **C**;

**END;**

**La Función debe ser probada de 2 formas:**

**FORMA 1: usando la tabla DUAL.**

SELECT **OPERACION** (10, 20, 1) "SUMA" FROM DUAL;

SELECT **OPERACION** (10, 20, 2) "RESTA" FROM DUAL;

SELECT **OPERACION** (10, 20, 3) "MULTIPLICACION" FROM DUAL;

SELECT **OPERACION** (10, 20, 4) "DIVISION" FROM DUAL;

**FORMA 2: desde un bloque anónimo**

**SET SERVEROUTPUT ON;**

**DECLARE**

Valor NUMBER;

**BEGIN**

Valor := **OPERACION** (10, 10, 1);

DBMS\_OUTPUT.PUT\_LINE ('LA SUMA ES: ' || Valor);

Valor := **OPERACION** (10, 10, 2);

DBMS\_OUTPUT.PUT\_LINE ('LA RESTA ES: ' || Valor);

Valor := **OPERACION** (10, 10, 3);

DBMS\_OUTPUT.PUT\_LINE ('LA MULTIPLICACION ES: ' || Valor);

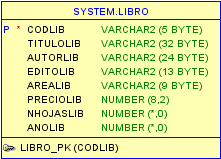
Valor := **OPERACION** (10, 10, 4);

DBMS\_OUTPUT.PUT\_LINE('LA DIVISION ES: ' || Valor);

**END;**

**EJEMPLO 2:**

Se tiene la siguiente tabla:



**CREATE TABLE LIBRO** (

CODLIB VARCHAR2 (5) PRIMARY KEY NOT NULL,

TITULOLIB VARCHAR2 (32),

AUTORLIB VARCHAR2 (24),

EDITOLIB VARCHAR2 (13),

AREALIB VARCHAR2 (9),

PRECIOLIB NUMERIC (8, 2),

NHOJASLIB INT,

ANOLIB INT);

Se pide crear una función que permita contabilizar los libros según el área del libro.

**SOLUCION:**

**CREATE OR REPLACE FUNCTION** Contar\_Lib

(**Area** Varchar2)

**RETURN** INT

**IS**

CONTEO INT:=0;

**BEGIN**

SELECT COUNT (AREALIB) INTO CONTEO FROM LIBRO

WHERE AREALIB = **Area**;

**RETURN** CONTEO;

**END;**

**La función debe ser probada de 2 formas:**

**FORMA 1: usando la tabla DUAL.**

SELECT **Contar\_Lib** ('LPROG') "Conteo de Libros" FROM DUAL;

**FORMA 2: desde un bloque anónimo.**

**SET SERVEROUTPUT ON;**

**DECLARE**

Ar Varchar2(20):='LPROG';

**BEGIN**

DBMS\_OUTPUT.PUT\_LINE('Area: ' || Ar || ' Son: ' || Contar\_Lib(Ar));

**END;**

**EJEMPLO 3:**

Se pide crear una función que permita obtener la totalización los libros según el área del libro.

**SOLUCION:**

**CREATE OR REPLACE FUNCTION** Total\_Lib

(**Area** Varchar2)

**RETURN** INT

**IS**

TOTAL INT:=0;

**BEGIN**

SELECT SUM (AREALIB) INTO TOTAL FROM LIBRO WHERE AREALIB = **Area**;

**RETURN** TOTAL;

**END;**

**La Función debe ser probada de 2 formas:**

**FORMA 1: usando la tabla DUAL.**

SELECT **Total\_Lib** ('LPROG') "Total de Libros" FROM DUAL;

**FORMA 2: desde un bloque anónimo.**

**SET SERVEROUTPUT ON;**

**DECLARE**

Ar Varchar2 (20):='LPROG';

**BEGIN**

DBMS\_OUTPUT.PUT\_LINE ('Area: ' || Ar || ' Son: ' ||

Total\_Lib(Ar));

**END;**

**EJEMPLO 4:**

Crear las funciones necesarias, el cual devuelva Titulo y el precio de un determinado libro, para ello deberá usar como campo de búsqueda al código de libro, a continuación, calcular el importe según cantidad de pedido, y se pide calcular adicional mente el IGV y el Total a Pagar por dicho libro.

**SOLUCION:**

**CREATE OR REPLACE FUNCTION** BuscarLibro

(Codigo LIBRO.CODLIB%TYPE)

**RETURN** VARCHAR2

**IS**

Nombre LIBRO.TITULOLIB%TYPE:='';

**BEGIN**

SELECT TITULOLIB INTO Nombre FROM LIBRO WHERE CODLIB = Codigo;

**RETURN** Nombre;

**END;**

**CREATE OR REPLACE FUNCTION** BuscarPrecio

(Codigo LIBRO.CODLIB%TYPE)

**RETURN** NUMBER

**IS**

PRECIO LIBRO.PRECIOLIB%TYPE:=0;

**BEGIN**

SELECT PRECIOLIB INTO PRECIO FROM LIBRO WHERE CODLIB = Codigo;

**RETURN** PRECIO;

**END;**

**La Función debe ser probada de 2 formas:**

**FORMA 1: usando la tabla DUAL.**

SELECT **BuscarPrecio** (1000) "PRECIO" FROM DUAL;

SELECT **BuscarLibro** (1000) "LIBRO" FROM DUAL;

SELECT **BuscarPrecio** (1000) \* 4 "IMPORTE" FROM DUAL;

SELECT **BuscarPrecio** (1000) \* 4 \* 0.18 "IGV" FROM DUAL;

SELECT **BuscarPrecio** (1000) + (BuscarPrecio (1000) \* 4 \* 0.18)

"TOTAL PAGAR" FROM DUAL;

**FORMA 2: desde un bloque anónimo:**

**SET SERVEROUTPUT ON;**

**DECLARE**

CANTIDAD NUMBER:= **&**CANTIDAD;

C LIBRO.CODLIB**%TYPE**:=**1000**;

IMPORTE NUMBER(10, 2);

PRECIO LIBRO.PRECIOLIB**%TYPE**;

IGV NUMBER (10, 2):=0.0;

TOTAL\_PAGAR NUMBER(10, 2):=0.0;

**BEGIN**

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

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

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

DBMS\_OUTPUT.PUT\_LINE ('Libro : ' || BuscarLibro(C));

DBMS\_OUTPUT.PUT\_LINE ('Precio : ' || BuscarPrecio(C));

DBMS\_OUTPUT.PUT\_LINE ('Cantidad : ' || CANTIDAD);

PRECIO:=BuscarPrecio(C);

IMPORTE:= CANTIDAD \* PRECIO;

IGV:=IMPORTE\*0.18;

TOTAL\_PAGAR:=PRECIO + IGV;

DBMS\_OUTPUT.PUT\_LINE ('Importe : ' || IMPORTE);

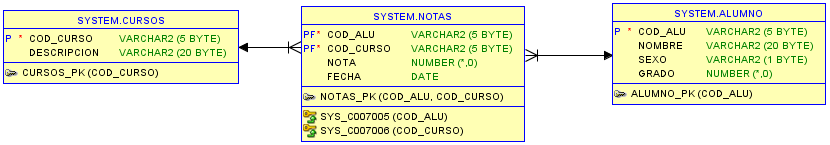
DBMS\_OUTPUT.PUT\_LINE ('IGV : ' || IGV);

DBMS\_OUTPUT.PUT\_LINE ('Total a Pagar : ' || TOTAL\_PAGAR);

**END;**

**EJEMPLO 5:**

Se tiene las siguientes tablas:



Crear una función el cual sea capaz de contar alumnos y alumnas por separado.

**Solución:**

**CREATE OR REPLACE FUNCTION** Conteo\_Generos

(Gen Varchar2)

**RETURN** INT

**IS**

V\_CONTEO INT:=0;

**BEGIN**

SELECT COUNT(\*) INTO V\_CONTEO FROM ALUMNO WHERE SEXO=Gen;

RETURN V\_CONTEO;

**END;**

**La función debe ser probada de 2 formas:**

**FORMA 01: usando la tabla DUAL.**

Conteo de Varones, usando la tabla DUAL:

SELECT Conteo\_Generos ('M') “VARONES” From Dual;

SELECT Conteo\_Generos ('F') “DAMAS” From Dual;

**FORMA 2: usando bloque anónimo.**

**SET SERVEROUTPUT ON;**

**DECLARE**

V\_CONTEO INT;

F\_CONTEO INT;

**BEGIN**

V\_CONTEO:=Conteo\_Generos ('M');

F\_CONTEO:=Conteo\_Generos ('F');

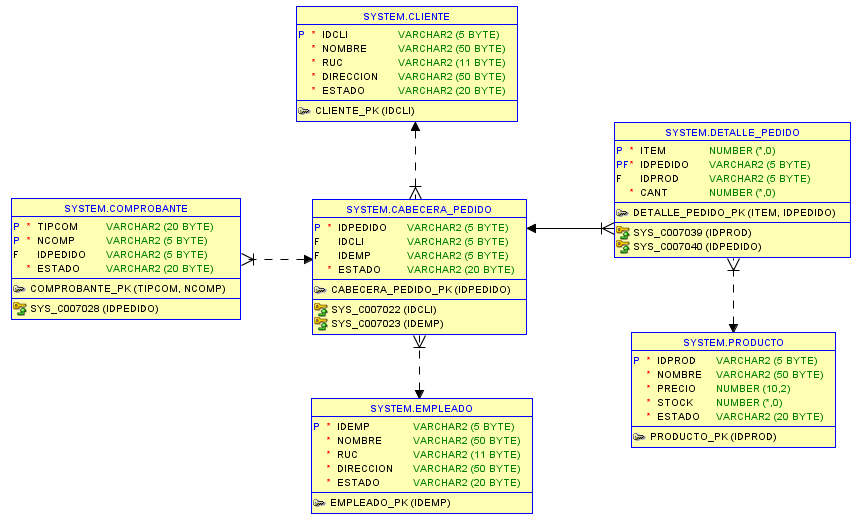
DBMS\_OUTPUT.PUT\_LINE ('CONTEO DE DAMAS : ' || F\_CONTEO);

DBMS\_OUTPUT.PUT\_LINE ('CONTEO DE VARONES : ' || V\_CONTEO);

**END;**

**EJEMPLO 6:**

Se tiene el siguiente modelo relacional:



Obtener el total de un determinado pedido, entre ellos tenemos: ‘00001’, ‘00002’ y ‘00003’, cada pedido tiene 3 productos.

Pedido ‘**00001’** tiene 3 productos y hace una totalización de importes de: **16499,91**

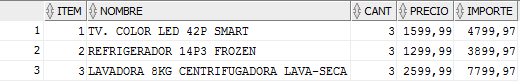
**SELECT D.ITEM, P.NOMBRE, D.CANT,**

**P.PRECIO, (D.CANT\* P.PRECIO) IMPORTE**

**FROM DETALLE\_PEDIDO D, PRODUCTO P**

**WHERE D.IDPROD = P.IDPROD AND**

**D.IDPEDIDO = '00001';**



Pedido ‘**00002’** tiene 3 productos y hace una totalización de importes de: **16199,94**

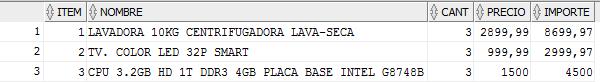
**SELECT D.ITEM, P.NOMBRE, D.CANT,**

**P.PRECIO, (D.CANT\* P.PRECIO) IMPORTE**

**FROM DETALLE\_PEDIDO D, PRODUCTO P**

**WHERE D.IDPROD = P.IDPROD AND**

**D.IDPEDIDO = '00002';**



Pedido ‘**00003’** tiene 3 productos y hace una totalización de importes de: **18399,91**

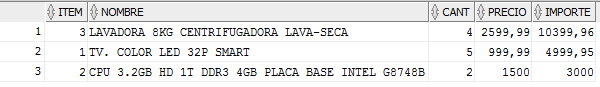
**SELECT D.ITEM, P.NOMBRE, D.CANT,**

**P.PRECIO, (D.CANT\* P.PRECIO) IMPORTE**

**FROM DETALLE\_PEDIDO D, PRODUCTO P**

**WHERE D.IDPROD = P.IDPROD AND**

**D.IDPEDIDO = '00003';**



**SOLUCION:**

**CREATE OR REPLACE FUNCTION** TOTALIZAR\_PEDIDOS

(CodigoP Varchar2)

**RETURN** NUMBER

**IS**

TOTAL NUMBER:=0;

**BEGIN**

SELECT SUM (D.CANT\* P.PRECIO) INTO TOTAL

FROM DETALLE\_PEDIDO D, PRODUCTO P

WHERE D.IDPROD = P.IDPROD AND

D.IDPEDIDO = CodigoP;

RETURN TOTAL;

**END;**

**La Función debe ser probada de 2 formas:**

**FORMA 01: usando la tabla DUAL.**

SELECT **TOTALIZAR\_PEDIDOS** ('00001') "TOTAL PEDIDO" FROM DUAL;

SELECT **TOTALIZAR\_PEDIDOS** ('00002') "TOTAL PEDIDO" FROM DUAL;

SELECT **TOTALIZAR\_PEDIDOS** ('00003') "TOTAL PEDIDO" FROM DUAL;

**FORMA 2: usando bloque anónimo.**

**SET SERVEROUTPUT ON;**

**DECLARE**

CodPed Varchar2(5);

TotalPed Number:=0;

**BEGIN**

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

DBMS\_OUTPUT.PUT\_LINE('TOTALIZACION DE PEDIDOS REALIZADOS');

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

CodPed:='00001';

TotalPed:=TOTALIZAR\_PEDIDOS(CodPed);

DBMS\_OUTPUT.PUT\_LINE('Pedido: ' || CodPed || ', Total: ' || TotalPed);

CodPed:='00002';

TotalPed:=TOTALIZAR\_PEDIDOS(CodPed);

DBMS\_OUTPUT.PUT\_LINE('Pedido: ' || CodPed || ', Total: ' || TotalPed);

CodPed:='00003';

TotalPed:=TOTALIZAR\_PEDIDOS(CodPed);

DBMS\_OUTPUT.PUT\_LINE('Pedido: ' || CodPed || ', Total: ' || TotalPed);

**END;**

**EJEMPLO 7:**

Crear una función el cual permita multiplicar una cadena una cierta cantidad de veces.

**SOLUCION:**

**CREATE OR REPLACE FUNCTION** REPLICATE (Cadena VARCHAR2, N NUMBER)

**RETURN** VARCHAR2

**IS**

REPETIDOR VARCHAR2 (4000);

**BEGIN**

REPETIDOR := '';

FOR I IN 1..N LOOP

REPETIDOR := REPETIDOR || Cadena;

END LOOP;

RETURN REPETIDOR;

**END;**

**La Función debe ser probada de 2 formas:**

**FORMA 01: usando la tabla DUAL, repetirá el digito cero 5 veces.**

SELECT REPLICATE ('0', 5) FROM DUAL;

**FORMA 2: usando bloque anónimo.**

**SET SERVEROUTPUT ON;**

**DECLARE**

TEXTO Varchar2 (5):='0';

**BEGIN**

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

DBMS\_OUTPUT.PUT\_LINE ('REPETIDOR DE TEXTOS');

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

DBMS\_OUTPUT.PUT\_LINE (REPLICATE (TEXTO, 10));

**END;**

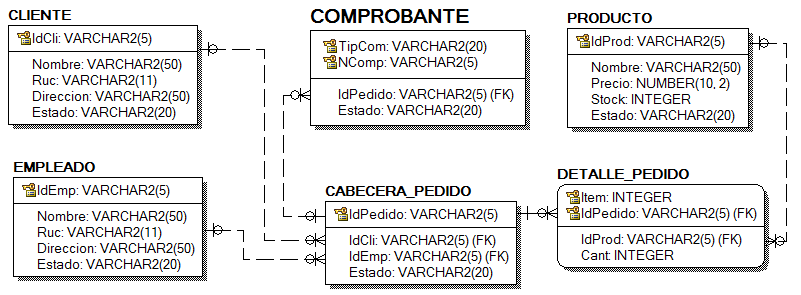
**CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA**

La codificación PL/SQL de Oracle es tan amplia, que ofrece a los estudiantes una alta gama de alternativas para la resolución de problemas en procedimientos y funciones, no difíciles de entender, con la prioridad de manejar los datos con suma responsabilidad por el programador Oracle.

**ACTIVIDAD VIRTUAL**

Revisar y analizar el material presentado en la carpeta del Tema 08.

**SE TIENE EL SIGUIENTE MODELO RELACIONAL:**



**LA ESTRUCTURA Y DATOS DE LAS TABLAS DEL MODELO RELACIONAL SE ENCUENTRAN**

**EN EL ARCHIVO SCRIPT DE BASE DE DATOS.**

**Semana08\_PL\_SQL\_3.sql,**

**FAVOR DE DESCARGAR DICHO ARCHIVO.**

De acuerdo al material presentado Semana 11, Responda a las siguientes Preguntas:

**RESOLVER:**

Crear las siguientes funciones en PL/SQL y probarlas en sus 2 formas:

1. Obtener el producto más caro con nombre de producto incluido.
2. Obtener el total de comprobantes, pruebe con los comprobantes ‘F0001’, ‘B0001’ y ‘B0002’, adicional mente obtener IGV (solo facturas) y TOTAL A PAGAR.
3. Obtener el producto más caro, de un determinado comprobante.