ADMINISTRACIÓN DE BASE DE DATOS.

Tema Nº9:Programación PL/SQL V.

Indicador de logro Nº9:Construye programas, a través de procedimientos Almacenados.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº9:**

Programación PL/SQL V.

**Subtema 9.1:**

¿Qué es un Procedimiento Almacenado?

Un stored procedure es un procedimiento cuyo código se guarda en la base de datos y tiene como objeto realizar una acción específica.

**Su sintaxis es la siguiente:**

**CREATE OR REPLACE PROCEDURE** <NOMBRE\_PROCEDURE> (**PARAMETROS**)

**IS**

<DEFINICION\_VARIABLES\_LOCALES\_DEL\_PROCEDIMIENTO,>

**BEGIN**

<CODIGO\_PLSQL>

. . . . . . . . . . . . . .

**END;**

* El uso de **OR REPLACE** permite sobre escribir un procedimiento existente. Si se omite, y el procedimiento existe, se producirá, un error.
* La sintaxis es muy parecida a la de un bloque anónimo, salvo porque se reemplaza la sección **DECLARE** por la secuencia **PROCEDURE... IS** en la especificación del procedimiento.
* Debemos especificar el tipo de datos de cada parámetro. Al especificar el tipo de dato del parámetro no debemos especificar la longitud del tipo.

**LOS PARÁMETROS QUE PUEDE RECIBIR UN STORED PROCEDURE PUEDEN SER DE 3 TIPOS:**

Los parámetros son variables que reciben valores desde el exterior y después de ciertos procesos son capaces de hacer retorno de valores.

**IN (Default):**

Si un parámetro es IN (**Entrada**) no puede ser modificado en el transcurso del código PLSQL.

**OUT:**

Si un parámetro es OUT (**Salida**) siempre llega al código del stored procedure con el valor de NULL y cuando termina el código de PLSQL el parámetro mantiene el valor de forma persistente.

**IN OUT: (Entrada/Salida)**

Es una combinación de ambos.

**Subtema 9.2:**

Implementa procedimientos almacenados, utilizando el lenguaje PL/SQL. Empleo de la sentencia **Create or Replace Procedure**, y uso de parámetros **in**, **out**, **in out**.

**PROCEDIMIENTOS ALMACENADOS SIN PARAMETROS.**

Es aquel procedimiento que no usa variables, después del nombre del procedimiento.

**Ejemplo 1:**

Ejemplo Básico de Procedimiento Almacenado sin usar parámetros: realiza las operaciones básicas entre dos valores y mostrar los resultados.

**CREATE OR REPLACE PROCEDURE** SP\_OPERACIONES\_BASICAS

**IS**

X NUMBER:=100; Y NUMBER:=50;

**BEGIN**

DBMS\_OUTPUT.PUT\_LINE ('SUMA :' || TO\_CHAR(X+Y));

DBMS\_OUTPUT.PUT\_LINE ('RESTA :' || TO\_CHAR(X-Y));

DBMS\_OUTPUT.PUT\_LINE (‘MULTIPLICACION :' || TO\_CHAR(X\*Y));

DBMS\_OUTPUT.PUT\_LINE ('DIVISION :' || TO\_CHAR(X/Y));

**END;**

Este procedimiento almacenado lleva como nombre **SP\_OPERACIONES\_BASICAS**, sin parámetros, usa dos variables locales (**después de IS**) X de tipo numéricos e Y variable de tipo numérica con los valores 100 y 50 correspondiente mente.

Desde un bloque anónimo, prueba el Procedimiento Almacenado SP\_OPERACIONES\_BASICAS

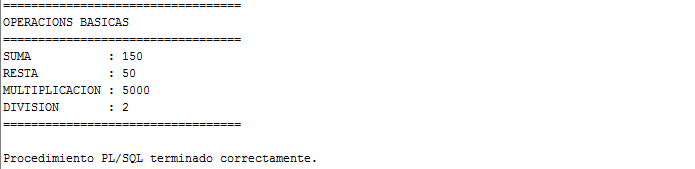
**SET SERVEROUTPUT ON;**

B**EGIN**

SP\_OPERACIONES BASICAS;

**END;**

**Resultado:**



**PROCEDIMIENTOS ALMACENADOS CON PARAMETROS (IN).**

Es aquel procedimiento que usa variables, después del nombre del procedimiento y entre paréntesis, en modalidad de Entrada.

**Ejemplo 2:**

Ejemplo Básico de Procedimiento Almacenado usando parámetros de entrada de valores (**IN**): realiza las operaciones básicas entre dos valores de entrada al procedimiento y mostrar los resultados.

**CREATE OR REPLACE PROCEDURE** SP\_OPERACIONES\_BASICAS2

**(X IN NUMBER, Y IN NUMBER)**

**IS**

**BEGIN**

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

DBMS\_OUTPUT.PUT\_LINE ('OPERACIONS BASICAS');

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

DBMS\_OUTPUT.PUT\_LINE ('SUMA : ' || TO\_CHAR(X+Y));

DBMS\_OUTPUT.PUT\_LINE ('RESTA : ' || TO\_CHAR(X-Y));

DBMS\_OUTPUT.PUT\_LINE ('MULTIPLICACION : ' || TO\_CHAR(X\*Y));

DBMS\_OUTPUT.PUT\_LINE (‘DIVISION : ' || TO\_CHAR(X/Y));

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

**END;**

Este procedimiento almacenado lleva como nombre SP\_OPERACIONES\_BASICAS2, usando dos parámetros (**X IN NUMBER, Y IN NUMBER**), y como vera, no hay variables locales (**después de IS**).

Desde un bloque anónimo, pruebe el Procedimiento Almacenado SP\_OPERACIONES\_BASICAS2, enviándole como argumentos los valores 100 y 50 en una línea de declaración en las variables locales **A** y **B**.

**SET SERVEROUTPUT ON;**

**DECLARE**

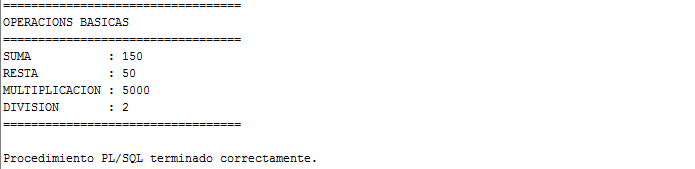
**A** NUMBER:= 100; **B** NUMBER:= 50;

**BEGIN**

SP\_OPERACIONES\_BASICAS2 (**A**, **B**);

**END;**

**Resultado:**



**PROCEDIMIENTOS ALMACENADOS CON PARAMETROS (IN - OUT).**

Es aquel procedimiento que usa variables, después del nombre del procedimiento y entre paréntesis, en modalidad de Entrada y Salida.

**Ejemplo 3:**

Ejemplo Básico de Procedimiento Almacenado usando parámetros de entrada y salida de valores separados (**IN - OUT**): donde se realizan las operaciones básicas entre dos valores de entrada al procedimiento y un valor de salida de datos como respuesta a los procesos, mostrando los resultados.

**PROCEDA A CREAR PROCEDIMIENTOS ALMACENADOS PARA CADA OPERACIÓN:**

**CREATE OR REPLACE PROCEDURE** **SP\_SUMA**

(X IN NUMBER, Y IN NUMBER, Z OUT NUMBER)

IS

**BEGIN**

Z := X + Y;

**END;**

**CREATE OR REPLACE PROCEDURE** **SP\_RESTA**

(X IN NUMBER, Y IN NUMBER, Z OUT NUMBER)

IS

**BEGIN**

Z := X - Y;

**END;**

**CREATE OR REPLACE PROCEDURE** **SP\_MULTIPLICACION**

(X IN NUMBER, Y IN NUMBER, Z OUT NUMBER)

IS

**BEGIN**

Z := X \* Y;

**END;**

**CREATE OR REPLACE PROCEDURE** **SP\_DIVISION**

(X IN NUMBER, Y IN NUMBER, Z OUT NUMBER)

IS

**BEGIN**

Z := X / Y;

**END;**

**CREATE OR REPLACE PROCEDURE** SP\_PRUEBA

(**X** IN NUMBER, **Y** IN NUMBER)

**IS**

**Z** NUMBER:=0;

**BEGIN**

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

DBMS\_OUTPUT.PUT\_LINE ('OPERACIONS BASICAS');

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

**SP\_SUMA**(X, Y, **Z**);

DBMS\_OUTPUT.PUT\_LINE ('SUMA : ' || TO\_CHAR(**Z**));

**SP\_RESTA**(X, Y, **Z**);

DBMS\_OUTPUT.PUT\_LINE ('RESTA : ' || TO\_CHAR(**Z**));

**SP\_MULTIPLICACION**(X, Y, **Z**);

DBMS\_OUTPUT.PUT\_LINE ('MULTIPLICACION : ' || TO\_CHAR(**Z**));

**SP\_DIVISION**(X, Y, **Z**);

DBMS\_OUTPUT.PUT\_LINE ('DIVISION : ' || TO\_CHAR(**Z**));

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

**END;**

Los procedimientos almacenados **SP\_SUMA**, **SP\_RESTA**, **SP\_MULTIPLICACION** y **SP\_DIVISION**, usan parámetros de entrada y salida **IN - OUT**, tal como se muestra en cada proceso.

El procedimiento almacenado **SP\_PRUEBA**, es aquel que hace la llamada de los demás procedimientos almacenados, este usa parámetros de solo entrada **IN** A y B de tipo **Number**. Y la declaración de una variable local **C** de tipo **Number** e inicializado a cero, el cual será el responsable de capturar los valores de cada operación.

**Desde un bloque anónimo**, probaremos el procedimiento almacenado **SP\_PRUEBA** que usa 2 parámetros P y Q, enviándole como argumentos, los valores 100 y 50 en una línea de declaración en las variables locales A y B.

**SET SERVEROUTPUT ON;**

**DECLARE**

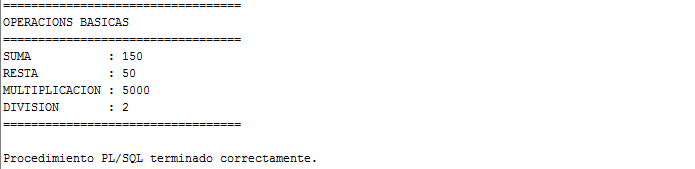
**P** NUMBER:=100; **Q** NUMBER:=50;

**BEGIN**

SP\_PRUEBA (**P**, **Q**);

**END;**

**Resultado:**



**PROCEDIMIENTOS ALMACENADOS CON PARAMETROS (IN OUT).**

Es aquel procedimiento que usa variables, después del nombre del procedimiento y entre paréntesis de modalidad Entrada - Salida.

**Ejemplo 4:**

Ejemplo Básico de Procedimiento Almacenado usando parámetros de **entrada/salida** de modalidad (**IN OUT**): donde se realiza la operación del área de un triángulo, con todos los parámetros de **entrada / salida** el procedimiento pueden recepcionar y remitir valores, mostrando los resultados.

**PROCEDA A CREAR PROCEDIMIENTOS ALMACENADO SP\_AREA\_TRIANGULO:**

**CREATE OR REPLACE PROCEDURE** SP\_AREA\_TRIANGULO

(**BASE** IN OUT NUMBER, **ALTURA** IN OUT NUMBER, **AREAT** IN OUT NUMBER)

**IS**

**BEGIN**

**AREAT** := (**BASE** \* **ALTURA**)/2;

**END;**

**Desde un bloque anónimo**, probaremos el procedimiento almacenado **SP\_AREA\_TRIANGULO** que usa 3 parámetros, enviándole como argumentos, los valores **100**, **50** y **0** en una línea de declaración en las variables locales **B, H, AREA**.

**SET SERVEROUTPUT ON;**

**DECLARE**

**B** NUMBER:=100; **H** NUMBER:=50; **AREA** NUMBER:=0;

**BEGIN**

**AREA\_TRIANGULO** (**B**, **H**, **AREA**);

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

DBMS\_OUTPUT.PUT\_LINE ('AREA DEL TRIANGULO');

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

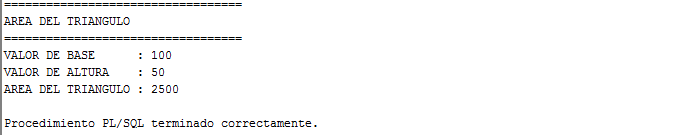
DBMS\_OUTPUT.PUT\_LINE ('VALOR DE BASE : ' || **B**);

DBMS\_OUTPUT.PUT\_LINE ('VALOR DE ALTURA : ' || **H**);

DBMS\_OUTPUT.PUT\_LINE ('AREA DEL TRIANGULO : ' || **AREA**);

**END;**

**Resultado:**



**PROCEDIMIENTOS ALMACENADOS BASADO EN DATOS**

**FILTROS – CONSULTAS (CURSORES)**

**FAVOR DE DESCARGAR EL ARCHIVO**

Semana11\_PL\_SQL\_1.sql

Semana11\_PL\_SQL\_2.sql

Se tiene la siguiente tabla de datos llamada **LIBRO**.

**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);

**Sus datos son los siguientes:**



**Ejemplo 5:**

Se desea crear un procedimiento almacenado, el cual permita el filtrado de libros, solicitándolas por el área del mismo libro y estas son: **LPROG**, **ANALISIS**, **ARTE**, **LIDEREAZGO**.

**SOLUCION:**

**CREATE OR REPLACE PROCEDURE** SP\_FILTRO\_AREA

(AREA IN VARCHAR2)

**IS**

**BEGIN**

**DECLARE CURSOR** FILTRO\_1 IS (**SELECT \* FROM LIBRO**

**WHERE AREALIB = AREA**);

A LIBRO.CODLIB**%TYPE**;

B LIBRO.TITULOLIB**%TYPE**;

C LIBRO.AUTORLIB**%TYPE**;

D LIBRO.EDITOLIB**%TYPE**;

E LIBRO.AREALIB**%TYPE**;

F LIBRO.PRECIOLIB**%TYPE**;

G LIBRO.NHOJASLIB**%TYPE**;

H LIBRO.ANOLIB**%TYPE**;

**BEGIN**

**OPEN** FILTRO\_1;

**FETCH** FILTRO\_1 **INTO** A, B, C, D, E, F, G, H;

**WHILE** FILTRO\_1**%FOUND**

**LOOP**

DBMS\_OUTPUT.PUT\_LINE (A || ' - ' || B || ' - ' ||

C || ' - ' || D || ' - ' || E ||

' - ' || F || ' - ' || G || ' - ' || H);

**FETCH** FILTRO\_1 **INTO** A, B, C, D, E, F, G, H;

**END LOOP;**

**CLOSE** FILTRO\_1;

**END;**

**END;**

**Probando el procedimiento almacenado SP\_FILTRO\_AREA**

**SET SERVEROUT ON;**

**BEGIN**

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

DBMS\_OUTPUT.PUT\_LINE ('LIBROS DE LENGUAJE DE PROGRAMACION');

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

**SP\_FILTRO\_AREA ('LPROG');**

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

DBMS\_OUTPUT.PUT\_LINE ('LIBROS DE ANALISIS');

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

**SP\_FILTRO\_AREA ('ANALISIS');**

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

DBMS\_OUTPUT.PUT\_LINE ('LIBROS DE ARTE');

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

**SP\_FILTRO\_AREA ('ARTE');**

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

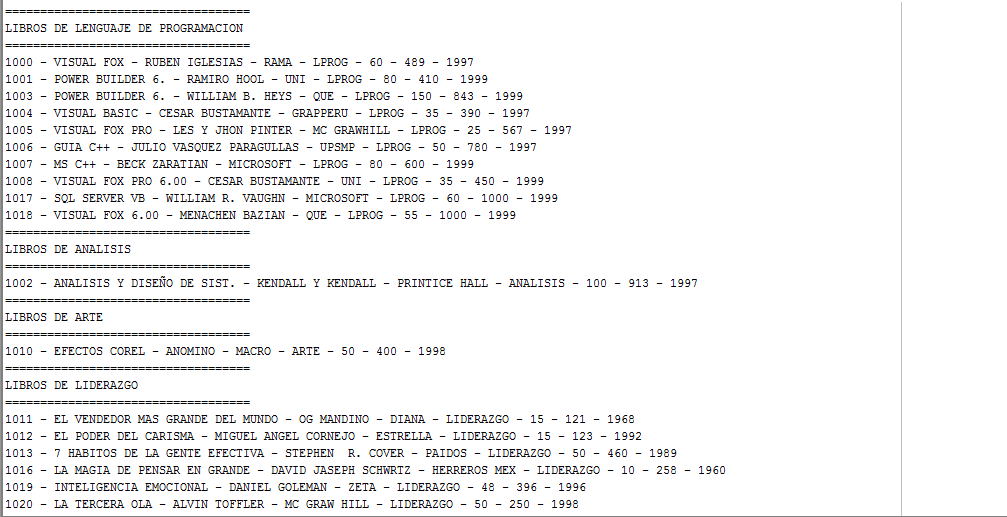
DBMS\_OUTPUT.PUT\_LINE ('LIBROS DE LIDERAZGO');

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

**SP\_FILTRO\_AREA ('LIDERAZGO');**

**END;**

**Resultado:**



**Ejemplo 6:**

Se tiene la tabla TRABAJADOR, con los siguientes datos:

**CREATE TABLE TRABAJADOR** (

COD\_TRA VARCHAR2 (5) NOT NULL,

AP\_TRA VARCHAR2 (15) NULL,

AM\_TRA VARCHAR2 (15) NULL,

NOM\_TRA VARCHAR2 (15) NULL,

FECHA\_NA DATE NULL,

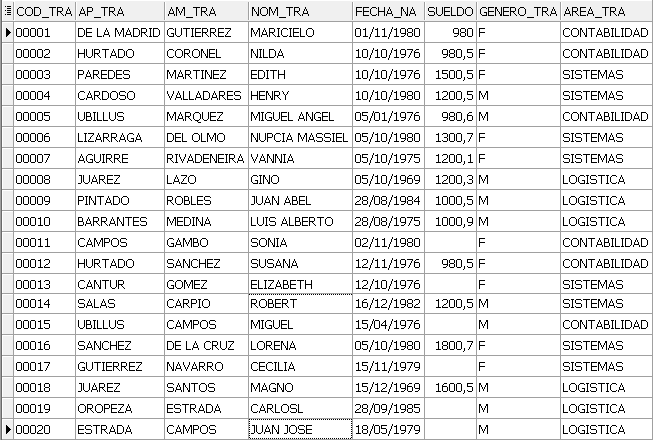
SUELDO NUMERIC (8,2) NULL,

GENERO\_TRA VARCHAR2 (1) NULL,

AREA\_TRA VARCHAR2 (30) NULL,

PRIMARY KEY (COD\_TRA));

**Y tiene los siguientes datos:**



Mediante un procedimiento almacenado se desea asignar el valor **1000** a un determinado trabajador que no tenga sueldo definido.

**CREATE OR REPLACE PROCEDURE** SP\_ACTUALIZA\_SUELDOS

(CODIGO IN VARCHAR2)

**IS**

**BEGIN**

**DECLARE CURSOR** FILTRO\_2 **IS** (**SELECT \* FROM TRABAJADOR**

**WHERE COD\_TRA = CODIGO**);

A TRABAJADOR.COD\_TRA**%TYPE**;

B TRABAJADOR.AP\_TRA**%TYPE**;

C TRABAJADOR.AM\_TRA**%TYPE**;

D TRABAJADOR.NOM\_TRA**%TYPE**;

E TRABAJADOR.FECHA\_NA**%TYPE**;

F TRABAJADOR.SUELDO**%TYPE**;

G TRABAJADOR.GENERO\_TRA**%TYPE**;

H TRABAJADOR.AREA\_TRA**%TYPE**;

**BEGIN**

**OPEN** FILTRO\_2;

**FETCH** FILTRO\_2 **INTO** A, B, C, D, E, F, G, H;

**WHILE** FILTRO\_2**%FOUND**

**LOOP**

IF F **IS NULL** THEN

**UPDATE TRABAJADOR SET SUELDO=1000**

**WHERE COD\_TRA=A;**

**COMMIT;**

END IF;

**FETCH** FILTRO\_2 **INTO** A, B, C, D, E, F, G, H;

**END LOOP;**

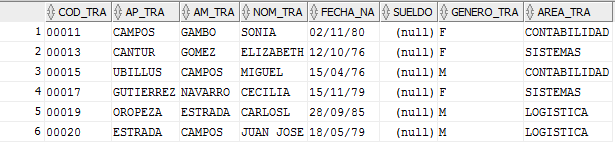
**CLOSE** FILTRO\_2;

**END;**

**END;**

Primero buscamos una persona que no tenga sueldo

**SELECT \* FROM TRABAJADOR;**



Probando el procedimiento almacenado desde un bloque anónimo, en este caso el trabajador seleccionado es el que lleva el código **‘00015’.**

Ahora se procede a cambiarle el sueldo por 1000.00

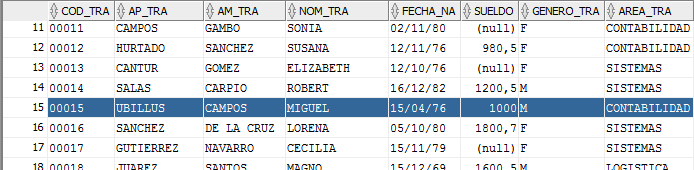
BEGIN

SP\_ACTUALIZA\_SUELDO ('00015');

END;

**VERIFICANDO**

**SELECT \* FROM TRABAJADOR;**



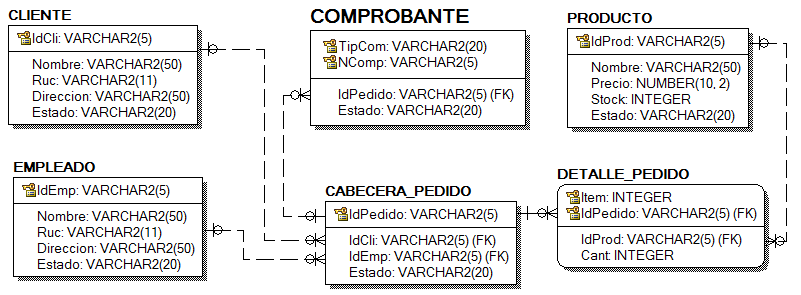
**CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA**

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 en base de datos Oracle.

**ACTIVIDAD VIRTUAL**

Revisar y analizar el material presentado del Tema 09.

**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.**

**Semana09\_PL\_SQL\_3.sql,**

**FAVOR DE DESCARGAR DICHO ARCHIVO.**

De acuerdo al material presentada, responda a las siguientes Preguntas:

**RESOLVER:**

Crear las siguientes Procedimientos Almacenados en PL/SQL y probarlas.

1. Los libros de LPROG, tendrán un incremento del 15% en sus precios, proceda a actualizar la tabla libros que cumplan esta condición.
2. Todos los trabajadores reciben un incremente en su sueldo en un 20%, proceda a actualizar la tabla trabajadora que cumplan esta condición.
3. Mediante un procedimiento almacenado se pide mostrar los productos, precio, cantidad e importes de cualquier comprobante de pago.