EJERCICIOS PROCEDIMIENTOS – FUNCIONES –DISPARADORES. SOLUCIONES

1) Confeccionar el programa “Hola, Mundo” en PL/SQL

```
-- Programa "Hola Mundo" en PL/SQL
-- Colocar la variable SERVEROUTPUT a ON antes de ejecutar el bloque
SET SERVEROUTPUT ON;

/* Se trata de un bloque anónimo.
Al no tener nombre, Oracle no lo guarda en su catálogo
Por tanto, se compilará cada vez que lo ejecutemos*/
BEGIN
     DBMS_OUTPUT.PUT_LINE ('¡¡Hola, Mundo !!');
END;
```

2) Escribir un procedimiento PL/SQL que pida dos números enteros e imprima su suma

```
-- Procedimiento Para realizar la suma de dos números en PL/SQL

CREATE OR REPLACE PROCEDURE SumaDos (num1 INTEGER, num2 INTEGER)
-- Bloque de declaraciones. Aquí definimos variables y objetos locales
AS
     resultado INTEGER;
      salida VARCHAR2(200);
--Bloque ejecutable
BEGIN
     resultado := num1 + num2;
     salida:= 'La suma de  ' || num1 || '  y de  ' || num2 || '  es igual a :
' || resultado;
      DBMS_OUTPUT.PUT_LINE (salida);
END SumaDos;
```

-- Para probar el procedimiento una vez compilado, lanzar EXEC SumaDos(30,40),...

3) Modificar el procedimiento anterior para que pueda realizar cualquiera de las operaciones básicas con los dos números enteros introducidos. Además de los dos números, el procedimiento recibirá un tercer parámetro con el tipo de operación (+ - * /).

Deberán controlarse los siguientes errores:

- Se intenta dividir entre cero (“When ZERO_DIVIDE)
- Se intenta realizar una operación no permitida (excepción personalizada)

```
  CREATE OR REPLACE PROCEDURE Operacion (num1 INTEGER, num2 INTEGER, operando
VARCHAR2)
  -- Bloque de declaraciones. Aquí definimos variables y objetos locales
  AS
       resultado INTEGER;
        salida VARCHAR2(200);
      no_permitida EXCEPTION;

  --Bloque ejecutable
  BEGIN
     CASE operando
           WHEN '+' THEN
                 resultado := num1 + num2;
                 salida:= 'La suma de  ' || num1 || '  y de  ' || num2 || '  es
igual a :  ' || resultado;

           WHEN '-' THEN

                 resultado := num1 - num2;
                 salida:= 'La diferencia de  ' || num1 || '  y de  ' || num2 ||
'  es igual a :  ' || resultado;

           WHEN '*' THEN

                 resultado := num1 * num2;
                 salida:= 'El producto de  ' || num1 || '  y de  ' || num2 || '
es igual a :  ' || resultado;

           WHEN '/' THEN

                 resultado := num1 / num2;
                 salida:= 'El cociente  de  ' || num1 || '  y de  ' || num2 ||
'  es igual a :  ' || resultado;

           ELSE
                 RAISE no_permitida;
           END CASE;   -- Aquí finaliza el bloque CASE

        DBMS_OUTPUT.PUT_LINE (salida);
  --Bloque para el tratamiento de los errores
```

```
EXCEPTION
          WHEN ZERO_DIVIDE THEN
                DBMS_OUTPUT.PUT_LINE ('No sabemos cómo dividir entre cero');

          WHEN no_permitida THEN
                DBMS_OUTPUT.PUT_LINE (' No sé qué operación es  '|| operando);
END Operacion;

/* Para probarlo, ejecutar los siguientes casos y ver el resultado
exec operacion(30,40,'+')

exec operacion(100,40,'-')

exec operacion(30,40,'*')

exec operacion(100,20,'/')

exec operacion(100,40,'/')

exec operacion(100,0,'/')

exec operacion(100,0,'%')
*/
```

4) Escribir un bloque anónimo PL/SQL que, utilizando una variable de sustitución, permita contar e imprimir en pantalla el número de registros en una tabla cuyo nombre se le indicará en tiempo de ejecución.

```
-- Bloque anónimo para contar el número de filas de una table
DECLARE
            NumFilas INTEGER;
BEGIN
      SELECT COUNT (*) INTO NumFilas
                  FROM &Tabla;
     DBMS_OUTPUT.PUT_LINE ( 'Número de filas: ' || NumFilas);
EXCEPTION
   When NO_DATA_FOUND then
          DBMS_OUTPUT.PUT_LINE ( 'La tabla no tiene nada’);
END;
```

5) Escribir una función que realice la separación de los componentes de una cadena que están delimitados por un carácter que actuará como separador (función “split”). Cada uno de estos

componentes se guardará en una tabla llamada “Temporal” (que se vaciará antes de realizar la separación). En definitiva, la función tomará como parámetros de entrada:

- La cadena a separar
- El carácter que se tomará como separador

Como salida, retornará el número de componentes separados

```
CREATE OR REPLACE FUNCTION SepararCadena (Cadena VARCHAR2, Sep VARCHAR2)
RETURN INTEGER
AS
                NumComponentes INTEGER default 0;
    Posicion INTEGER;
     BEGIN
                        Posicion := INSTR (Cadena, Sep);
          IF Posicion >0 THEN
                INSERT INTO Temporal VALUES (SUBSTR( Cadena, 1, Posicion-1));
                NumComponentes:= SepararCadena(SUBSTR(Cadena, Posicion+1,
Length(Cadena)-Posicion),Sep)+1;
                RETURN NumComponentes;
          ELSIF LENGTH(Cadena)>0 THEN
                 INSERT INTO Temporal VALUES (Cadena);
                RETURN 1;
          ELSE
                RETURN 0;
          END IF;

 END SepararCadena;
```

Supongamos que la tabla Entregas tenía la estructura siguiente:

Deliveries(degree,SubjectName, workNum, teamNum, delivery_date.mark)

En este caso el disparador quedaría así:

```
--Disparador para cada registro que se añada o se borre sobre la Tabla Deliveries
CREATE OR REPLACE TRIGGER TriggerDeliveries
AFTER  INSERT OR DELETE ON Deliveries FOR EACH ROW
DECLARE
-- Variables que necesitamos
fecha VARCHAR2(20):= TO_CHAR(SYSDATE, 'DD-MM-YYYY,HH24:MI:SS');
mensaje VARCHAR2(300);

BEGIN
IF INSERTING THEN  -- Se insertó el registro, por tanto hay que recuperar sus valores con :new
          mensaje:= 'Fecha: ' || fecha || '* Asignatura :' || :new.degree ||':'||:new.SubjectName|| '* Núm Práctica :' ;
          mensaje:= mensaje || :new.workNum || ' , ' || '* Grupo :' || :new.teamNum'  || '* INSERCIÓN* ';
ELSIF DELETING THEN   -- El registro se ha borrado, por tanto hay que recuperar sus valores con :old
         mensaje:= 'Fecha: ' || fecha || '* Asignatura :' || :old.degree ||':'||:old.SubjectName|| '* Núm Práctica :' ;
          mensaje:= mensaje || :old.workNum || ' , ' || '* Grupo :' || :old.teamNum'  || '* BORRADO* ';
END IF;
      -- Insertamos el registro de auditoría en la tabla correspondiente
     INSERT INTO AuditDeliveries  VALUES (mensaje);
END TriggerDeliveries;    -- Fin del disparador
```

Propuesta: Ampliar el disparador anterior para que también incluya el caso en que se modifique la calificación de una entrega, auditando la calificación anterior y la nueva.

```
--Disparador para cada registro que se añada o se borre sobre la Tabla Deliveries
CREATE OR REPLACE TRIGGER TriggerDeliveries
AFTER  INSERT OR DELETE OR UPDATE OF mark
 ON Deliveries FOR EACH ROW
DECLARE
-- Variables que necesitamos
fecha VARCHAR2(20):= TO_CHAR(SYSDATE, 'DD-MM-YYYY,HH24:MI:SS');
mensaje VARCHAR2(300);
```

```
  BEGIN
  IF INSERTING THEN  -- Se insertó el registro, por tanto hay que recuperar sus valores con :new
            mensaje:= 'Fecha: ' || fecha || '* Asignatura :' || :new.degree ||':'||:new.SubjectName|| '* Núm Práctica :';
            mensaje:= mensaje || :new.workNum || ' , ' || '* Grupo :' || :new.teamNum || '* INSERCIÓN*';
  ELSIF DELETING THEN   -- El registro se ha borrado, por tanto hay que recuperar sus valores con :old
          mensaje:= 'Fecha: ' || fecha || '* Asignatura :' || :old.degree ||':'||:old.SubjectName|| '* Núm Práctica :' ;
            mensaje:= mensaje || :old.workNum || ' , ' || '* Grupo :' || :old.teamNum || '* BORRADO* ';
  ELSIF UPDATING ('mark') THEN  -- Modifico la calificación
          mensaje:= 'Fecha: ' || fecha || '* Asignatura :' || :old.degree ||':'||:old.SubjectName|| '* Núm Práctica :' ;
            mensaje:= mensaje || :old.workNum || ' , ' || '* Grupo :' || :old.teamNum ;
      mensaje:= mensaje || 'Calificamensaje:= ' La fecha de entrega está fuera del período de entrega establecido para este trabajo';
            dbms_output.put_line (mensaje);izacion*';

  END IF;
       -- Insertamos el registro de auditoría en la tabla correspondiente
      INSERT INTO AuditDeliveries VALUES (mensaje);
  END TriggerDeliveries;    -- Fin del disparador
```

8) Usando el ejemplo de las prácticas de laboratorio, construir un disparador que impida la insertar entregas de prácticas fuera del período de entrega establecido.

En este caso, se necesita utilizar la tabla siguiente:

Labwork (degree, subjectName, worknum, startdate, duedate)

```
  CREATE OR REPLACE TRIGGER CheckDeliveriesDate
  BEFORE  INSERT ON Deliveries FOR EACH ROW
  DECLARE
  -- Variables que necesitamos
  fecha_ini DATE;
  fecha_fin DATE;
  mensaje VARCHAR2(300);
  miError EXCEPTION;

  BEGIN

  SELECT startdate, duedate  INTO fecha_ini, fecha_fin FROM  labWork
```

```
  WHERE      degree=   :new.degree   and   subjectName=   :new.subjectName   and
workNum= :new.workNum;


  IF  (:new.delivery_date NOT BETWEEN  fecha_ini and fecha_fin) THEN
          RAISE miError;
  ELSE
      mensaje:= ' La fecha de entrega está dentro del plazo';
          dbms_output.put_line (mensaje);


  END IF;


  EXCEPTION
     WHEN miError THEN
          /* Si el disparador se invoca como consecuencia de una orden INSERT,
  hemos de usar este truco de dar a los atributos de la clave primaria un valor
imposible -null- para que la operación falle, además de mostrarnos el mensaje de
error
  */
          :new.degree := null;
          :new.subjectName:= null;
          :new.workNum:= null;
          mensaje:= ' La fecha de entrega está fuera del período de entrega
establecido para este trabajo';
          dbms_output.put_line (mensaje);


  END CheckDeliveriesDate;    -- Fin del disparador
```