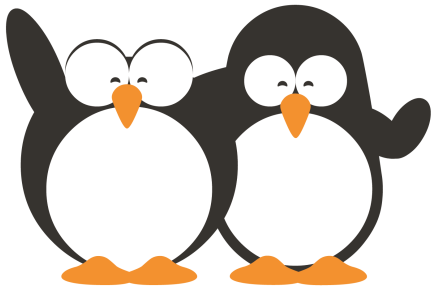
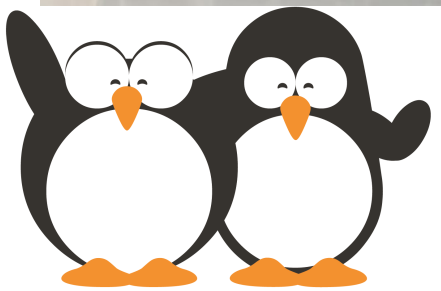
ÍNDICE

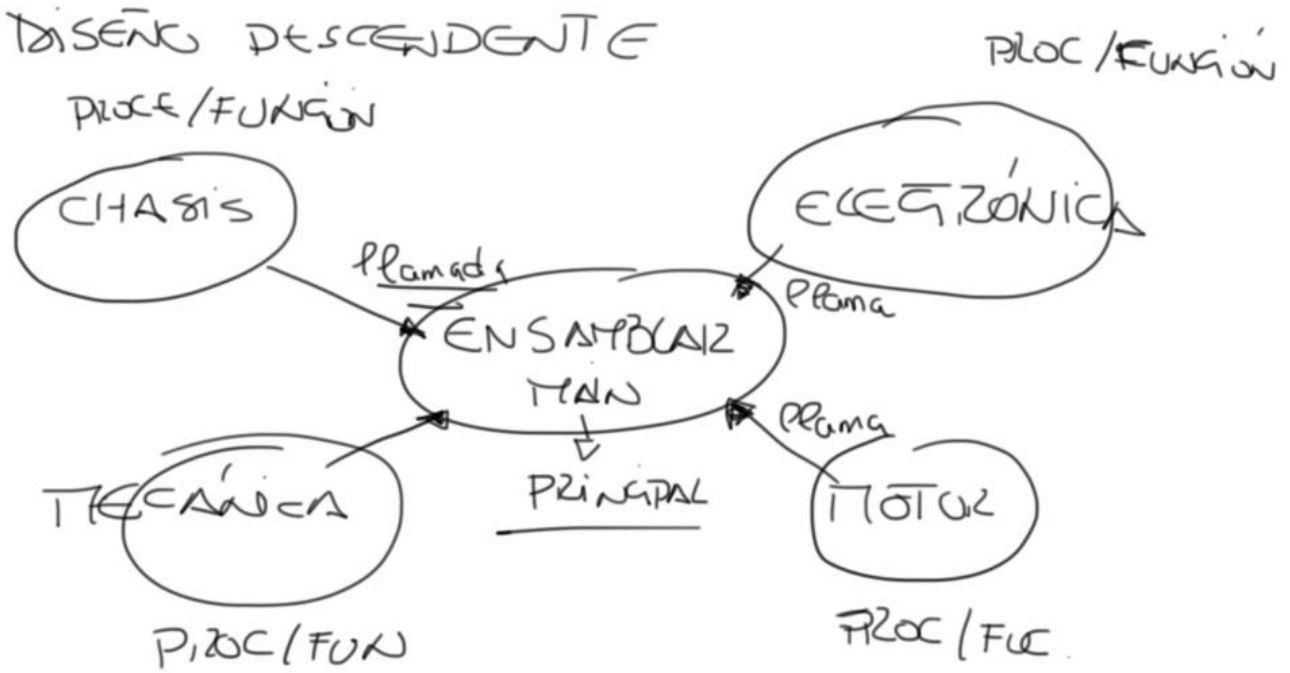
1. Procedimientos Y Funciones
2. Disparadores
3. Dudas

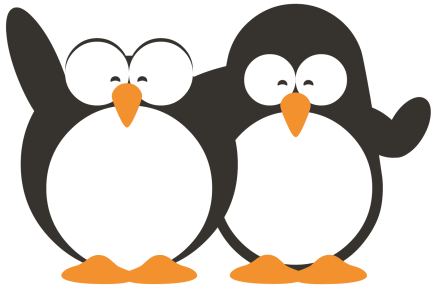


DISEÑO DESCENDENTE



DISEÑO DESCENDENTE



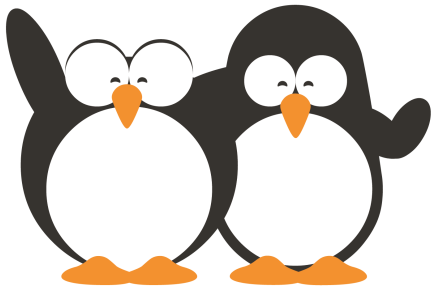


Procedimientos

Introducción

- Un procedimiento es un subprograma que ejecuta una acción específica y que no devuelve ningún valor. Un procedimiento tiene un nombre, un conjunto de parámetros (opcional) y un bloque de código.
- La sintaxis para declararlo es:

```
CREATE [OR REPLACE]
PROCEDURE <procedure_name> [(<param1> [IN|OUT|IN OUT] <type>,
                             <param2> [IN|OUT|IN OUT] <type>,
...)]
IS
 -- Declaración de variables locales
BEGIN
  -- Sentencias
[EXCEPTION]
  -- Sentencias control de excepcion
END [<procedure_name>];
```

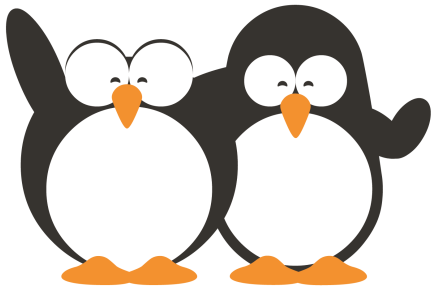


Procedimientos

Ejemplo

- En el ejemplo hemos creado un procedimiento que muestra los números desde el 1 hasta el valor pasado por parámetro.

```
create or replace procedure joc(i number) is
 num constant number := 2525;
begin
 case
   when i < num then
     dbms_output.put_line('El  numero es mas pequeño, continua buscando...');
   when i > num then
     dbms_output.put_line('El  numero es mas grande, continua buscando...');
   when i = num then
     dbms_output.put_line('HAS  ACERTADO !!!!');
  end case;
end;
```



- Para llamar al procedimiento una vez creado simplemente debemos poner su nombre y, entre paréntesis el parámetro o parámetros necesarios.

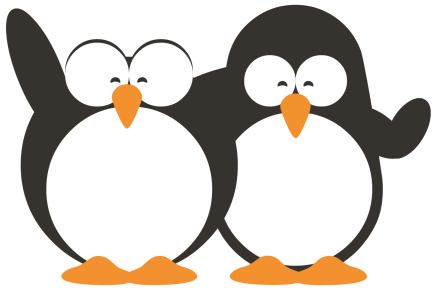
Procedimientos

Parámetros

- Para declarar los parámetros de un procedimiento debemos seguir la siguiente sintaxis.

  <param1> **[IN|OUT|IN OUT]** *<type>*

  - <param1> es el nombre del parámetro
  - <type> es el tipo de dato
  - [IN|OUT|IN OUT] → Indica la forma en la que pasamos el parámetro al procedimiento.
    - IN: El parámetro es de entrada, significa que la variable original (Fuera de la función) no se verá afectada.
    - OUT: El parámetro es de salida. El parámetro se usará para almacenar un valor de salida del procedimiento. No se puede emplear como parámetro de entrada.
    - IN OUT: El parámetro actúa como parámetro de entrada/salida, es decir, se emplea para pasar un valor al procedimiento y, además, como forma de almacenar un valor de salida.



PARÁMETROS IN –OUT / IN

pass by reference

cup =

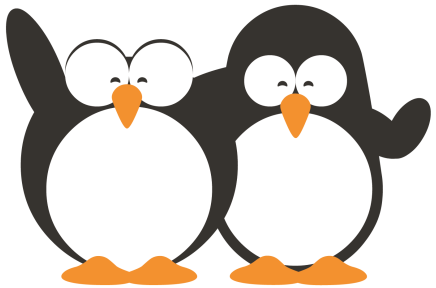
```
fillCup(      )
```

pass by value

cup =

```
fillCup(      )
```

www.penjee.com



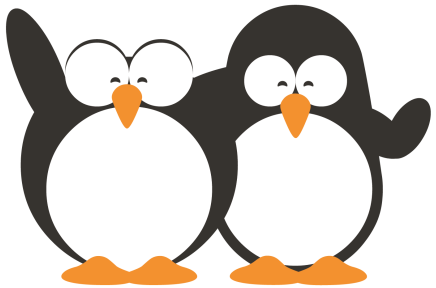
Funciones

Introducción

- Las funciones PL/SQL son unidades funcionales similares a los procedimientos, la principal diferencia radica en que las funciones devuelven un resultado tras su ejecución.
- Es necesario indicar el tipo de dato que la función va a devolver en la definición de la misma.
- Sintaxis:

```
CREATE [OR REPLACE]
FUNCTION <fn_name>[(<param1> IN <type>, <param2> IN <type>, ...)]
RETURN <return_type>
IS
  result <return_type>;
BEGIN

  return(result);
[EXCEPTION]
  -- Sentencias control de excepción
END [<fn_name>];
```

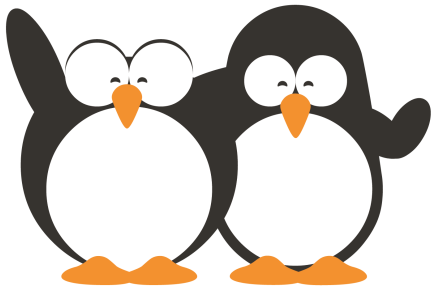


Funciones

Ejemplo

- En el ejemplo hemos programado una función que recibe dos números por parámetro y devuelve la suma de ambos.

```
CREATE OR REPLACE
FUNCTION sumarNumeros(num1 number, num2 number)
RETURN NUMBER
IS
   resultado NUMBER;
BEGIN
   resultado:=num1+num2;
   return(resultado);
END ;
```



Funciones

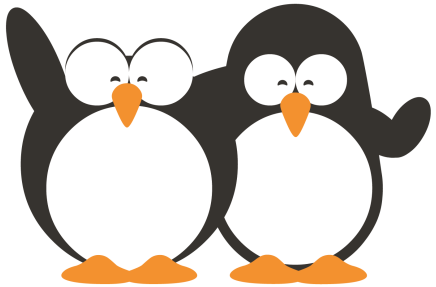
Llamadas a funciones

- Para llamar a una función se hace de forma similar a como llamábamos a los procedimientos, sin embargo, ahora podemos usar el dato que nos devuelve la función asignando a una variable o bien imprimiendo por pantalla.

```
Begin
  sumarNumeros(2,2);
End;
```

```
Declare
  suma Number;
Begin
  suma:=sumarNumeros(2,2);
end;
```

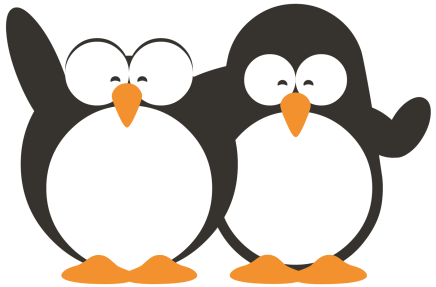
```
Begin
  DBMS_OUTPUT.PUT_LINE(sumarNumeros(2,2));
end;
```



DISPARADORES

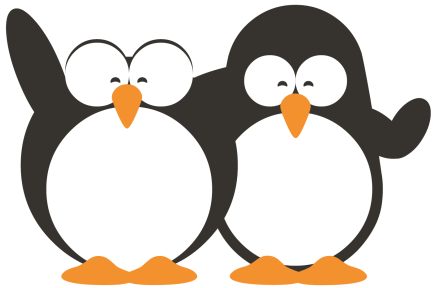
- Un trigger es un bloque PL/SQL asociado a una tabla, que se ejecuta como consecuencia de una determinada instrucción SQL (una operación DML: INSERT, UPDATE o DELETE) sobre dicha tabla.

```
CREATE [OR REPLACE] TRIGGER <nombre_trigger>
{BEFORE|AFTER}
          {DELETE|INSERT|UPDATE [OF col1, col2, ..., colN]
        [OR {DELETE|INSERT|UPDATE [OF col1, col2, ..., colN]...]}
ON <nombre_tabla>
[FOR EACH ROW [WHEN (<condicion>)]]
DECLARE
 -- variables locales
BEGIN
 -- Sentencias
[EXCEPTION]
 -- Sentencias control de excepcion
END <nombre_trigger>;
```



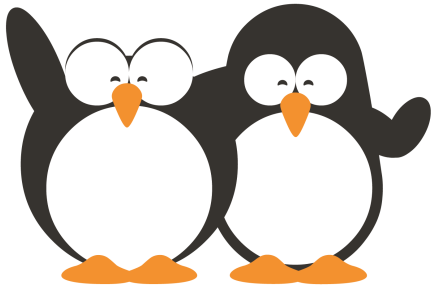
DISPARADORES

- Los triggers pueden definirse para las operaciones INSERT, UPDATE o DELETE, y pueden ejecutarse antes o después de la operación. (After/ Before)
- El modificador FOR EACH ROW indica que el trigger se disparará cada vez que se realizan operaciones sobre una fila de la tabla.
- Si se acompaña del modificador WHEN, se establece una restricción; el trigger solo actuará, sobre las filas que satisfagan la restricción.



DISPARADORES

- Dentro del ambito de un trigger disponemos de las variables OLD y NEW . Estas variables se utilizan del mismo modo que cualquier otra variable PL/SQL, con la salvedad de que no es necesario declararlas, son de tipo **%ROWTYPE** y contienen una copia del registro antes (OLD) y despues(NEW) de la acción SQL (INSERT, UPDATE, DELTE) que ha ejecutado el trigger. Utilizando esta variable podemos acceder a los datos que se están insertando, actualizando o borrando.



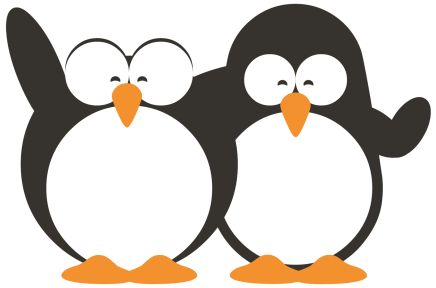
DISPARADORES

- El siguiente ejemplo muestra un trigger que inserta un registro en la tabla PRECIOS_PRODUCTOS cada vez que insertamos un nuevo registro en la tabla PRODUTOS:

```
CREATE OR REPLACE TRIGGER TR_PRODUCTOS_01
 AFTER INSERT ON PRODUCTOS
 FOR EACH ROW
DECLARE
 -- local variables
BEGIN
 INSERT INTO PRECIOS_PRODUCTOS
 (CO_PRODUCTO,PRECIO,FX_ACTUALIZACION)
 VALUES
 (:NEW.CO_PRODUCTO,100,SYSDATE);
END ;
```

El trigger se ejecutará cuando sobre la tabla PRODUCTOS se ejecute una sentencia INSERT.

```
INSERT INTO PRODUCTOS
(CO_PRODUCTO, DESCRIPCION)
VALUES
('000100','PRODUCTO 000100');
```



DISPARADORES

EJERCICIO 1

```
create or replace TRIGGER auditasueldo
AFTER UPDATE of SALARIO ON emp
FOR EACH ROW
DECLARE
 ca number (5);
BEGIN
  select (max(id_cambio)+1) into ca
from auditaemple;
if (ca is null )then ca:=1;
end if;
 INSERT into auditaemple values (ca,'El salario del empleado '||:old.emp_no||'sal viejo'||:old.salario||'sal nuevo'||:new.salario,sysdate);
END;
```