7.- PROCEDIMIENTOS

7.1.- INTRODUCCIÓN

Un procedimiento es un bloque PL nombrado que realiza una determinada acción. Está almacenado en la Base de Datos como un objeto de la misma y puede ser ejecutado múltiples veces.

Los procedimientos promueven la reutilización y el mantenimiento de los programas, y una vez validados, pueden utilizarse en muchas aplicaciones. Si la definición cambia, solo se ve afectado el procedimiento y esto simplifica en gran mediada el mantenimiento.

Sintaxis:

```
CREATE [OR REPLACE] PROCEDURE nb_procedimiento
        (argumento1 [ modo 1] tipo_de_dato1,
         argumento2  [ modo2] tipo_de_dato2,
        ...)
IS | AS
Bloque PL/SQL;
```

Donde:

ARGUMENTO: nombre de la variable PL cuyo valor se recibe
MODO: IN, OUT, IN OUT.
TIPO_DE_DATO: tipo de dato del argumento.

Creación de un procedimiento en SQL * PLUS

1. Introducir CREATE PROCEDURE en un editor y salvarlo en un SCRIPT.
2. Desde SQL*PLUS, ejecutar el SCRIPT, para compilar el código fuente y almacenarlo en la Base de Datos.
3. Llamar al procedimiento y ejecutarlo si no tiene errores.

Los procedimientos no llevan la palabra DECLARE. Si los procedimientos se crean dentro de bloques PL no se introducirá la cláusula CREATE PROCEDURE se pone únicamente PROCEDURE nb_procedimiento

7.2.- MODOS DE PARÁMETROS.

Podemos transferir valores desde el entorno de llamada a través de parámetros. Existen 3 modos de parámetros:

IN	OUT	IN OUT
Por defecto.	Tiene que especificarse.	Tiene que especificarse.
Valor que se pasa al subprograma.	Devuelve al entorno de llamada.	Valor que se pasa al subprograma. Devuelve al entorno de llamada.
Parámetro formal constante.	Variable no inicializada.	Variable inicializada.
Parámetro actual puede ser un literal, expresión, constante o variable inicializada.	Tiene que ser una variable.	Tiene que ser una variable.

IN

Pasa un valor constante desde el entorno de llamada al procedimiento. Valor por defecto, es el valor que se pasa a un subprograma, pudiendo ser literal, expresión, constante o variable.

Ejemplo:

Crear un procedimiento que incremente el salario de un empleado un 10%, se introducirá el código del empleado como parámetro de entrada.

```
CREATE OR REPLACE PROCEDURE PROC1
(V_ID IN EMP.EMPNO%TYPE)
IS
BEGIN
        UPDATE EMP SET SAL= SAL*1.10 WHERE EMPNO = V_ID;
END PROC1;
/

SQL> EXECUTE PROC1(7934);
```

OUT

Pasa un valor desde el procedimiento al entorno de llamada. Tiene que ser una variable.

Ejemplo**:**

Crear un procedimiento QUERY_EMP con tres parámetros: uno de entrada, que permita introducir el código del empleado y dos de salida que permita visualizar el salario y el oficio.

```
CREATE OR REPLACE PROCEDURE QUERY_EMP
        (V_ID IN EMP.EMPNO%TYPE,
        V_SAL OUT EMP.SAL%TYPE,
        V_JOB OUT EMP.JOB%TYPE)
```

```
IS
BEGIN
        SELECT SAL, JOB INTO V_SAL,V_JOB FROM EMP
        WHERE EMPNO = V_ID;
END QUERY_EMP;
/
```

El procedimiento está creado, ahora hay crear unas variables host y ejecutarlo, puesto que no hay programa principal que llame al procedimiento.

```
SQL> VARIABLE G_SAL  NUMBER;
SQL> VARIABLE G_JOB VARCHAR2(15)
SQL> EXECUTE QUERY_EMP (7839,:G_SAL,:G_JOB)
SQL> PRINT G_SAL
SQL> PRINT G_JOB
```

IN OUT

Valor que se pasa al procedimiento y vuelve al entorno de llamada. Tiene que ser una variable inicializada.

Ejemplo:

Crear un procedimiento llamado CAMBIO, que introduciendo una cadena la devuelva entre paréntesis.

```
CREATE OR REPLACE PROCEDURE CAMBIO
(V_CAM IN OUT VARCHAR2)
IS
BEGIN
V_CAM :=  '( ' || V_CAM || ')' ;
END CAMBIO;
/

SQL> VARIABLE G_CAM  VARCHAR2(25);
BEGIN
 :G_CAM := 'HOLA';
END;
/

SQL> EXECUTE CAMBIO (:G_CAM);
PRINT G_CAM;
```

7.3.- MÉTODOS PARA PASAR PARÁMETROS

Existen tres métodos para pasar parámetros:

1. **Posicional**: Lista de valores en el orden que se declaran los parámetros
2. **Asociación nombrada**: Lista de valores en orden arbitrario mediante la asignación (=>).
3. **Mixto**: Mezcla de los dos anteriores. Lista de los primeros valores posicionalmente, y el resto, utilizando la sintaxis especial del método nombrado.

Para llamar desde un programa principal a un procedimiento se escribe: NB_PROCEDIMIENTO (PARÁMETROS).

Ejemplo:

Crear un procedimiento llamado AÑADIR que permita introducir elementos en la tabla DEPT, teniendo en cuenta que el código de departamento es obligatorio y que si no se asigna valor al resto de los campos tendrán un valor por defecto. Crear posteriormente un bloque PL anónimo que invoque al procedimiento y añada registros por los métodos anteriormente explicados.

```
CREATE OR REPLACE PROCEDURE AÑADIR
   (V_DEPTNO IN NUMBER,
        V_DNAME IN DEPT.DNAME%TYPE DEFAULT 'XXXXX',
        V_LOC IN DEPT.LOC%TYPE DEFAULT 'CCCCC')
IS
BEGIN
        INSERT INTO DEPT VALUES (V_DEPTNO, V_DNAME, V_LOC);
END AÑADIR;
/
```

Programa principal:

```
BEGIN
        AÑADIR(60);
        AÑADIR(70,'DATOS','MADRID');
        AÑADIR (V_LOC=> 'BARCELONA',V_DEPTNO=> 75);
END;
/
```

7.4.- MÁS SOBRE PROCEDIMIENTOS

	SQL * PLUS
Ejecución de forma aislada de un procedimiento.	EXECUTE nb_procedimiento (parámetros);
Ejecución de procedimiento desde un programa principal.	Nb_procedimiento (parámetros);
Borrado de un procedimiento.	DROP PROCEDURE nb_procedimiento;

Los procedimientos los encontramos en la tabla USER_OBJECTS del Diccionario de Datos.

```
SELECT OBJECT_NAME, OBJECT_TYPE, STATUS
FROM USER_OBJECTS
WHERE OBJECT_TYPE = 'PROCEDURE';
```

La tabla USER_SOURCE del Diccionario de Datos, permite ver el código fuente de los procedimientos, mediante la siguiente sentencia:

```
SELECT LINE, SUBSTR (TEXT,1,60)
FROM USER_SOURCE
WHERE NAME = 'NB_PROCEDURE';
```

Para volver a compilar un procedimiento utilizamos la orden ALTER

```
ALTER { PROCEDURE|FUNCTION} nb_subprograma  COMPILE;
```

Esto vale tanto para procedimientos como para funciones.