BASE DE DATOS II
UNIDAD II

PROCEDIMIENTOS ALMACENADOS, FUNCIONES Y PAQUETES

PROCEDIMIENTOS ALMACENADOS

Un procedimiento almacenado (SP) es un subprograma que ejecuta una acción especifica y que no devuelve ningún valor. Un procedimiento tiene un nombre, un conjunto de parámetros (opcional) y un bloque de código.

La sintaxis de un procedimiento almacenado es la siguiente:

```
CREATE OR REPLACE PROCEDURE <NOMBRE SP>(<parámetro 1>
<IN|OUT|IN OUT> <tipo de dato>,…, <parámetro n> <IN|OUT|IN OUT>
<tipo de dato>)
IS
        <DECLARACIÓN DE VARIABLES>
BEGIN
        <INSTRUCCIONES SQL>

        <BLOQUE DE EXCEPCIONES>
END
```

PROCEDIMIENTOS ALMACENADOS DECLARACOIÓN DE PARÁMETROS

- **Nombre parámetro**: es el nombre que queramos dar al parámetro. Podemos utilizar múltiples parámetros. En caso de no necesitarlos, podemos omitir los paréntesis y los nombres de los parámetros.
- **IN**: especifica que el parámetro es de entrada y que por tanto dicho parámetro tiene que tener un valor en el momento de llamar a la función o procedimiento. Si no se especifica nada, los parámetros son por defecto de tipo entrada.
- **OUT**: especifica que se trata de un parámetro de salida. Son parámetros cuyo valor es devuelto, después de la ejecución del procedimiento, al bloque PL/SQL que lo llamó.
- **IN OUT**: Son parámetros de entrada y salida a la vez.
- **Tipo-de-dato**: Indica el tipo de dato PLSQL que corresponde al parámetro (NUMBER, VARCHAR2, etc).

PROCEDIMIENTOS ALMACENADOS PARÁMETROS DE ENTRADA Y SALIDA

IN – entrada
OUT – salida
IN OUT – entrada salida

Parámetro IN – entrada:
El comportamiento común de este tipos de parámetros es estar siempre pendiente de recibir un valor para ser procesado. Acepta valores por defectos haciendo uso del operador de asignación (:=) o utilizando la palabra reservada DEFAULT.

Parámetro OUT – salida:
Los parámetros de salida inicialmente arrancan como NULL, es decir al iniciar el proceso el parámetro estará como nulo o en todo caso, puede tomar un valor inicial y ser cambiado durante la ejecución del proceso; y al finalizar el proceso se encarga de tomar o asignar el último valor al parámetro de salida.

Parámetro IN OUT – entrada salida:
Los parámetros de entrada salida comúnmente cumplen las 2 funcionalidades arriba mencionadas con la diferencia de que si no se define un valor de entrada entonces retornará un valor nulo o retornará el último valor que se le ha asignado.

PROCEDIMIENTOS ALMACENADOS

El uso de OR REPLACE permite sobrescribir un procedimiento existente. Si se omite, y el procedimiento existe, se producirá un error.

La sintaxis del SP es muy parecida a la de un bloque anónimo, salvo porque se reemplaza la sección DECLARE por la secuencia PROCEDURE … IS y 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 pueden ser de entrada (IN), de salida (OUT) o de entrada salida (IN OUT). El valor por defecto es IN, y se toma ese valor en caso de que no especifiquemos nada.

PROCEDIMIENTOS ALMACENADOS

Ejemplo:

```
CREATE OR REPLACE PROCEDURE guardarPais (codPais NUMBER, nombre VARCHAR2)
IS
BEGIN
    INSERT INTO TBLPAISES VALUES(codPais, nombre);
    COMMIT;
END;
```

FUNCIONES

Una función es un subprograma que devuelve un valor. La sintaxis para construir funciones es la siguiente:

CREATE OR REPLACE FUNCTION <nombre funcion> >(<parámetro 1> <IN|OUT|IN OUT> <tipo de dato>,…, <parámetro n> <IN|OUT|IN OUT> <tipo de dato>)

RETURN <tipo de dato>

IS

<DECLARACIÓN DE VARIABLES>

BEGIN

<INSTRUCCIONES SQL>

RETURN <VALOR O VARIABLE A RETORNAR>

<BLOQUE DE EXCEPCIONES>

END;

FUNCIONES

El uso de OR REPLACE permite sobrescribir una función existente. Si se omite, y la función existe, se producirá un error. La sintaxis de los parámetros es la misma que en los procedimientos almacenados.

Ejemplo:

```
CREATE OR REPLACE FUNCTION nombHabitante (idHab NUMBER)
RETURN VARCHAR2
IS
        nombre VARCHAR2(100);
BEGIN
        SELECT nombreHabitante INTO nombre FROM TBLHABITANTES
        WHERE idHabitante=idHab;
        RETURN nombre;
END;
```

DEFINICIÓN DE PAQUETES

Un Paquete es un objeto PL/Sql que agrupa lógicamente otros objetos PL/Sql relacionados entre sí, encapsulándolos y convirtiéndolos en una unidad dentro de la base de datos.

Los paquetes están divididos en 2 partes: especificación o cabecera (obligatoria) y cuerpo (no obligatoria). La especificación o encabezado es donde se declaran los tipos, variables, constantes, excepciones, cursores, procedimientos y funciones que podrán ser invocados desde fuera del paquete.

Para acceder a los elementos declarados en un paquete basta con anteceder el nombre del objeto referenciado con el nombre del paquete donde está declarado y un punto, de esta manera: **Paquete.Objeto** donde Objeto puede ser un tipo, una variable, un cursor, un procedimiento o una función declarados dentro del paquete.

DEFINICIÓN DE PAQUETES

La creación de un paquete pasa por dos fases:

- Crear la cabecera del paquete donde se definen que procedimientos, funciones, variables, cursores, etc. Están disponibles para su uso posterior fuera del paquete. En esta parte solo se declaran los objetos, no se implementa el código.
- Crear el cuerpo del paquete, donde se definen los bloques de código de las funciones y procedimientos definidos en la cabecera del paquete.

DEFINICIÓN DE PAQUETES

Para crear la cabecera del paquete utilizaremos la siguiente instrucción:

```
CREATE OR REPLACE PACKAGE <NOMBRE DEL PAQUETE>
IS
    <VARIABLES>
    <ENCABEZADO DE FUNCIONES>
    < ENCABEZADO DE PROCEDIMIENTOS>END;
```

Para crear el cuerpo del paquete utilizaremos la siguiente instrucción:

```
CREATE OR REPLACE PACKAGE BODY <NOMBRE DEL PAQUETE>
IS
    <CUERPO DE FUNCIONES>
    < CUERPO DE PROCEDIMIENTOS>
END;
```

DEFINICIÓN DE PAQUETES

Hay que tener en cuenta que toda declaración de función o procedimiento debe estar dentro del cuerpo del paquete, y que todo bloque de código contenido dentro del cuerpo debe estar declarado dentro de la cabecera de paquete.