Tabla de contenido

[1. Introducción 2](#_Toc133914415)

[2. Bloques anónimos 2](#_Toc133914416)

[3. Tipos de datos 2](#_Toc133914417)

[4. Variables 2](#_Toc133914418)

[5. Operadores 2](#_Toc133914419)

[6. Estructuras de control CONDICIONES 2](#_Toc133914420)

[6.1. Estructuras de control BUCLES 3](#_Toc133914421)

[7. Sentencia SQL 3](#_Toc133914422)

[8. Excepciones 3](#_Toc133914423)

[9. Procedimientos y funciones 3](#_Toc133914424)

[10. Cursores 4](#_Toc133914425)

# Introducción

PLSQL es un lenguaje procedimental y orientado a objetos, se ejecutan en el servidor y la estructura básica es un bloque

# Bloques anónimos

Los bloques anónimos, no tienen nombre y se ejecutan en el servidor, la sintaxis es :

* **Declare** //Aquí declaramos las variables que usaremos. Podemos usar todos los datos de Oracle más el bolean. Las variables no inicializadas equivalen a NULL.
* **Begin** //Contiene las partes ejecutables.

dbms\_output.put\_line(‘’); --usamos esta estructura para imprimir por pantalla

* end; //Finalizacion de ejecución

# Tipos de datos

Dispone de los mismos tipos de datos en SQL, pero añadiendo uno nuevo, que es el boolean

# Variables

Se declaran en la primera parte del bloque (declare), su sintaxis es

* v\_edad integer not null := 0; --valor inicial obligatorio

Si no se inicializan las variables, su valor es null

* **%type**: Se declara una variable de mismo tipo que otra
* **%Rowtype**: Se declara una variable de tipo registro en la tabla
* **Constant**: Representa un valor que no cambia

# Operadores

|  |  |
| --- | --- |
| **Operadores** | |
| Asignación | := |
| Concatenación | || |
| Comparación | =, !=, <, >, <=, >= |
| Aritmeticos | +, -, \*, / |
| Logicos | AND, OR, NOT |

# Estructuras de control CONDICIONES

* Condición if

if condicion then

else

end if;

* Condición case

case <expresion>

when valor1 then

when valor2 then

# Estructuras de control BUCLES

* Loop Ejecuta el código hasta que se cumpla, su sintaxis es

LOOP  
instrucciones;  
EXIT [WHEN <condición>];  
END LOOP;

* For loop
* While loop Ejecuta el código hasta que se cupla, su sintaxis es

WHILE <condición>  
LOOP  
instrucciones;  
END LOOP

# Sentencia SQL

Se puede ejecutar cualquier orden DML (INSERT, DELETE UPDATE),si se ejecuta select, no se muestra en la terminal , sino que se queda en memoria llamado cursor, el formato básico es

**SELECT <columna/s> INTO <variable/s> FROM <tabla> [ WHERE ... ];**

# Excepciones

Las excepciones sirven para tratar los errores, algunos de estos errores, son siguientes

* NO\_DATA\_FOUND es cuando una sentencia no devuelve ningún valor
* TOO\_MANY\_VALUES Es cuando una consulta, devuelve mas de una fila

La palabra reservada **exception** se encarga de gestionar las excepciones con **when**

# Procedimientos y funciones

Un **procedimiento** no devuelve ningún valor por parámetro, tienen un nombre, para crear un procedimiento, se hace de la siguiente forma:

CREATE [OR REPLACE] PROCEDURE <nombre\_procedimiento> [(param1 tipo1, param2 tipo2, ...)]  
IS|AS  
[<declaraciones>;]  
BEGIN  
<instrucciones>;  
[EXCEPTION  
<excepciones>;]  
END [<nombre\_procedimiento>];

Una función si que devueleve un valor como parámetro y se hace de la siguiente forma :

CREATE [OR REPLACE] FUNCTION <nombre\_función> [(param1 tipo1, param2 tipo2, ...)]  
RETURN <tipo\_valor\_devuelto>  
IS|AS  
[<declaraciones>;]  
BEGIN  
<instrucciones>;  
RETURN <expresión>;  
[EXCEPTION  
<excepciones>;]  
END [<nombre\_función>]

# Cursores

Un cursor es una zona de memoria para analizar e interpretar cualquier sentencia SQL, solamente puede devolver una fila (Normalmente, si una consulta devuelve una fila, se utiliza un cursor explicito puede haber 2 tipos de cursores

* **Cursores explícitos** Lo genera el usuario, se trabajan en las consultas que devuelven una fila, las operaciones básicas con

1. Declarar el cursor

**CURSOR <nombre\_cursor> IS <sentencia SELECT>**

1. Apertura del cursor en la zona de instrucciones:

**OPEN <nombre\_cursor>;**

1. Recogida de información almacenada en el cursor:

**FETCH <nombre\_cursor> INTO {<variable>l<lista\_variables>};**

1. Cierre del cursor:

**CLOSE <nombre\_cursor>**

La instrucción **open** ejecuta automáticamente la sentencia select

FETCH recupera una fila y el cursor avanza automáticamente a la fila siguiente