# 1.- GENERALIDADES DEL LENGUAJE PL/SQL

1.1. Introducción al lenguaje PL/SQL.

1.3. Estructura de un bloque anónimo
1.4. Uso de variables en PL/SQL.
1.5. Tipos de variable
1.6. Sintaxis y directrices de bloque
1.7. Código de comentarios
1.8. Operadores
1.9. Funciones
1.10. Bloques anidados y ámbito de las variables
1.11. Directrices de programación

## 1.1. - INTRODUCCIÓN AL LENGUAJE PL/SQL

El PL/SQL es un lenguaje de bases de datos para realizar programas que incluyan sentencias de SQL. Estos programas se denominan bloques y se clasifican en:

-**Anónimos**: No suelen ser reutilizables. No llevan ningún tipo de nombre. Son semejantes a cualquier programa de 3ª Generación.

-**Subprogramas o bloques con nombre**: Suelen tener parámetros y pueden ser invocados por otro bloque. Existen 2 tipos de subprogramas:

- **Funciones**: Se utilizan para calcular un valor.
- **Procedimientos**: Ejecutan una serie de acciones.

Los beneficios que presentan los subprogramas son los siguientes:

- Mejorar el mantenimiento.
- Mejorar la seguridad e integridad de los datos.
- Mejorar el rendimiento.

## 1.3.- ESTRUCTURA DE UN BLOQUE ANÓNIMO

La estructura de un bloque anónimo de PL es la siguiente:

[DECLARE] – Opcional

- Variables, cursores y excepciones definidas por el usuario.

BEGIN – Obligatorio

- Sentencias SQL.
- Sentencias de control PL/SQL.

[EXCEPTION] – Opcional

- Acciones a realizar cuando se producen errores.

END; - Obligatorio

/

Estos bloques se suelen almacenar en scripts, al igual que los informes, para luego ejecutarlos. Un bloque PL se puede dividir en bloques lógicos o declarativos (declare), ejecutables (begin) y de excepciones (exception):

***1º Sección Declarativa***

Contiene todas las variables, constantes, y excepciones definidas por el usuario a las que hace referencia en las secciones ejecutables y excepciones.

***2º Sección Ejecutable***

Contiene sentencias SQL para manipular datos de la BD y sentencias PL/SQL para manipular datos del bloque.

***3º Sección para Gestión de Excepciones***

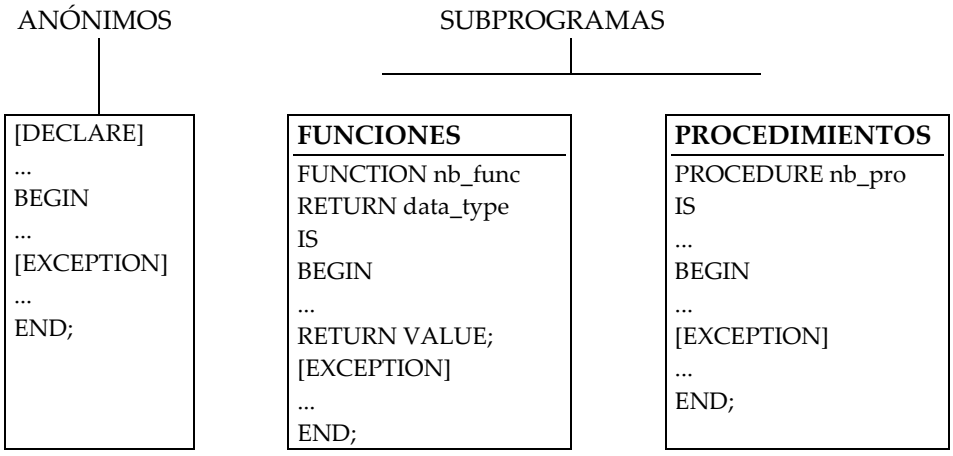
Especifica las acciones que se han de ejecutar cuando aparecen errores y condiciones anormales de la sección ejecutable.

La ejecución de sentencias y bloques desde SQL*PLUS se hará de la siguiente forma:

- Colocar `;` al final de una sentencia SQL*PLUS o de una sentencia de control PL/SQL.
- Colocar `/` al ejecutar un bloque anónimo en el buffer.

- Colocar **.** para cerrar el buffer de SQL*PLUS.

Tipos de Bloques PL/SQL.




- **Bloques anónimos**: son bloques sin nombrar. Son declarados en una aplicación en la que se van a ejecutar y se transfieren al motor PL/SQL para que se ejecuten al momento.
- **Subprogramas**: son bloques PL/SQL especificados que pueden tomar parámetros y pueden ser invocados. Puede declararlos como procedimientos o como funciones. Generalmente se usan los procedimientos para ejecutar una acción y una función para calcular un valor.

## 1.4.- USO DE VARIABLES EN PL/SQL

Las variables se utilizan para:

- Almacenamiento de forma temporal de datos.
- Manipulación de valores almacenados.
- Reusabilidad.
- Fácil mantenimiento.

La gestión de variables en PL/SQL presentan los siguientes aspectos:

- Las variables se declaran e inicializan dentro de la sección de declaración.
- Asignar nuevos valores a las variables dentro de la sección de códigos.
- Pasar valores a los bloques PL/SQL a través de parámetros.
- Ver los resultados a través de las variables de salida.

Las variables se clasifican en:

❖ **Variables PL/SQL:**

- **Escalares**: Contienen un valor único. Los principales tipos de datos son los que corresponden a tipos de columnas en las tablas del servidor Oracle8; PL/SQL también soporta variables booleanas.

- **Compuestas:** Hacen referencia a tipos de datos compuestos, como los registros y las tablas.
- **Referenciada:** o punteros, que designan otros artículos del programa.

❖ **Variables no PL/SQL:** Su misión es transferir información desde el puesto al servidor.

- Enlace o Bind.
- Host.

Todas ellas tienen un tipo de datos que se especifica en un formato de almacenamiento, restricciones, y rango de valores.

Sintaxis:

```
Nb_variable [CONSTANT] data_type [NOT NULL]
[:= expr | DEFAULT expr ];
```

Dónde:

CONSTANT: restringe la variable para que no pueda cambiar de valor. Las constantes deben inicializarse.

DATA_TYPE: Es un tipo de datos escalar, compuesto o LOB.

NOT NULL: restringe la variable. La obliga a contener algún valor. Las NOT NULL también se deben inicializar.

EXPR: Cualquier expresión PL/SQL, pudiendo ser un literal, otra variable, etc.

Ejemplos:

```
V_HIREDATE NUMBER(2) NOT NULL:=10;
V_HIREDATE DATE;
V_LOC VARCHAR2(30):='ATLANTA';
C_COMM CONSTANT NUMBER(4):=10;
```

**Recomendaciones a seguir**

1 – Intentar poner los identificadores: V variable, C constante, G global.
2 – Inicializar las constantes y las variables NOT NULL.
3 – Inicializar los identificadores con := ó DEFAULT.
4 – Declarar como máximo un identificador por línea.

**Reglas para los nombres**

- Dos variables pueden tener el mismo nombre si están en dos bloques diferentes.
- El nombre de la variable (identificador) no debe ser el mismo que el de una columna de una tabla utilizada en el bloque.
- Se puede asignar cualquier valor a una variable simplemente nb_var:=expresion;
- Se pueden usar las funciones TO_NUMBER, TO_DATE y TO_CHAR.

**Funciones para imprimir y ver la ejecución de bloques**

Para poder imprimir frases en pantalla no existen funciones específicas, pero se debe usar:

**DBMS_OUTPUT.PUT_LINE(<expresion>);**

Para poder ver la ejecución:

**SET SERVEROUTPUT ON**

Ejemplo 2:

```
DECLARE
        V_TEXTO VARCHAR2(20):='HOLA MUNDO';
BEGIN
        DBMS_OUTPUT.PUT_LINE(V_TEXTO);
END;
/
```

**EL ATRIBUTO %TYPE:**

Se utiliza la mayoría de las veces cuando el valor almacenado en la variable se deriva de una columna de la tabla de la BD. Además se utiliza si se hace referencia a una variable declarada previamente.

Ejemplo

v_ename emp.ename%TYPE;
v_ename adopta el tipo de dato de emp.ename.

v_balance number(7,2);
v_balance se le asigna el tipo number de 7,2

v_min_val v_balance%TYPE;
v_min_val adopta el tipo number(7,2) como v_balance

## 1.5.- TIPOS DE VARIABLES

- **VARIABLES PL/SQL**

**Variables escalares**:

Las principales son:

| Tipos | Descripción |
|---|---|
| VARCHAR2(n) | Para datos tipo carácter de longitud 3270 bytes. |
| NUMBER[(p,s)] | Enteros y decimales. |
| DATE | Fechas y horas. |
| CHAR(n) | Cadenas de longitud fija. |
| LONG | Datos carácter de longitud variable. |
| LONG RAW | Datos binarios y cadenas. |
| BOOLEAN | Utilizada para cálculos lógicos, admite TRUE, FALSE y NULL. Sólo pueden estar conectadas por los operadores lógicos AND, NOT, OR y las expresiones aritméticas de carácter y fecha que pueden devolver un valor booleano. |
| BINARY_INTEGER | Tipo de base de enteros entre –2147483647 y 2147483647. |

**Variables de datos compuestos:**

Existen fundamentalmente 2 tipos de datos:

- **Registros**: trata datos relacionados pero no iguales como una unidad lógica.
- **Tablas**: hacen referencia a colecciones de datos que permiten ser manipulados.

- **VARIABLES NO PL/SQL**

**Variables de enlace ó bind:**

Es una variable que declara cualquier usuario y es utilizada después para transferir valores bien sean numéricos o de caracteres dentro o fuera de uno o más programas PL.

**Host:**

Tienen el mismo funcionamiento a las de enlace con la diferencia que a la hora de definirlas hay que precederlas de 2 puntos (:HOST).

## 1.6 – SINTAXIS Y DIRECTRICES DE BLOQUE

- Una sentencia puede terminar en una línea y continuar en la otra si ponemos al final de toda la sentencia un punto y coma;

- Los identificadores, literales, comentarios, etc. pueden separarse por uno o más espacios.
- Los identificadores sirven para dar nombre a las unidades y artículos de los programas PL/SQL. Pueden contener hasta 30 caracteres, no pueden contener palabras reservadas, deben comenzar por una letra y el nombre de un identificador no es aconsejable que coincida con el nombre de una columna de una tabla.

- Sintaxis y directrices de los literales: Los números pueden ser simples o ir dentro de una notación científica. Han de ir entre comillas las fechas y las cadenas.

## 1.7 – CÓDIGO DE COMENTARIOS

Si el comentario ocupa una línea se comienza poniendo -- comentario. Si ocupa varias líneas ponemos /* al abrirlo y para cerrarlo */

## 1.8 – OPERADORES

| **Operadores por orden de operación.** | **Operación** |
|---|---|
| ** , NOT | Exponenciación, negación lógica. |
| + , - | Identidad y negación. |
| * , / | Multiplicación y división. |
| + , - , \|\| | Suma, resta y concatenación. |
| = , != , < , >, <= , >= , IS NULL, LIKE, BETWEEN, IN | Comparación. |
| AND | Conjunción. |
| OR | Inclusión. |

## 1.9 - FUNCIONES

**ARITMÉTICAS**: Devuelven un número:

| **Funciones** | **Descripción** |
|---|---|
| ABS(n) | Devuelve el valor absoluto de un número. |
| CEIL(n) | Devuelve el entero más pequeño o igual al número. |
| COS(n) | Coseno de n. |
| EXP(n) | Número e elevado a n. |
| FLOOR(n) | Entero más grande o igual a n. |
| LN(n) | Devuelve el logaritmo neperiano de n. |
| LOG(n1,n2) | Logaritmo en base n1 de n2. |
| MOD(n1,n2) | Devuelve el resto de dividir n1 entre n2. |
| POWER(n1,n2) | Devuelve n1 elevado a n2. |
| ROUND(n1,n2) | Redondea n1 con n2 decimales. |
| SVGN(n) | Devuelve –1 si el número es menor de 0, 0 si es igual a 0 y 1 si es |

| | |
|---|---|
| | mayor de 0. |
| SIN(n) | Seno de n. |
| SQRT(n) | Raíz cuadrada de n, que ha de ser siempre mayor de 0. |
| TAN(n) | Tangente de n. |
| TRUNC(n1,n2) | Trunca n1 con n2 decimales. |

**CARÁCTER:** Devuelven cadenas de caracteres.

| **Funciones** | **Descripción** |
|---|---|
| ASCII(C) | Devuelve el ASCII del carácter. |
| CHR(n) | Devuelve el carácter correspondiente al número n. |
| CONCAT (c1, c2) | Concatena 2 cadenas. |
| INITCAP(c) | Transforma la primera letra a mayúscula. |
| LENGHT(c) | Longitud de la cadena. |
| LOWER(c) | Transforma a minúsculas. |
| LPAD (c1, n, c2) | Justifica a la derecha el valor del carácter c1 n posiciones y los blancos los reemplaza por el carácter c2. |
| LTRIM(c1, c2) | Suprime caracteres a la izquierda hasta el primer carácter que no está en c2. Si se omite c2 lo pone todo en blanco. |
| REPLACE(c1,c2,c3) | Devuelve c1 con la cadena c2, reemplazada por c3 y así sucesivamente. |
| RPAD(c1,n,c2) | Iguala a la derecha. |
| RTRIM(c1,c2) | Igual que LTRIM pero por la derecha. |
| SUBSTR(c1,m,n) | Substrae de la cadena c1, y desde el carácter m, n caracteres. |
| TRANSLATE(c1,c2,c3) | Devuelve la cadena c1 con cada carácter de c2 que contenga reemplazado por el carácter de c3 con el que se corresponda. |
| UPPER(c) | Convierte c a mayúsculas. |

**CONVERSIÓN**

| **Funciones** | **Descripción** |
|---|---|
| TO_CHAR(c1,'fmt') | Convierte un número o fecha a cadena de caracteres. |
| TO_DATE(c1,'fmt') | Convierte una cadena de caracteres a un número. |
| TO_NUMBER(c1,'fmt') | Convierte una cadena de caracteres a una fecha. |

**FECHA**

| **Funciones** | **Descripción** |
|---|---|
| ADD_MONTH(fecha,n) | Agrega meses a una fecha. |
| LAST_DAY(fecha) | Último día del mes. |
| MONTHS_BETWEEN(fecha1,fecha2) | Número de meses entre dos fechas. |
| NEXT_DAY(fecha, 'caracter') | Próximo día de la fecha especificada. |

**DIVERSAS**

| **Funciones** | **Descripción** |
|---|---|
| DUMP(expr) | Devuelve la expresión en el formato interno de Oracle. |
| GREATEST(lista de valores) | Da como resultado el valor más grande de la lista. |
| LEAST (lista de valores) | Da como resultado el valor más pequeño de la lista. |
| NVL(expr1, expr2) | Reemplaza los nulos por otro valor. |
| USER | Devuelve al usuario conectado. |
| USERENV(opción) | Devuelve información sobre el usuario. Las opciones pueden ser:<br>• SESSIONID: Se identifica la sesión.<br>• TERMINAL: Identificador del terminal.<br>• LANGUAGE: Lenguaje activo. |
| DECODE | Realiza la función de IF/THEN/ELSE. |

## 1.10 - BLOQUES ANIDADOS

También se pueden anidar bloques de forma que las variables tengan su propio ámbito de ejecución:

```
DECLARE
        X BINARY_INTEGER;
BEGIN
        DECLARE
                Y BINARY_INTEGER;
        BEGIN
                ..............
        EXCEPTION
                .........
        END;
EXCEPTION
        .......
END;
```

Las sentencias pueden ser anidadas cuando el bloque PL/SQL lo requiera. Un bloque anidado se convierte en una sentencia. La sección de excepciones también puede contener bloques, pero generalmente el lugar donde se suele trabajar es en la zona ejecutable. El ámbito de un objeto es la región de programa que hace referencia a dicho objeto.

Las variables tienen como ámbito su propio bloque. En este caso, la variable Y sólo será reconocida por el segundo bloque, y una vez finalizado este el bloque principal no la reconocerá, por el contrario, la variable X tiene como ámbito todo el bloque, ya que tiene vigencia en su propio bloque y en el bloque anidado, ya que este pertenece al bloque principal.

## 1.11 – DIRECTRICES DE PROGRAMACIÓN

Para hacer un bloque PL/SQL Oracle marca las siguientes directrices:

- Documentar el código con comentarios.

- Utilizar mayúsculas para sentencias SQL*PLUS, para palabras claves de PL/SQL y para los tipos de datos y minúsculas para identificadores, parámetros, columnas y tablas de la BD.

- Desarrollar convenciones de nomenclatura para los identificadores y otros objetos, evitando ambigüedades en los nombres. Así pues la variable se define como V_, la constante C_, el cursor Nombre_cursor, la excepción como E_, etc.

- Sangrar los bloques para obtener mayor claridad en el código.