LENGUAJE PL/SQL

Presentado por: Daniel Alejandro Guerrero Suárez

LENGUAJES DE PROCEDIMIENTO

SQL es un lenguaje de consulta para operar con conjuntos. Es casi estandarizado y utilizado por la mayoría de los sistemas de gestión de bases de datos relacionales: SQL Server, Oracle, MySQL, PostgreSQL, DB2, Informix, etc.

Los lenguajes de procedimiento están diseñados para ampliar las capacidades de SQL siendo capaces debe integrarse con SQL. Ejemplos:

- PL/SQL (Procedural Language / SQL) es un lenguaje de procedimiento patentado y utilizado por Oracle
- PL/pgSQL es un lenguaje de procedimiento utilizado por PostgreSQL
- T-SQL (Transact-SQL) es un lenguaje de procedimiento patentado y utilizado por Microsoft en SQL Server.

DEFINICION DE PL/SQL

SQL es un lenguaje de consulta para los sistemas de bases de datos relaciónales, pero que no posee la potencia de los lenguajes de programación.

Sql es un lenguaje de consulta, no un lenguaje de programación.

Cuando se desea realizar una aplicación completa para el manejo de una base de datos relacional, resulta necesario utilizar alguna herramienta que soporte la capacidad de consulta del SQL y la versatilidad de los lenguajes de programación tradicionales.

PL/SQL es el lenguaje de programación que proporciona Oracle para extender el SQL estándar con otro tipo de instrucciones.

LENGUAJE PL/SQL

OPERADORES

Tipo de operador	Operadores
Operador de asignación	:= (dos puntos + igual)
Operadores aritméticos	+ (suma) - (resta) * (multiplicación) / (división) ** (exponente)
Operadores relacionales o de comparación	= (igual a) <> (distinto de) < (menor que) > (mayor que) >= (mayor o igual a) <= (menor o igual a)
Operadores lógicos	
Operador de concatenación	

LENGUAJE PL/SQL

PARAMETROS (PROCEDIMIENTO Y FUNCIONES)

Los parámetros se declaran con los tipos de datos pero sin la longitud o precisión. Eso significa que un parámetro se puede declarar como VARCHAR2 pero no será declarado con un componente de la longitud (VARCHAR2 (30) no sería válido).

Los parámetros pueden tener un valor predeterminado. Se utilizar al operador de asignación (: =) o la palabra clave DEFAULT. Cuando un parámetro tiene un valor predeterminado, no es necesario incluir ese parámetro en la llamada.

Declaraciones del parámetro :

```
parameter_1 IN VARCHAR2 := 'ABC',
parameter_2 IN VARCHAR2 DEFAULT 'ABC',
parameter_3 IN OUT NUMBER
```

LENGUAJE PL/SQL

VARIABLES

Las variables son nombres para procesar los elementos de los datos. Los Operadores := y DEFAULT son lo mismo. Si ponemos NOT NULL es obligatorio inicializar la variable.

Sintaxis:
Nombre_variable tipo [NOT NULL] [:= valor | DEFAULT valor]

Ejemplos:
num_dep number(2) NOT NULL :=20
nom_emple varchar2(15) default ‘Pedro’

LENGUAJE PL/SQL

VARIABLES

%TYPE se usa para declarar una variable que tendrá el mismo tipo que una columna de una tabla. Ejemplo:

```
DECLARE
        v_EmpName emp.ename%TYPE;
BEGIN
        SELECT ename INTO v_EmpName FROM emp WHERE emp_id = 1;
        DBMS_OUTPUT.PUT_LINE('Name = '|| v_EmpName);
END;
/
```

%ROWTYPE se usa para declarar un registro con los mismos tipos de la tabla, vista o cursor de la base de datos. Ejemplo:

```
DECLARE
        v_EmpName emp%ROWTYPE;
BEGIN
        SELECT * INTO v_EmpName FROM emp WHERE  emp_id = 1;
        DBMS_OUTPUT.PUT_LINE('Name = '|| v_EmpName);
END;
/
```

LENGUAJE PL/SQL

CONSTANTES

Las constantes son como las variables pero no puede modificarse su valor. Se declaran de la siguiente manera:

nombre_constante CONSTANT tipo_de_dato := valor

Por ejemplo, el IVA es un valor fijo, y para declararlo se hace de la siguiente manera:

```
Imp_iva constant number(2,2) := 16,5
pi constant number(2,2 ):= 3.1416;
```

LENGUAJE PL/SQL

BUCLES

Un bucle o ciclo, es una sentencia que se realiza repetidas veces un código, hasta que la condición asignada a dicho bucle deje de cumplirse.

Generalmente, un bucle es utilizado para hacer una acción repetida sin tener que escribir varias veces el mismo código, lo que ahorra tiempo, deja el código más claro y facilita su modificación en el futuro.

En PL/SQL se tiene los siguientes iteradores o bucles:

- LOOP
- WHILE
- FOR

Las sentencias de dentro del bucle se ejecutarán durante un número indefinido de vueltas, hasta que aparezca la instrucción EXIT.

Otra opción es incluir la estructura EXIT WHEN condición, se terminará el bucle cuando la condición se cumpla.

LENGUAJE PL/SQL

BUCLES (LOOP)

El bucle LOOP, se repite tantas veces como sea necesario hasta que se fuerza su salida con la instrucción EXIT. Su sintaxis es la siguiente:

```
LOOP
        --Sentencias
EXIT WHEN condición;
        --Sentencias
END LOOP;
```

```
LOOP
        --Sentencias
IF (expresion) THEN
        --Sentencias
EXIT;
END IF;
END LOOP;
```

Ejemplo:

```
DECLARE
        total NUMBER(9) := 0;
         contador NUMBER(6) := 0;
BEGIN
        LOOP
                 contador := contador + 1;
                total := total + contador * contador;
        EXIT WHEN total >= 20;
        END LOOP;
    DBMS_OUTPUT.PUT_LINE('contador: ' || TO_CHAR(contador) || ' Total: ' || TO_CHAR(total));
END;
/
```

LENGUAJE PL/SQL

BUCLES (WHILE)

El bucle WHILE, se repite mientras que se cumpla la expresion.

```
WHILE (expresion) LOOP
  -- Instrucciones
END LOOP;
```

Ejemplo:

```
DECLARE
  Num NUMBER := 1;
  Num_cubo NUMBER;
BEGIN
  WHILE Num <= 10 LOOP
    Num_cubo := Num**3;
    DBMS_OUTPUT.PUT_LINE('Numero:'||TO_CHAR(Num)||' Cubo: '||TO_CHAR(Num_cubo));
    Num := Num + 1;
  END LOOP;
END;
/
```

LENGUAJE PL/SQL

BUCLES (FOR)

El bucle FOR, se repite tanta veces como le indiquemos en los identificadores inicio y final.

```
FOR contador IN [REVERSE] limite_inferior..limite_superior LOOP
    --sentencias
END LOOP;
```

En el caso de especificar REVERSE el bucle se recorre en sentido inverso.

```
BEGIN
 FOR loop_contador IN 1..10 LOOP
       DBMS_OUTPUT.PUT_LINE('Numero: ' || TO_CHAR(loop_contador) || '   Cuadrado:     '     ||
TO_CHAR(loop_contador**2));
 END LOOP;
END;
/
```

LENGUAJE PL/SQL

SENTENCIA GO TO

La sentencia GOTO desvía el flujo de ejecución a una determinada etiqueta. En PL/SQL las etiquetas se indican del siguiente modo: << etiqueta >>

```
DECLARE
  Num NUMBER;
BEGIN
 Num :=1 ;
  IF (Num = 1) THEN
    GOTO paso2;
  END IF;
<<paso1>>
        dbms_output.put_line('Ejecucion de paso 1');
<<paso2>>
        dbms_output.put_line('Ejecucion de paso 2');
END;
/
```

LENGUAJE PL/SQL

SUBPROGRAMAS

Con PL/SQL se programan unidades o bloques de código (Subprogramas) de la base de datos en ORACLE, están son:

- Procedimientos almacenados
- Funciones
- Scripts (Bloque anónimos)
- Paquetes
- Triggers

Pero además PL/SQL permite realizar programas sobre las siguientes herramientas de ORACLE:

- Oracle Forms (Ventanas de Formularios tipo Windows)
- Oracle Reports (Generación de Reportes o informes)
- Oracle Graphics (Interacción con gráficos estadísticos)
- Oracle Aplication Server (desarrollo, integración e implementación de aplicaciones)

LENGUAJE PL/SQL

SUBPROGRAMAS

Los Subprogramas en Oracle están compuestos por:

- **Zona de Declaraciones:** Una parte opcional en la que se definen de variables y cursores.
- **Zona de Instrucciones:** Es una parte Obligatoria que es ejecutable y esta compuesta por sentencias SQL y PLSQL.
- **Zona de tratamiento de excepciones:** Parte opcional enfocada a manejar las excepciones y errores ocurridos durante la ejecución.

```
CREATE {OR REPLACE} { PROCEDURE / FUNCTION } nombre_subp ( param1 [IN | OUT | IN OUT] tipo,... )
 {DECLARE} { IS / AS }
 -- Declaración de variables locales
BEGIN
 -- Instrucciones de ejecución
[EXCEPTION]
-- Instrucciones de excepción
END;
```

LENGUAJE PL/SQL

SUBPROGRAMAS (FUNCIONES)

Una función es un subprograma que devuelve un valor (return) y no recibe parámetros de salida. Las funciones se compilan y almacenan en la base de Datos. La sintaxis para construir funciones es la siguiente:

```
CREATE [OR REPLACE]
FUNCTION <fn_name> [(<param1> IN <type>, <param2> IN <type>, ...)]
RETURN <return_type>
        IS
                result <return_type>;
        BEGIN
                -- Sentencias de Consultas u operaciones
                return(result);
        [EXCEPTION]
                -- Sentencias control de excepcion
END [<fn_name>];
/
```

El uso de OR REPLACE permite sobrescribir una función existente. Si se omite, y la función existe, se producirá un error en la compilación.

LENGUAJE PL/SQL

SUBPROGRAMAS (FUNCIONES)

```
CREATE OR REPLACE FUNCTION MULTIPLICAR(A NUMBER, B NUMBER) RETURN NUMBER
 IS
       RESULTADO NUMBER(30);
 BEGIN
        RESULTADO := A * B;
       return(RESULTADO);
  END MULTIPLICAR;
```

Llamar a la Función y ejecutarla:

```
select MULTIPLICAR (3,5)  from dual;
select MULTIPLICAR (3,5) as Multiplicacion from dual;
```

Para mostrar Resultados en pantalla:

```
set serveroutput on size unlimited;
```

LENGUAJE PL/SQL

SUBPROGRAMAS (PROCEDIMIENTOS)

Un procedimiento es un subprograma que ejecuta una acción especifica y que no devuelve ningún valor(no Return) pero admite parámetros de salida (out). Un procedimiento tiene un nombre, un conjunto de parámetros (opcional) y un bloque de código. La sintaxis de un procedimiento almacenado es:

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

LENGUAJE PL/SQL

SUBPROGRAMAS (PROCEDIMIENTOS)

```
CREATE OR REPLACE PROCEDURE SUMAR(A NUMBER,B NUMBER) IS
        RESULTADO NUMBER(10);
BEGIN
         RESULTADO := A + B;
        DBMS_OUTPUT.PUT_LINE('EL RESULTADO DE LA SUMA ES : '||RESULTADO);
END SUMAR;
```

Llamar al procedimiento y ejecutarlo:

```
        Execute SUMAR(10,25);
```

Para mostrar Resultados en pantalla:

```
        set serveroutput on size unlimited;
```

LENGUAJE PL/SQL

SUBPROGRAMAS (PROCEDIMIENTOS)

```
CREATE OR REPLACE PROCEDURE RESTAR(A NUMBER,B NUMBER) IS
        RESULTADO NUMBER(10);
        Valor NUMBER(30);
BEGIN
        RESULTADO := A - B;
         DBMS_OUTPUT.PUT_LINE('EL RESULTADO DE LA RESTA ES : '||RESULTADO);
END RESTAR;
```

Llamar al procedimiento y ejecutarlo:

```
        Execute Restar(10,25);
```

LENGUAJE PL/SQL

SUBPROGRAMAS (BLOQUES ANONIMOS)

Un Bloque anónimo (Sin nombre) es un subprograma que ejecuta una acción especifica. Siempre comienza con DECLARE o directamente con BEGIN. Su sintaxis es:

```
[DECLARE]
        --Define objetos PL/SQL que serán utilizados dentro del mismo bloque
BEGIN
        --Sentencias Ejecutables
[EXCEPTION]
        --Qué hacer si la acción ejecutada causa error
END;
/
```

Un bloque anónimo no se guarda en la Base de Datos.

LENGUAJE PL/SQL

SUBPROGRAMAS (PAQUETES)

Se usan para agrupar procedimientos y funciones. Un paquete es una estructura que agrupa objetos de PL/SQL compilados(procedures, funciones, variables, tipos ...) en la base de datos. Esto nos permite agrupar la funcionalidad de los procesos en programas.

Lo primero que debemos tener en cuenta es que los paquetes están formados por dos partes:

- La especificación: es la interfaz con las aplicaciones. En ella es posible declarar los tipos, variables, constantes, excepciones, cursores y subprogramas disponibles para su uso posterior desde fuera del paquete. En la especificación del paquete sólo se declaran los objetos (procedures, funciones, variables ...), no se implementa el código.

- El cuerpo: El cuerpo del paquete debe implementar lo que se declaró inicialmente en la especificación. Es el donde debemos escribir el código de los subprogramas. En el cuerpo de un package podemos declarar nuevos subprogramas y tipos, pero estos serán privados para el propio package.

La especificación del un paquete y su cuerpo se crean por separado.

LENGUAJE PL/SQL

SUBPROGRAMAS (TRIGGERS)

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. La sintaxis para crear un trigger es la siguiente:

```
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>;
/
```

LENGUAJE PL/SQL

EXCEPCIONES

En PL/SQL una advertencia o condición de error es llamada una excepción. Las excepciones se controlan dentro de su propio bloque.

```
DECLARE
        -- Declaraciones
BEGIN
        -- Ejecucion
EXCEPTION
        WHEN NO_DATA_FOUND THEN
        -- Se ejecuta cuando ocurre una excepcion de tipo NO_DATA_FOUND
        WHEN ZERO_DIVIDE THEN
        -- Se ejecuta cuando ocurre una excepcion de tipo ZERO_DIVIDE
        WHEN OTHERS THEN
        -- Se ejecuta cuando ocurre una excepcion de un tipo no tratado
        -- en los bloques anteriores
END;
```

Cuando ocurre un error, se ejecuta el bloque **EXCEPTION**, transfiriéndose el control a las sentencias del bloque. Una vez finalizada la ejecución del bloque de **EXCEPTION** no se continua ejecutando el bloque anterior.

LENGUAJE PL/SQL

EXCEPCIONES

Excepcion	Se ejecuta ...	SQLCODE