PL/SQL - Oracle

PL/SQL (Procedural Language/SQL) es una extensión de SQL, que agrega ciertas construcciones propias de lenguajes procedimentales, obteniendose como resultado un lenguaje estructural mas poderoso que SQL.

La unidad de programación utilizada por PL/SQL es el bloque.

Todos los programas de PL/SQL están conformados por bloques. Tipicamente, cada bloque lleva a cabo una acción lógica en el programa. Un bloque tendrá siempre la siguiente estructura:

```
DECLARE

//Sección declarativa: variables, tipos, y subprogramas
//de uso local

BEGIN

//Sección ejecutable: las instrucciones procedimentales, y de SQL
//aparecen aquí. Es la unica sección obligatoria en el bloque.

EXCEPTION

//Sección de manejo de excepciones. Las rutinas de manejo de errores
//aparecen aqui

END;
```

Solo se requiere que aparezca la sección ejecutable. Lo demas es opcional. Las unicas instrucciones SQL permitidas en un bloque PL/SQL son INSERT, UPDATE, DELETE y SELECT, ademas de algunas instrucciones para manipulación de datos, e instrucciones para control de transacciones. Otras instrucciones de SQL como DROP, CREATE o ALTER no son permitidas. Se permite el uso de comentarios estilo C (/* . . .*/). PL/SQL no es case sensitive por lo que no hay distinción entre nombres con mayusculas y minusculas.

En la sección de declaraciones, se indican las variables que serán usadas dentro del bloque y sus tipos. Por ejemplo:

```
DECLARE
myBeer VARCHAR(20);
price NUMBER(6,2);
```

En algunos casos, es posible que se desee que el tipo de una variable coincida con el tipo usado para una columna de una tabla determinada, en esos casos se puede usar la construcción:

```
DECLARE
myBeer Beers.name%TYPE;
```

Con lo cual se logra que la variable myBeer tenga el mismo tipo que la columna name de la tabla Beers.

Tambien es posible inicializar las variables, mediante el operador :=. Ademas, mediante el uso del mismo operador es posible hacer asignaciones en el cuerpo del programa. Por ejemplo:

```
DECLARE
price NUMBER := 300;
```

```
BEGIN
price := price + 150;
END;
.
run
```

La ejecución de este bloque no tendrá ningun efecto, ya que no se están haciendo cambios sobre la base de datos.

Ademas es posible usar sentencias condicionales y ciclos dentro de los bloques de PL/SQL. Una sentencia condicional tipica es de la forma:

```
IF (condicion)
THEN (lista de acciones)
ELSE (lista de acciones)
END IF;
```

Si se desea, se puede hacer el uso de varios casos de condición, mediante el uso de:

```
IF . . . THEN . . .
ELSIF . . . THEN . . .
ELSIF . . . THEN . . .
.
.
.
ELSE . . .
END IF;
```

En ambos casos, la clausula ELSE es opcional.

Si se desea crear un lazo, se puede usar la instrucción:

```
LOOP
lista_de_instrucciones
END LOOP;
```

Al menos alguna de las instrucciones debe ser:

```
EXIT WHEN condicion;
```

De esta manera, el lazo terminará cuando la condición sea verdadera. Además es posible utilizar la instrucción:

```
WHILE (condicion) LOOP
lista_de_instrucciones
END LOOP;
```

De esta forma, el ciclo solo se inicia si la condicion es verdadera en principio. Es posible que el programa nunca entre en el ciclo. Usando la instrucción LOOP se garantizaba que siempre se ejecutaría el cuerpo del ciclo al menos una vez. Por último, es posible usar ciclos que se ejecuten un numero predeterminado de veces, mediante el uso de la instrucción:

```
FOR i IN a..b LOOP
lista_de_instrucciones
END LOOP;
```

En este caso i es una variable de uso local, por lo que no es necesario que sea declarada, y puede ser usada dentro del lazo, mientras que a y b son constantes.

Procedimientos almacenados

Un procedimiento almacenado es un conjunto de instrucciones en PL/SQL, que pueden ser llamado usando el nombre que se le haya asignado.

La sintaxis para crear un procedimiento es la siguiente:

```
CREATE [OR REPLACE] PROCEDURE name [(param [IN|OUT|IN OUT|]
datatype) . . .]
[IS|AS] pl/sql_subprogram
```

El uso de OR REPLACE permite sobreescribir un procedimiento existente. Si se omite, y el procedimiento ya existe, se producirá un error. Los modificadores IN, OUT, IN OUT indican si el parametro es de entrada, salida o ambos.

A continuación se presenta un ejemplo de creación de un procedimiento:

```
SQL> CREATE PROCEDURE credit (acc_no IN NUMBER, amount IN
NUMBER)
1> AS BEGIN
2> UPDATE accounts
3> SET balance = balance + amount
4> WHERE account_id = acc_no;
5> END;
```

Este procedimiento actualizará la(s) tupla(s) con numero de cuenta igual al parámetro acc_no con un incremento de amount en el balance de dicha cuenta.

Si se desea eliminar (borrar) un procedimiento almacenado, se usa la instrucción:

```
SQL> DROP PROCEDURE name;
```

Manejo de cursores

El conjunto de filas resultantes de una consulta con la sentencia *SELECT*, como vimos anteriormente, puede estar compuesto por ninguna, una o varias filas, dependiendo de la condición que define la consulta. Para poder procesar individualmente cada fila de la consulta debemos definir un cursor (que es un área de trabajo de memoria) que contiene los datos de las filas de la tabla consultada por la sentencia *SELECT*.

Los pasos para el manejo de cursores son:

- Definir el cursor, especificando la lista de parámetros con sus correspondientes tipos de datos y estableciendo la consulta a realizar con la sentencia *SELECT*.
- Abrir el cursor para inicializarlo, siendo éste el momento en que se realiza la consulta.
- Leer una fila del cursor, pasando sus datos a las variables locales definidas a tal efecto.
- Repetir el proceso fila a fila hasta llegar a la última.
- Cerrar el cursor una vez que se terminó de procesar su última fila.

A continuación veremos un ejemplo de cursor con las siguientes características:

Objetivo: Consultar las ventas de una fecha dada ordenadas de mayor a menor.

Nombre: *CVENTAS*.

Parámetros: cfecha, variable que contiene la fecha a consultar.

Código de definición del cursor: Ver figura 1

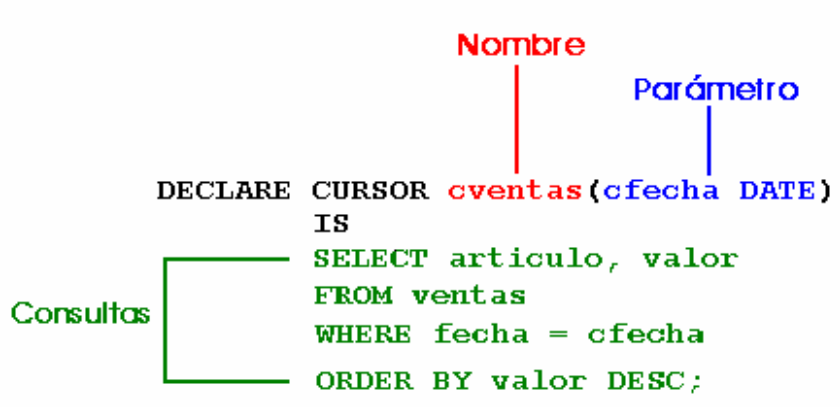

Figura 1: Código de definición de cursor

Con el procedimiento *VENTAS5* del fuente 5, mostraremos cómo usar el cursor *cventa* anteriormente definido, con el fin de registrar en la tabla *VENTAMAYOR* las 5 mayores ventas en una fecha dada.

```
/* --- Fuente 5 ----------------------------------------------------
--

PROCEDURE VENTAS5 (xfecha DATE) is
BEGIN

      DECLARE CURSOR cventas (cfecha DATE)
            IS SELECT articulo,valor
                  FROM ventas
                  WHERE fecha=cfecha
                  ORDER BY valor DESC;

            xarticulo   ventas.articulo%TYPE;

            xvalor      ventas.valor%TYPE;

      BEGIN

            OPEN cventas(xfecha);

            FOR i IN 1..5 LOOP
                  FETCH cventas INTO xarticulo,xvalor;
                  EXIT WHEN cventas%NOTFOUND;
                  INSERT INTO ventamayor VALUES
                              (xfecha,xarticulo,xvalor);
                  COMMIT;
            END LOOP;

            CLOSE cventas;
      END;
END;
```

Para llamar al procedimiento ventas5 en una fecha dada, se puede escribir, por ejemplo:

```
ventas5(to_date('15/11/95','DD/MM/YY')
```

o

```
ventas5(sysdate).
```

A continuación detallaremos las sentencias usadas en este procedimiento:

```
DECLARE cursor
```

Define el cursor, su consulta y la lista de parámetros que se pasan a la orden WHERE, es solo la declaración del cursor y no la realización de la consulta.

```
xarticulo ventas.articulo%TYPE;
```

Define la variable xarticulo igual a la columna articulo de la tabla ventas, que con el uso del atributo de variable %TYPE permite declarar una variable del mismo tipo que una columna de la tabla. No es necesario conocer cómo está definida esa columna en la tabla y, en caso que la definición de la columna sea modificada, automáticamente se cambia la variable xarticulo.

```
OPEN cventas(xfecha);
```

Realiza la consulta asociada al cursor, pasando el valor del parámetro y guardando sus resultados en un área de la memoria, desde la cual, posteriormente, se pueden leer estas filas.

```
FOR i IN 1..5 LOOP
```

Ciclo numérico de repetición para poder consultar las 5 primeras ventas devueltas por el cursor.

```
FETCH cventas INTO xarticulo,xvalor;
```

Lee la siguiente fila de datos del cursor cventas y pasa los datos de la consulta a las variables xarticulo y xvalor.

```
EXIT WHEN cventas%NOTFOUND;
```

Garantiza la salida del ciclo antes de las última repetición, en caso que para una fecha dada se hayan efectuado menos de 5 ventas, ya que en esta situación la consulta del cursor devuelve menos de 5 filas.

`%NOTFOUND` es un atributo de cursor que es verdadero cuando la última sentencia FETCH no devuelve ninguna fila.

```
INSERT INTO ventamayor
   VALUES(xfecha,xarticulo,xvalor);
```

Insertar en la tabla ventamayor los valores leídos desde el cursor.

```
COMMIT;
```

Actualización de la tabla ventamayor.

```
END LOOP;
```

Fin del ciclo.

```
CLOSE cventas;
```

Cierra el cursor, eliminado sus datos del área de memoria.