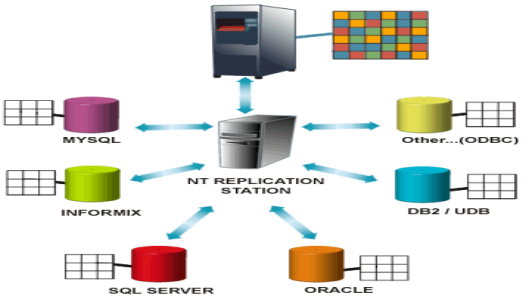
UNIVERSIDAD TECNOLOGICA DE PANAMA

FACULTAD DE INGENIERIA DE SISTEMAS

LICENCIATURA EN INGENIERIA DE SISTEMAS DE INFORMACION.

SISTEMAS DE BASE DE DATOS II
ORACLE PROGRAMACION PL/SQL

Funciones- Disparadores-Vistas–PL/SQL ORACLE



1

CONTENIDO

Capítulo V. Funciones disparadores y Vistas

- *Funciones*
- *Disparadores*
- *Vistas*

2

5.1 Funciones

Creacion de Funciones

- Una función es similar a un procedimiento. Ambos aceptan argumentos y estos pueden ser de cualquiera de los modos presentados.
- Ambos son formas diferentes de bloques PL/SQL, con sus secciones declarativas, ejecutable y de excepciones.
- Ambos pueden ser almacenados en la Base de Datos o ser declarados dentro de un bloque(procedimientos y funciones que no son almacenados en la base de datos)
- Sin embargo, una llamada a un procedimiento es una orden PL/SQL en si misma, mientras que una llamada a una función se realiza como parte de una expresión.
- Una llamada a una función es un valor.

Para el ejemplo la siguiente función devuelve un valor TRUE si la clase especifica tiene ocupación mayor del 90% y FALSE en caso contrario.

3

5.1 Funciones

Creacion de Funciones

Para el ejemplo la siguiente función devuelve un valor TRUE si la clase especifica tiene ocupación mayor del 90% y FALSE en caso contrario.

```
CREATE  OR REPLACE  FUNCTION AlmostFull (
    p_Department   classes.department%TYPE,
    p_Course          classes.course%TYPE))
    RETURN BOOLEAN IS
    v_CurrentStudents              NUMBER;
    v_MaxStudents   NUMBER;
    v_ReturnValue    BOOLEAN;
    v_FullPercent     CONSTANT NUMBER := 90;
BEGIN
   -- Obtiene el valor actual y máximo de estudiantes para el cuyo solicitado
   SELECT   current_students, max_students
    INTO     v_CurrentStudents, v_MaxStudents
    FROM classes
    WHERE  deparment    = p_Department
    AND  course  = p_Course;
   --  Si la clase  esta mas llena  que el porcentaje  dado por  v_FullPercent , devuelve TRUE.  En caso Contrario,
FALSE.
    IF  (v_CurrentStudents   / v_MaxStudent  * 100  ) >  v_FullPercent  THEN
         v_ReturnValue   : = TRUE;
     ELSE
         v_ReturnValue   := FALSE;
     END IF;
     RETURN  v_ReturnValue;
END AlmostFull;
```

La función AlmostFull devuelve un valor booleano

4

5.1 Funciones

Creacion de Funciones

Se puede llamar a la función **AlmostFull** desde el siguiente bloque PL/SQL , en el que podemos observar que la función no es una orden en si misma, sino que usa como parte de la orden IF situada dentro del bucle.

```
DECLARE
  CURSOR  c_Classes IS
    SELEC T department, course
     FROM classes;
BEGIN
    FOR  v_ClassRecord  IN c_Classes LOOP
   -- Registra todos los cursos que no tienen mucho espacio vacio en  temp_table
    IF   AlmostFull( v_ClassRecord.department, v_ClassRecord.course ) THEN
         INSERT  INTO temp_table (char_col) VALUES
         (v_ClassRecord.department || ' ' || v_Classrecord.course ||  ' is almost full! ' );
    END IF;
  END LOOP
END AlmostFull;
```


5

5.1 Funciones

Sintaxis de la Funciones

Las sintaxis para crear una función almacenada es muy similar a la de un procedimiento:

```
CREATE  [OR REPLACE]  FUNCTION nombre_función
[(argumento [ {IN | OUT | IN OUT }] tipo,
…
  argumento [ {IN | OUT | IN OUT }] tipo)]
  RETURN  tipo_retorno  { IS | AS}
cuerpo_función
```

Donde ***nombre_función*** es el nombre de la función, ***argumento y tipo*** son iguales que un procedimiento, ***tipo_retorno*** es el tipo del valor que devuelve la función y ***cuerpo_función*** es un bloque PL/SQL que contiene el código de la función.

Al igual que con los procedimiento, la lista de argumento es opcional. Si no hay argumento no hay paréntesis ni en la declaración de la función ni en la llamada a ella. Sin embargo *el tipo de retorno de la función es obligatorio*, dado que la llamada a la función parte de una expresión. El tipo de la función se usa para determinar el tipo de la expresión que contiene la llamada a la función.

6

5.1 Funciones

La Orden RETURN

Dentro del cuerpo de la función la orden RETURN se emplea para devolver el control y un valor, al entorno que hizo llamada. Las sintaxis general de la orden RETURN es:

RETURN *expresión*;

Donde ***expresion*** es el valor que la función devuelve, el cual se convierte en el tipo especificado en la clausula RETURN de la definición de la función, si es que no es ya de ese tipo. Cuando se ejecuta la orden RETURN , se devuelve el control inmediatamente al entorno que hizo la invocación.

Puede haber de una orden RETURN e una función, aunque solo se ejecutara una de ellas. Es un error que una función concluya sin ejecutar una orden RETURN.

7

5.1 Funciones

La Orden RETURN

El siguiente ejemplo ilustra el caso de múltiples ordenes RETURN dentro de una función. Aunque se presentan 5 ordenes diferentes de RETURN en la función , solo una de ellas se ejecutara.

```
CREATE OR REPLACE FUNCTION ClassInfo (
        p_Department classes.department%TYPE,
        p_Course                    classes.course%TYPE)
       RETURN   VARCHAR2 IS
        v_CurrentStudents           NUMBER;
        v_MaxStudents               NUMBER;
        v_PercentFull               NUMBER;
BEGIN
   -- Obtiene la cantidad actual y máxima de estudiantes para  el curso solicitado
   SELECT  current_students,        max_students
    INTO v_CurrentStudens,  v_MaxStudents
    FROM  classes
    WHERE  department   = p_Department
     AND  course    = p_Course;
   -- Calcula el porcentaje  actual
   v_PercentFull   := v_CurrentStudents  / v_MaxStudents * 100;
    IF  v_PercentFull  = 100  THEN
       RETURN  'Full';
    ELSIF  v_PercentFull > 80  THEN
       RETURN 'Some Room';
    ELSIF  v_PercentFull > 60  THEN
       RETURN 'More Room';
    ELSIF  v_PercentFull >  0  THEN
       RETURN 'Lost of Room';
    ELSE
       RETURN 'Empty';
    END IF;
END ClassInfo;
```

/ Devuelve 'Full' si la clase esta completamente llena, 'Some Full' si la clase esta llena por encima del 80%, 'More Full' si la clase esta por encima del 60%, 'Lost of Room' si esta llena por debajo del 60% y 'Empty' sino hay estudiantes matriculados. */*

- Cuando se emplea una función, la orden RETURN debe tener una expresión asociada.

8

5.1 Funciones

Cuando utiliza una Función.

La función comparten muchas de las características de los procedimientos:

- Las funciones pueden devolver mas de un valor, mediante parámetros OUT
- El código de la función tiene secciones declarativas, ejecutables y de manejo de excepciones.
- Las funciones pueden aceptar valores predeterminados.
- Puede llamarse a las funciones utilizando notación posicional o nominal

Cuando debemos utilizar una función y cuando un procedimiento?

- Generalmente esto depende de cuantos valores deba devolver el programa y de como vaya a usarse dichos valores.
- Una regla practica es que se use un procedimiento siempre que haya mas de un valor de retorno. Si el valor de retorno es único, entonces se puede emplear una función.
- Aunque es legal que las funciones incluyan parámetros OUT (y por lo tanto devolverán mas de un valor), no resulta recomendable desde el punto de vista del estilo de programación.

9

5.1.1. PROCEDIMIENTO Y FUNCIONES

Eliminacion de Procedimiento y Funciones

Al que las tablas, los procedimientos y funciones también pueden ser eliminados, lo que los borra de diccionario de datos.

La sintaxis para eliminar un procedimiento es la siguiente

DROP PROCEDURE *nombre_procedimiento;*

La sintaxis para eliminar una función es la siguiente

DROP FUNCTION *nombre_función;*

Donde *nombre_procedimiento y nombre_funcion* son el nombre de un procedimiento o función existentes, respectivamente. Por ejemplo la eliminación de AddNewStudent;

DROP PROCEDURE AddNewStudent;

DROP es una orden DDL, asi que se ejecuta una orden COMMIT implícita tanto antes como despues de la orden DROP.

10

5. SITUACIONES DE LOS SUBPROGRAMAS

Los programas almacenados y el diccionario de los Datos

- Los subprogramas pueden ser almacenados en el diccionario de datos, como todos ejemplos mostrados. Primero se crea con la orden CREATE OR REPLACE y luego puede ser llamado desde otro bloque PL/SQL.
- También podemos definir un subprograma en sección declarativa de un bloque, en cuyo caso se denomina subprograma local.

Intentamos crear este procedimiento. Que pasara cuando se almacena en el diccionario de la Base de Datos?

```
SQL> create or replace procedure simple as
  2  v_counter number;
  3  begin
  4  v_counter := 7
  5  end simple;
  6  /

Advertencia: Procedimiento creado con errores de compilación.

SQL> show error
Errores para PROCEDURE SIMPLE:

LINE/COL ERROR
-------- -----------------------------------------------------------------
5/1      PLS-00103: Encountered the symbol "END" when expecting one of the
         following:
         * & = - + ; < / > at in is mod remainder not rem
         <an exponent (**)> <> or != or ~= >= <= <> and or like like2
         like4 likec between || multiset member submultiset
         The symbol ":" was substituted for "END" to continue.

SQL>
```


11

5.1.2 SITUACIONES DE LOS SUBPROGRAMAS

Los programas almacenados y el diccionario de los Datos

Podemos observar la situación de objetos que estamos creado accesando el contenedor de estos en la base de datos 'user_objects' .

Si desde un bloque intentamos invocar este procedimiento , se producirá el error:
PLS-905: object is invalid.

```
SQL> select object_name, object_type, status
  2   from user_objects
  3  where object_name = 'SIMPLE';

OBJECT_NAME
--------------------------------------------------------------------------------
OBJECT_TYPE         STATUS
------------------- -------
SIMPLE
PROCEDURE           INVALID


SQL>
```


12

5.1.2 SITUACIONES DE LOS SUBPROGRAMAS

Subprogramas Locales

Ejemplo de un subprograma local, declarado en la sección declarativa del PL/SQL.

```
DECLARE
   CURSOR c_AllStudents IS
     SELECT first_name, last_name
     FROM  students;
   v_FormattedName VARCHAR2(50);
   --Función que devolverá el nombre y apellido concatenado y separado por un espacio
    FUNCTION FormatName (p_FirstName IN VARCHAR2,
                         p_LastName  IN VARCHAR2)
    RETURN VARCHAR2 IS
    BEGIN
        RETURN  p_FirstName  || ' ' || p_LastName;
    END FormatName;
-- Inicia el programa principal
BEGIN
     FOR   v_StudentRecord IN c_AllStudents LOOP
           v_FormattedName := FormatName (v_StudentRecord.first_name,
                                          v_StudentRecord.last_name);
          INSERT INTO temp_table (char_col) VALUES (v_FormattedName);
     END LOOP;
END;
```

La función **FormatName** se declara en la sección declarativa del bloque anónimo. El nombre de la función es un identificador PL/SQL y sigue por lo tanto las misma reglas de ámbito y visibilidad que cualquier otro identificador. Para ser mas exacto, la función solo es visible en el bloque que ha sido declarada y su ámbito se extiende desde el punto de la declaración hasta el final del bloque. Ningún otro bloque puede hacer llamado a **FormatName**, dado que no es visible fuera del bloque.

13

5.1.2 SITUACIONES DE LOS SUBPROGRAMAS

Subprogramas Locales

Los subprogramas locales deben ser declarados al final de la sección declarativa. Si situamos **FormatName** por encima de la declaración de **C_AllStudents**, como se muestra en el siguiente ejemplo, obtendríamos un error de compilación.

```
DECLARE
 /* Declara en primer lugar FormatName. Esto generará un error de compilación, ya que todas las declaraciones tienen que estar antes de cualquier subprograma local. */
    FUNCTION FormatName (p_FirstName IN VARCHAR2,
                         p_LastName  IN VARCHAR2)
     RETURN VARCHAR2 IS
     BEGIN
         RETURN  p_FirstName  || ' ' || p_LastName;
     END FormatName;
     CURSOR c_AllStudents IS
      SELECT first_name, last_name
      FROM  students;
     v_FormattedName VARCHAR2(50);
-- Inicio del bloque principal
BEGIN
     NULL;
END;
```


14

FUNCION QUE CALCULA Seguro Social

- CREATE or REPLACE FUNCTION seguro_social(
- p_salarioq salarioquincenal.salario%type)
- RETURN number as
- Begin
- RETURN p_salarioq * (9.75/100);
- End seguro_social;

EN EL PROCEDIMIENTO

v_segurosocial := seguro_social(v_salarioq);