PL/SQL Cursores dinámicos

Antonio Espín Herranz

Consultas dinámicas

- SQL Dinámico.
- Execute immediate.
- Cursores dinámicos.

SQL Dinámico

- El SQL Dinámico es una técnica que permite crear sentencias SQL de forma dinámica durante la ejecución del código PL/SQL.

- Permite crear aplicaciones mas flexibles, ya que los nombres de los objetos pueden ser desconocidos en el momento de la compilación.
  - Por ejemplo, no conocer el nombre de una tabla.

SQL Dinámico

- **El SQL es estático, conocemos todos los datos en el momento de la compilación**.
- En cuanto a rendimiento, **el código estático tiene mejor rendimiento que el dinámico**, conocemos todos los objetos en tiempo de compilación y ya sabemos que son válidos antes de ejecutar, en cambio con el dinámico no.
- Esto implica un proceso de compilación por parte del motor de Oracle → consumo de tiempo.

SQL Dinámico

- ¿Cuándo utilizarlo?
  - El usuario tiene que introducir datos para crear los comandos SQL.
  - Hacer referencia a objetos que en tiempo de compilación no existen.

EXECUTE IMMEDIATE

- Permite verificar la sintaxis y ejecutar de forma dinámica la instrucción.
- **Sintaxis**:
  - EXECUTE IMMEDIATE cadena_dinamica
    - [into {variable, … | registro}]
    - [using [in | out | in out] argumento …]
    - [{ returning | return } into argumento, …]

- **Parámetros**:
  - Cadena_dinamica: El SQL.
  - Variable: Almacenar el valor de una columna.
  - Registro: Variable que contendrá un fila.
  - Argumentos: Especifica los valores pasados a la instrucción.

Ejemplo

```
DECLARE sql_stmt VARCHAR2(200);
        plsql_block VARCHAR2(500);
        emp_id NUMBER(4) := 7566;
        salary NUMBER(7,2);
        dept_id NUMBER(2) := 50;
        dept_name VARCHAR2(14) := 'PERSONNEL';
        location VARCHAR2(13) := 'DALLAS';
        emp_rec emp%ROWTYPE;
BEGIN
        EXECUTE IMMEDIATE 'CREATE TABLE bonus (id NUMBER, amt NUMBER)';
        sql_stmt := 'INSERT INTO dept VALUES (:1, :2, :3)';
        EXECUTE IMMEDIATE sql_stmt USING dept_id, dept_name, location;
        sql_stmt := 'SELECT * FROM emp WHERE empno = :id';
        EXECUTE IMMEDIATE sql_stmt INTO emp_rec USING emp_id;
        plsql_block := 'BEGIN emp_pkg.raise_salary(:id, :amt); END;';
        EXECUTE IMMEDIATE plsql_block USING 7788, 500;
        sql_stmt := 'UPDATE emp SET sal = 2000 WHERE empno = :1 RETURNING sal INTO :2';
        EXECUTE IMMEDIATE sql_stmt USING emp_id RETURNING INTO salary;
        EXECUTE IMMEDIATE 'DELETE FROM dept WHERE deptno = :num' USING dept_id;
        EXECUTE IMMEDIATE 'ALTER SESSION SET SQL_TRACE TRUE';
END;
```

Cursores Dinámicos
Open For, Fetch y Close

- Se emplean para tratar consultas dinámicas que se van a ejecutar sobre filas de datos.

- Al igual que en los cursores tradicionales hay que abrirlos con **Open For**.
- Se extraen filas con **Fetch** y por último,
- Se cierran con **Close**.

Open For

- Asocia una variable de tipo cursor con una consulta select que devuelva varias filas.
- El cursor dinámico tiene la particularidad que puede recibir parámetros al abrirlo.
- Sintaxis:
  - Open variable FOR consulta_dinámica using arg1, …
- Ejemplo:

```
Declare
        Type CliCurType is ref cursor;
        Ccli CliCurType;
        Vciudad clientes.ciudad%type:='Orense';
Begin
        Open ccli for 'Select numcli, ciudad from clientes where ciudad=:v'
        Using vciudad    -- El valor que tenga vciudad se coloca en :v
        …
End
/
```

Fetch

- Permite extraer valores de cada fila que se procesa.
- Normalmente esta instrucción estará ligada a un bucle.
- Sintaxis:
  - FETCH variable_cursor into {var1, var2, .. | registro}

Ejemplo

```
Declare
    Type CliCurType is ref cursor;
    Ccli CliCurType;
    Vnum clientes.numcli%type;
    Vciudad clientes.ciudad%type:='Orense';
Begin
    Open ccli for 'Select numcli, ciudad from clientes where ciudad=:v' using
      vciudad;
    Loop
        Fetch ccli into vnum, vciudad;        -- Extraer la fila;
        Exit when ccli%notfound;              -- Salir si no hay datos;
    End loop;
    …
End;
/
```

Close

- Cierra el cursor:
- Después de tratar todas las filas:
  - Close ccli;

- También se pueden utilizar los mismo atributos que con los cursores estáticos:
  - %NOTFOUND, %FOUND, %ISOPEN, %ROWCOUNT

Paso de parámetros

```
-- En la llamada pasamos el número de cliente a borrar.
Create or replace procedure borrar_clientes(vnumcli number) as
Begin
      Execute immediate ‘delete from clientes where numcli=:v’ using vnumcli;
End;
/

--Podemos pasar el nombre de un objeto:
Create or replace procedure borrar_tabla(nombre_tabla in varchar2) as begin
            execute immediate ‘Drop table ’ || nombre_tabla;
            -- No se pueden utilizar parámetros para los nombres de las tablas.
   end;
```