BULK BINDING

Elaborado por: Antonio Akiyama (antonio.akiyama@gbsperu.net)
Consultor Senior Business Intelligence Platform
Fecha: 4 de Abril del 2007
Versión: 1.0

BULK BINDING

Una de las ventajas que nos ofrece PL/SQL es la capacidad de embeber sentencias SQL (*select, insert, update, delete*) en los procedimientos almacenados o bloques anónimos sin tener que hacer ninguna declaración y/o configuración adicional. Sin embargo, durante la ejecución de estas sentencias SQL se producen *switches* entre el *SQL Engine* y el *PL/SQL Engine* de la base de datos Oracle que pueden afectar la *performance.* Para evitar estos *switches* es conveniente utilizar *bulk binding* y las estructuras denominadas colecciones (*nested-table, varray, associative array*).

Binding es la asignación de variables PL/SQL en una sentencia SQL. Por ejemplo: "SELECT count(1) INTO nContador FROM Origen", donde nContador sería la variable PL/SQL.

La asignación del total de los elementos de las colecciones en un único *switch* es conocido como *bulk binding*.

Para ilustrar su uso y beneficio, crearemos una tabla de origen con 40,000 registros y dos procedimientos (bloques anónimos) que nos permitan copiar esa información en una tabla destino. El primer procedimiento incurre en numerosos *switches*, mientras que el segundo los minimiza. Definitivamente este escenario puede ser mejor resuelto con un simple "INSERT INTO destino FROM SELECT …. " pero sólo se intenta ilustrar el uso de las colecciones para mejorar la *performance* del código PL/SQL.

1. Configuramos SQL Plus	SET SERVEROUTPUT ON SET TIMING ON
2. Creamos la tabla donde estará la información de origen (input)	CREATE TABLE Origen (Fecha DATE, DiaSemana VARCHAR2(15), Mes VARCHAR2(15));

3. Cargamos la tabla con 40,001 registros	DECLARE
	BEGIN
	FOR i IN 0..40000 LOOP
	INSERT INTO origen
	VALUES
	(SYSDATE - i,
	to_char(SYSDATE - i, 'DAY'),
	to_char(SYSDATE - i, 'MONTH')
	);
	END LOOP;
	COMMIT;
	END;
	/

4. Creamos la tabla destino con la misma estructura del origen pero sin registros	CREATE TABLE Destino AS SELECT * FROM Origen WHERE 1 = 2;

5. Utilizando un cursor explícito insertamos los registros en la tabla destino. Cada FETCH del cursor implica un switch	DECLARE
	CURSOR curOrigen IS SELECT * FROM Origen;
	BEGIN
	FOR registro IN curOrigen LOOP
	/* simulamos algún tipo de procesamiento */
	registro.mes := SUBSTR(registro.mes, 1, 1) || LOWER(SUBSTR(registro.mes, 2));
	/* insertamos los registros */
	INSERT INTO Destino
	VALUES (registro.fecha, registro.diasemana, registro.mes);
	END LOOP;
	COMMIT;
	END;
	/
	
	PL/SQL procedure successfully completed.
	
	Elapsed: 00:00:05.06

6. Utilizamos el mismo cursor explícito pero con una colección (nested-table) para evitar los switches	DECLARE
	TYPE arregloRegistros IS TABLE OF origen%ROWTYPE;
	l_arreglo arregloRegistros;
	
	CURSOR curOrigen IS SELECT * FROM Origen;
	
	BEGIN
	OPEN curOrigen;
	FETCH curOrigen **BULK COLLECT INTO** l_arreglo;
	
	/* simulamos algún tipo de procesamiento */
	FOR i IN l_arreglo.FIRST..l_arreglo.LAST LOOP
	l_arreglo(i).mes := SUBSTR(l_arreglo(i).mes, 1, 1) || LOWER(SUBSTR(l_arreglo(i).mes, 2));
	END LOOP;
	
	/* insertamos los registros */
	FORALL i IN l_arreglo.FIRST..l_arreglo.LAST
	INSERT INTO Destino VALUES l_arreglo(i);
	
	COMMIT;
	
	CLOSE curOrigen;
	END;
	/
	
	PL/SQL procedure successfully completed.
	
	Elapsed: 00:00:00.64