L’interaction entre le moteur PL/SQL et le moteur SQL est fréquente. Pour réduire les performances, il faut réduire le nombre d’aller-retour entre les deux moteurs. La solution est la copie de données par blocs. Les copies de blocs se feront par des collections.

BULK COLLECT pour les interrogations et FORALL pour les modifications.

BULK COLLECT

Cette clause indique au moteur SQL que les données seront retournées dans une collection avant de retourner dans le moteur PL/SQL.

Instructions SELECT qui extraient plusieurs lignes en une seule extraction, ce qui accélère la récupération des données.

Les gains de temps sont spectaculaires.

… BULK COLLECT INTO nom\_collection [,nom\_collection,…]

Extraction du nom des employés dans une collection puis affichage des valeurs de la collection.

DECLARE

TYPE t\_emp is TABLE OF emp.ename%type;

v\_emp t\_emp;

BEGIN

select ename BULK COLLECT INTO v\_emp from emp;

FOR i in v\_emp.first..v\_emp.last loop

dbms\_output.put\_line(v\_emp(i));

end loop;

END;

/

Il est donc possible de ramener les données depuis une table vers une collection et ensuite de modifier les données par l’intermédiaire d’un FORALL.

Idem que précédemment puis modification des salaires par FORALL.

DECLARE

TYPE t\_emp is TABLE OF emp.empno%type;

v\_emp t\_emp;

BEGIN

select empno BULK COLLECT INTO v\_emp from emp;

FORALL i in indices of v\_emp

update emp set sal=sal\*2 where empno=v\_emp(i);

END;

/

La clause LIMIT est limitée à la copie par blocs depuis un curseur. Elle permet de limiter le nombre de lignes ramenées par le FETCH.

FETCH … BULK COLLECT INTO … [LIMIT nb\_lignes]

DECLARE

TYPE t\_emp is TABLE OF emp.empno%type;

v\_emp t\_emp;

CURSOR C\_emp IS select empno from emp;

BEGIN

open c\_emp;

loop

fetch c\_emp BULK COLLECT INTO v\_emp LIMIT 5;

exit when c\_emp%notfound;

for i in v\_emp.first..v\_emp.last loop

dbms\_output.put\_line(v\_emp(i));

end loop;

end loop;

close c\_emp;

END;

/

FORALL

Cette clause indique au moteur PL/SQL de travailler par blocs avec une collection avant d’envoyer la commande SQL (update, insert ou delete) au moteur SQL.

Obligation de passer par une collection et il doit exister des éléments dans la collection pour toutes les valeurs d’indice.

Si une erreur se produit, c’est la totalité qui est annulée (rollback). Pour permettre de continuer même en cas d’erreur, il faut ajouter la clause SAVE EXCEPTIONS dans l’instruction FORALL.

Si un ordre SQL échoue, aucune exception n’est levée et les informations relatives à l’erreur sont stockées dans la collection SQL%BULK\_EXCEPTIONS. Quand FORALL est treminé, l’exception ORA-24381 est levée. SQL%BULK\_EXCEPTIONS.COUNT donne le nombre d’erreurs.

SQL%BULK\_EXCEPTIONS(i).ERROR\_INDEX donne le numéro de l’instruction qui a échoué.

SQL%BULK\_EXCEPTIONS(i).ERROR\_CODE donne le code de l’erreur.

FORALL indice IN borne\_inférieure .. borne\_supérieure

Instruction DML

create table t1 (code number(8), nom varchar2(30));

truncate table t1;

alter system flush shared\_pool;

alter system flush buffer\_cache;

begin

for i in 1..2000000 loop

insert into t1 values (i,'NOM'||i);

end loop;

end;

/

Temps : 90s

truncate table t1;

alter system flush shared\_pool;

alter system flush buffer\_cache;

declare

type t\_code is table of t1.code%type INDEX BY PLS\_INTEGER;

type t\_nom is table of t1.nom%type INDEX BY PLS\_INTEGER;

v\_code t\_code;

v\_nom t\_nom;

begin

for i in 1..2000000 loop

v\_code(i) := i ;

v\_nom(i) := to\_char( 'NOM'||i ) ;

end loop ;

FORALL i in 1..2000000

insert into t1 values (v\_code(i),v\_nom(i));

end;

/

Temps : 10s