**Bulk collect**

Bulk collect is about reducing context switching and improving the query performance.

Context switching

Whenever we write PL/SQL block and execute it. The PL/SQL runtime engine starts processing it line by line , this engine processes all the pl/sql statements by itself but it passes all the SQL statement to the SQL engine. Those SQL statements will then get processed separately by the SQL engine.

Once it is done processing then the SQL engine then return the result back to the PL/SQL engine.so that combined result can be produced latter.

This to and fro hopping of control is called context switching.

Bulk collect clause reduces multiple control hopping by collecting all the SQL statement calls from the PL/SQL program and sending them to SQL engine in just one go and vice verse.

**Definition**

Bulk collect clause reduces/compresses multiple switches into a single context switch and increases the efficiency and performance of a PL/SQL program.

The process of fetching batches of data from PL/SQL runtime engine to SQL runtime engine and vice versa is called bulk data processing.

Bulk collect clause can be used with

* SELECT – INTO
* FETCH – INTO
* RETURNING – INTO

Example of SELECT-INTO

set serveroutput on;

DECLARE

TYPE tab\_name IS TABLE OF VARCHAR2(200);

rec\_name tab\_name;

BEGIN

SELECT first\_name

INTO rec\_name

FROM employees;

END;

In above example, incase employees table has 50 records then there will be 100 hopping between pl/sql and sql engine

1. From pl/sql engine to SQL engine
2. From SQL engine to pl/sql engine

And hence reducing the performance

It gives error as well

PLS-00642: local collection types not allowed in SQL statements

06550. 00000 - "line %s, column %s:\n%s"

\*Cause: Usually a PL/SQL compilation error.

\*Action:

Rewriting it

DECLARE

TYPE tab\_name IS TABLE OF VARCHAR2(200);

rec\_name tab\_name;

BEGIN

SELECT first\_name

BULK COLLECT INTO rec\_name

FROM employees;

END;

Here instead of processing each row from plsql engine to sql engine , bulk collect with get all 50 records at a time and reduces the hopping to two.

DECLARE

TYPE tab\_name IS TABLE OF VARCHAR2(200);

rec\_name tab\_name;

BEGIN

SELECT first\_name

BULK COLLECT INTO rec\_name

FROM employees;

FOR i in 1..rec\_name.COUNT

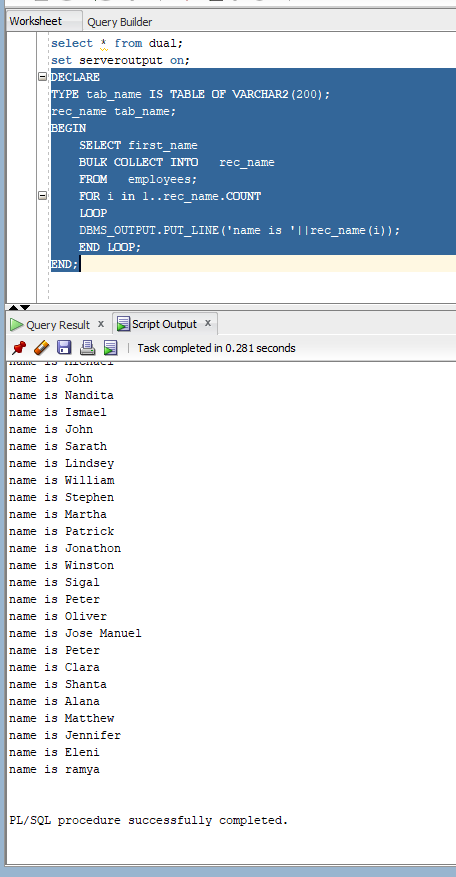
LOOP

DBMS\_OUTPUT.PUT\_LINE('name is '||rec\_name(i));

END LOOP;

END;

/



**FETCH INTO example**

SET sERVEROUTPUT ON;

DECLARE

CURSOR emp\_cur IS

SELECT first\_name

FROM employees;

TYPE my\_name IS TABLE OF VARCHAR2(20);

rec\_name my\_name;

BEGIN

OPEN emp\_cur;

LOOP

FETCH emp\_cur BULK COLLECT INTO rec\_name;

EXIT WHEN rec\_name.count=0;

FOR i in 1..rec\_name.count

LOOP

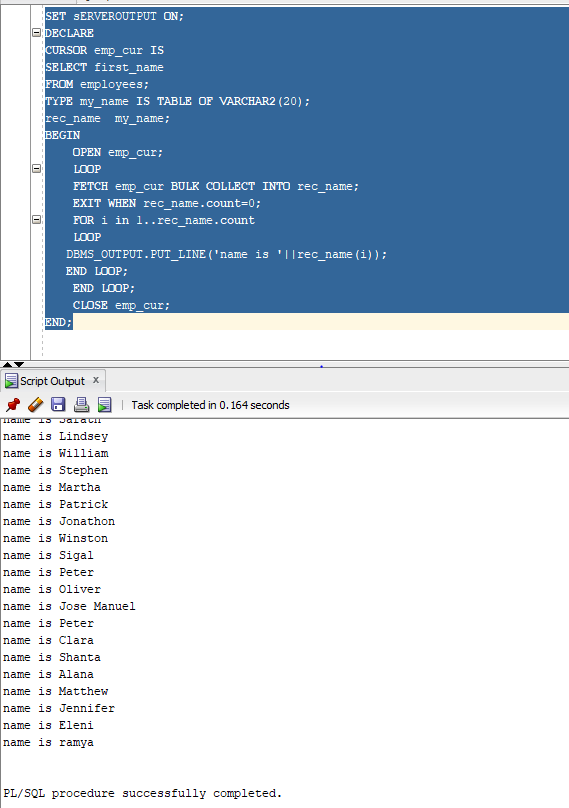
DBMS\_OUTPUT.PUT\_LINE('name is '||rec\_name(i));

END LOOP;

END LOOP;

CLOSE emp\_cur;

END;



**LIMIT**

Bulk collect clause will cause memory overhead. Our programs start consuming a lot of memory in order to be fast and efficient .That degrades the performance of the database.

This means that our query must surely be performing well but at the same time our database may not.

SET sERVEROUTPUT ON;

DECLARE

CURSOR emp\_cur IS

SELECT first\_name

FROM employees;

TYPE my\_name IS TABLE OF VARCHAR2(20);

rec\_name my\_name;

BEGIN

OPEN emp\_cur;

FETCH emp\_cur BULK COLLECT INTO rec\_name **limit 10**;

FOR i in 1..rec\_name.count

LOOP

DBMS\_OUTPUT.PUT\_LINE('name is '||rec\_name(i));

END LOOP;

CLOSE emp\_cur;

END;

**FORALL statement**

FORALL statement reduces context switches which occur during the execution of a DML statement in a loop.

FORALL is a bulk loop construct which executes one DML statement multiple times at once.

Syntax

FORALL index IN bound\_clause

[SAVE EXECEPTION]

DML statement ;

Unlike FOR loop, with FORALL statement we can only use one DML at a time.

Bound clauses

The bound clause controls the value of index as well as decides the number of iteration of a FORALL statement.

There are three types

1. Lower and upper bound
2. Indices of
3. Values of

Example 1

create table tut\_77(mul\_9 number);

DECLARE

TYPE my\_array IS TABLE of NUMBER INDEX BY PLS\_INTEGER;

col\_var my\_Array;

tot\_rec NUMBER;

BEGIN

FOR i in 1..10

LOOP

col\_Var(i):=9\*i;

END LOOP;

FORALL i in 1..10

insert into tut\_77 values(col\_Var(i));

select count(\*)

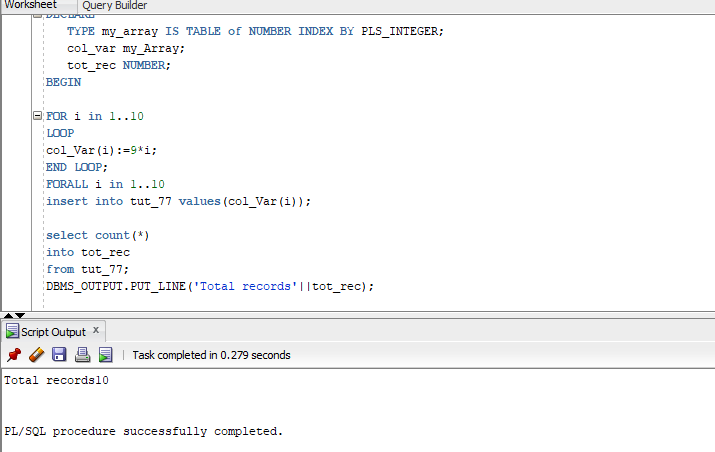
into tot\_rec

from tut\_77;

DBMS\_OUTPUT.PUT\_LINE('Total records'||tot\_rec);

END;

/



Example 2:

DECLARE

TYPE my\_array IS TABLE of NUMBER ;

col\_var my\_Array:=my\_Array(9,18,27,36,45,54,63,72,81,90);

tot\_rec NUMBER;

BEGIN

col\_Var.delete(3,6);

FORALL i in 1..10

insert into tut\_77 values(col\_Var(i));

rror report -

ORA-22160: element at index [3] does not exist

ORA-06512: at line 7

22160. 00000 - "element at index [%s] does not exist"

\*Cause: Collection element at the given index does not exist.

\*Action: Specify the index of an element which exists.

END;

/

Here because 4 elements are deleted from collection upper bound and lower bound throws error

It can solved by indices

DECLARE

TYPE my\_array IS TABLE of NUMBER ;

col\_var my\_Array:=my\_Array(9,18,27,36,45,54,63,72,81,90);

tot\_rec NUMBER;

BEGIN

col\_Var.delete(3,6);

FORALL i in **indices of col\_var**

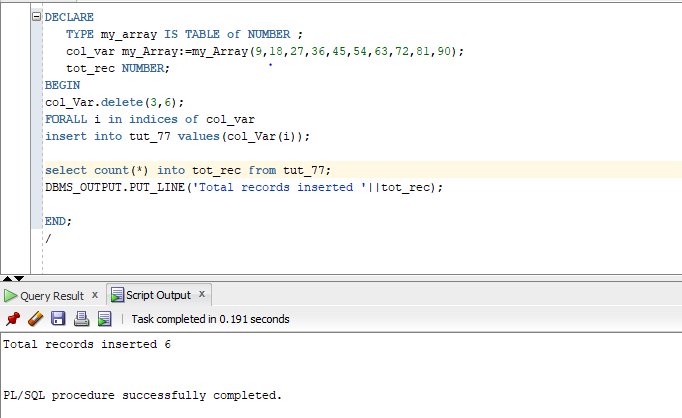
insert into tut\_77 values(col\_Var(i));

select count(\*) into tot\_rec from tut\_77;

DBMS\_OUTPUT.PUT\_LINE('Total records inserted '||tot\_rec);

END;

/



DECLARE

TYPE my\_nested\_Table IS TABLE of NUMBER ;

col\_var my\_nested\_Table:=my\_nested\_Table(9,18,27,36,45,54,63,72,81,90);

TYPE my\_Array is table of PLS\_INTEGER index by PLS\_INTEGER;

index\_col my\_Array;

BEGIN

index\_col(1):=3;

index\_col(5):=7;

index\_col(12):=8;

index\_col(28):=10;

FORALL i **in values of index\_col**

insert into tut\_77 values(col\_Var(i));

END;

/

Disadvantages of BULK COLLECT

1. We cannot bulk collect into an associative array having a string type for the key.
2. Bulk collect can only be used at server side
3. Collections should be used as target variables listed in a bulk collect into clause
4. Composite targets (such as objects) cannot be used in the returning caluse
5. Multiple composite target cannot be used in the bulk collect when implicit datatypr conversion needed.