Collections

A homogeneous single dimension data structure which is made of elements of same datatype is called collections.

Structure of PL/SQL collections consist of a cell with subscript called index.

Data is stored into these cells and can be accessed and identified using index number.

|  |
| --- |
| Index(i) |
| Data(d) |

Types of Collections

Persistent collection are those which store the collection structure with the data, physically into the database and can be accessed again if needed.

Non persistent collection only stores data and structure just for one session.

Collections

Persistent Non persistent

Nested table VaRRAY Associative Array

Nested table – are persistent collection and has no upper limit on rows.

VARRARY – are bounded in nature

Associative array – non persistent collections. They cannot be reused. Available in PL/SQL block for the session.

Nested table –

Syntax

TYPE nested\_table\_name IS TABLE OF element\_type [not null];

Example 1 :-

DECLARE

TYPE my\_nested\_table IS TABLE OF NUMBER;

Var\_nt my\_nested\_table:= my\_nested\_table(9,18,27,36,45,54);

Dbms\_ouput.put\_line(var\_nt(1));

For I in 1..var\_nt.count

LOOP

Dbms\_ouput.put\_line(‘Value at index ‘||i||’is’||var\_nt(i));

End loop;

End;

/

Example 2: created nested table on primitive or user defined datatype

Create or replace TYPE my\_nested\_table IS tABLE OF VARCHAR2(10);

/

CREATE table my\_subject(sub\_id number(5),sub\_name varchar2(10),sub\_schedule my\_nested\_table)

Nested table sub\_schedule STORE AS nested\_tab\_space;

Insert into my\_subject values (10,’Maths’, my\_nested\_table(‘Mon’,’Fri’));

Insert into my\_subject values (20,’Computer’, my\_nested\_table(‘Wed,’Thu’));

Example 3:

Create or replace type object\_type as object(obj\_id number, obj\_name varchar2(10));

/

Create or replace type my\_nt is table of type object\_type;

/

Create table base\_Table(tab\_id number,

Tab\_ele my\_nt)

Nested table tab\_ele store as store\_Tab\_1;

VARRARY

Variable sized array.

Unlike nested tables which requires an external table for its storage VARRAY are stored in-line with their parent record as row value in the parent table.

Varray similar to nested tables are persistent type of collection.

Which can be created

Database object

As a member of PL/SQL block

Syntax

CREATE [OR REPLACE ] TYPE type\_naem IS [VARRAY | VARRYING ARRAY ] (size limit) OF element\_Type;

VARRAY as block member

DECLARE

TYPE type\_naem IS (VARRAY|VARRYING ARRAY) (size limit) OF element\_Type;

Modify size limit

ALTER TYPE type\_name modify limit new\_size\_limit [invalidate |cascade ]

Example :1

Declare

Type inblock\_varray is varray (5) of number;

Var\_obj inblock\_varray:= inblock\_varray();

Begin

For i in 1..var\_obj.count

Loop

Var\_obj.extend;

Var\_obj(i):=10\*i;

Dbms\_ouptut.put\_line(var\_obj(i));

End loop;

End;

/

Or

Declare

Type inblock\_varray is varray (5) of number;

Var\_obj inblock\_varray:= inblock\_varray();

Begin

Var\_obj.extend(5);

For i in 1..var\_obj.count

Loop

Var\_obj(i):=10\*i;

Dbms\_ouptut.put\_line(var\_obj(i));

End loop;

End;

/

AS database object

Create or replace type dbobj\_vry is varray (5) of number;

/

Create table calendar(

Day\_name varchar2(25),

Day\_date dbobj\_vry);

Insert into calendar values(‘SUNDAY’, dbobj\_vry(7,14,21,28));

Select tab1.day\_name,vary.column\_value

From calendar tabl,table(tab1.day\_date) vry;

Associative Array

Hold similar kind of data type with key – value pair.

Non- persistent which means either array nor data will be stored in database.

Example:-

Declare

Type books books is table of number index by varchar2(20);

Isbn books;

Flag varchar2(50);

Begin

Isbn(‘Oracle Database’) :=1234;

Isbn(‘MYSQL’) L=9876;

Isbn(‘MYSQL’) L=1011;

Dbms\_output.put\_line(‘value’ || isbn(‘Oracle Database’));

Flag isbn.first;

While flag is not null

Loop

Dbms\_output.put\_line(‘Key 🡪’ ||flag ||’value 🡪’ ||isbn(flag));

Flag:=isbn.next(flag);

End loop;

End;

/

Collection Methods

Collection methods are PL/SQL built-in functions and procedures which can be used in conjunction with collection.

There are 10 collections methods

|  |  |
| --- | --- |
| functions | procedures |
| Count | Delete |
| Exists | Extend |
| First,LAST | Trim |
| Perior, NEXT |  |
| limit |  |

How do we use collection methods

Collection.method.

Count – returns the number of elements in an initialize collection. If used with an initialize collection with no elements it returns zero.

In case of nested table collection method, count() only return the number of non empty element.

Example:

Declare

Type my\_nested\_table is table of number;

Var\_nst my\_nested\_table:= my\_nested\_table(9,18,27,36,45,54,63,72,81,90);

Begin

Dbms\_output.put\_line(‘The size of nested table is ‘||var\_nst.count);

If var\_nst.count> 10 then

Dbms\_ouput.put\_line(‘you can perfor any action’);

End if;

For in 1 .. var\_nst.count

Loop

Dbms\_output.put\_line(‘value stored at index ‘||i|||’is ‘||var\_nst(i));

End loop;

End;

/

Exists - checks the existence of an element at a specific index in a collection.

Declare

Type my\_nested\_table is table of varchar2(20);

Var\_nst my\_nested\_table:= my\_nested\_table(‘batman’,’superman’,’ironman’);

Begin

If Var\_nst.exists(1) then

Dbms\_output.put\_line(‘found the element’);

Else

Dbms\_output.put\_line(‘sorry didt find the element’);

If var\_nst.exists(4) then

Dbms\_output.put\_line(‘found the element’||var\_nst(4));

Else

Var\_nst.extend;

Var\_nst(4):=’spiderman’;

End if;

Dbms\_output.put\_line(‘new element inserted at index 2 is ‘||var\_nst(4));

End ;

First and last

To know the first and last index values defined in the collection.

Can be used with all three collection types.

Example

Declare

Type my\_nested\_table is table of varchar2(20);

Var\_nst my\_nested\_table:= my\_nested\_table(‘batman’,’superman’,’ironman’);

Begin

Dbms\_output.put\_line(‘first elelment is ‘||var\_nst.first);

Dbms\_output.put\_line(‘first elelment is ‘||var\_nst.last);

Var\_nst.delete(1);

Dbms\_output.put\_line(‘first elelment is ‘||var\_nst.first);

Var\_nst.trim;

Dbms\_output.put\_line(‘first elelment is ‘||var\_nst.last);

Dbms\_output.put\_line(‘first elelment is ‘||var\_nst (var\_nst.last)); //to print last element instead if index

End;

/

LIMIT

Is actually PL/SQL function returns the maximum number of elements that a VARRAY can hold.

Declare

Type inblock\_vary is varray (5) of number;

Vry\_obj inblock\_vary:= inblock\_vary();

Begin

Dbms\_output.put\_line(‘total indexex ‘||vry\_obj.limit); //output 5

Dbms\_output.put\_line(‘result of function limit ‘||vry\_obj.count); // output 0 (bcoz collection is not storing any element yet)

Vry\_obj.extend;

Var\_obj(1):=10;

Dbms\_output.put\_line(‘result of function limit ‘||vry\_obj.count); // output 1

End ;

/

Limit for nested tables

Declare

Type my\_nested\_table is table of number;

Var\_nst my\_nested\_table:= my\_nested\_table(9,18,27,36,45,54,63,72,81,90);

Begin

Dbms\_output.put\_line(‘The size of nested table is ‘||var\_nst.limit); // output will retunf nothing bcoz it wont work for nested and associative array)

Dbms\_output.put\_line(‘The size of nested table is ‘||var\_nst.count); // ouput 10

End;

/

Prior and next

Prior – takes an index as input and return the value stored into the previous lowest index.

Next – returns values from next higher index.

Can be used with all three collection types.

Declare

Type my\_nested\_table is table of number;

Var\_nst my\_nested\_table:= my\_nested\_table(9,18,27,36,45,54,63,72,81,90);

Begin

Dbms\_output.put\_line(‘index before 3rd index is ‘||var\_nst.prior(3)); // return index

Dbms\_output.put\_line(‘data before 3rd index is ‘||var\_nst (var\_nst.prior(3))); // ouput is 18

Dbms\_output.put\_line(‘index after 3rd index is ‘||var\_nst.next(3));

Dbms\_output.put\_line(‘data before 3rd index is ‘||var\_nst (var\_nst.next(3)));

End;

/

delete

overloaded procedure.

If delete is used without any parameter then it will delete all the elements from collection, else it will delete specific element passed as index.

Declare

Type my\_nested\_table is table of number;

Var\_nst my\_nested\_table:= my\_nested\_table(9,18,27,36,45,54,63,72,81,90);

Begin

Dbms\_output.put\_line(‘value stored at index 5 before delete’ ||var\_nst(5));

Var\_nst.delete(5);

Dbms\_output.put\_line(‘value stored at index 5 after delete’ ||var\_nst(5)); // throw no data found it can be solved with below code

If var\_nst(5).exists then

Dbms\_output.put\_line(‘value stored at index 5 after delete’ ||var\_nst(5));

Else

Dbms\_output.put\_line(‘data is delted’);

Var\_nst.delete(2,6);

For I in 1..var\_nst.last

Loop

If var\_nst(i).exists then

Dbms\_output.put\_line(‘value stored at index ‘||i||var\_nst(5));

End if;

End loop;

End;

/

extend

overloaded pl/sql procedure which is used for appending elements to the collection.

Extend – extend single null element

Extend(1)- extend 1 number of elements with null

Extend (n,v) – n, number of elements to be appended

V,index number whose value will be copied and assigned to each of the newly appended element.

Declare

Type my\_nested\_table is table of number;

Var\_nst my\_nested\_table:= my\_nested\_table();

Begin

Var\_nst.extend;

Var\_nst(1):=10;

Dbms\_output.put\_line(‘data at index 1 is ‘||var\_nst(1));//10

Var\_nst.extend(3); /// to add three elements

Var\_nst(2):=10;

Var\_nst(3):=10;

Var\_nst(4):=10;

Var\_nst.extend(5,1); //5 elemetns will be appended with values present at index 1

End;

/

TRIM

Used to remove one or more elements from end of the collection.

Trim- will remove one element from end.

Trim(n) -remove n number of elements from end of the collection.

Declare

Type my\_nested\_table is table of number;

Var\_nst my\_nested\_table:= my\_nested\_table(1,2,3,4,5);

Begin

Var\_nst.trim;

Dbms\_ouput.put\_line(‘after trim procedrue’);

For I in 1..var\_nst.count

Loop

Dbms\_ouput.put\_line(var\_nst(i)); //1 2 3 4

End loop;

Var\_nst.trim(3);

Dbms\_ouput.put\_line(‘after trim procedrue’);

For I in 1..var\_nst.count

Loop

Dbms\_ouput.put\_line(var\_nst(i)); //1

End loop;

End;

/