

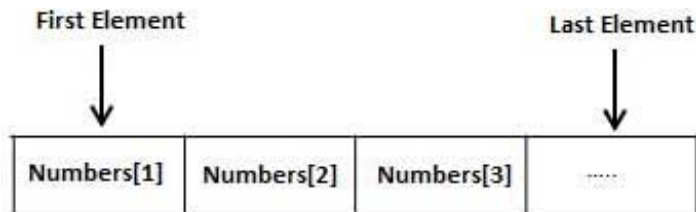
PL/SQL - ARRAYS

http://www.tutorialspoint.com/plsql/plsql_arrays.htm

Copyright © tutorialspoint.com

PL/SQL programming language provides a data structure called the VARRAY, which can store a fixed-size sequential collection of elements of the same type. A varray is used to store an ordered collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

All varrays consist of contiguous memory locations. The lowest address corresponds to the first element and the highest address to the last element.



An array is a part of collection type data and it stands for variable-size arrays. We will study other collection types in a later chapter 'PL/SQL Collections'.

Each element in a varray has an index associated with it. It also has a maximum size that can be changed dynamically.

Creating a Varray Type

A varray type is created with the CREATE TYPE statement. You must specify the maximum size and the type of elements stored in the varray.

The basic syntax for creating a VRRAY type at the schema level is:

```
CREATE OR REPLACE TYPE varray_type_name IS VARRAY(n) OF <element_type>
```

Where,

- *varray_type_name* is a valid attribute name,
- *n* is the number of elements (maximum) in the varray,
- *element_type* is the data type of the elements of the array.

Maximum size of a varray can be changed using the ALTER TYPE statement.

For example,

```
CREATE Or REPLACE TYPE namearray AS VARRAY(3) OF VARCHAR2(10);  
/  
Type created.
```

The basic syntax for creating a VRRAY type within a PL/SQL block is:

```
TYPE varray_type_name IS VARRAY(n) OF <element_type>
```

For example:

```
TYPE namearray IS VARRAY(5) OF VARCHAR2(10);  
Type grades IS VARRAY(5) OF INTEGER;
```

Example 1

The following program illustrates using varrays:

```
DECLARE
    type namesarray IS VARRAY(5) OF VARCHAR2(10);
    type grades IS VARRAY(5) OF INTEGER;
    names namesarray;
    marks grades;
    total integer;
BEGIN
    names := namesarray('Kavita', 'Pritam', 'Ayan', 'Rishav', 'Aziz');
    marks := grades(98, 97, 78, 87, 92);
    total := names.count;
    dbms_output.put_line('Total ' || total || ' Students');
    FOR i in 1 .. total LOOP
        dbms_output.put_line('Student: ' || names(i) || '
            Marks: ' || marks(i));
    END LOOP;
END;
```

When the above code is executed at SQL prompt, it produces the following result:

```
Student: Kavita Marks: 98
Student: Pritam Marks: 97
Student: Ayan Marks: 78
Student: Rishav Marks: 87
Student: Aziz Marks: 92

PL/SQL procedure successfully completed.
```

Please note:

- In oracle environment, the starting index for varrays is always 1.
- You can initialize the varray elements using the constructor method of the varray type, which has the same name as the varray.
- Varrays are one-dimensional arrays.
- A varray is automatically NULL when it is declared and must be initialized before its elements can be referenced.

Example 2

Elements of a varray could also be a %ROWTYPE of any database table or %TYPE of any database table field. The following example illustrates the concept:

We will use the CUSTOMERS table stored in our database as:

```
Select * from customers;
```

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00

Following example makes use of **cursor**, which you will study in detail in a separate chapter.

```
DECLARE
    CURSOR c_customers is
    SELECT name FROM customers;
```

```

type c_list is varray (6) of customers.name%type;
name_list c_list := c_list();
counter integer :=0;
BEGIN
  FOR n IN c_customers LOOP
    counter := counter + 1;
    name_list.extend;
    name_list(counter) := n.name;
    dbms_output.put_line('Customer'||counter||':'||name_list(counter));
  END LOOP;
END;
/

```

When the above code is executed at SQL prompt, it produces the following result:

```

Customer(1): Ramesh
Customer(2): Khilan
Customer(3): kaushik
Customer(4): Chaitali
Customer(5): Hardik
Customer(6): Komal

PL/SQL procedure successfully completed.

```