Chapter 23

VARRAY AND NESTED TABLE

- What is a collection?
- What is a VARRAY?
- Using VARRAY
- Nested Table
- Using DML commands with Nested Table
- Collection methods

What is a collection?

A collection is a group of values where all values are of the same type. Oracle provides three types of collections – Indexed Tables, Nested Tables, and VARRAYs.

All these collections are like a single dimension array. A column of a table can be declared as of a collection type, an attribute of an object can be of collection type and a collection can also contain a collection of object types.

First let us see what is a VARRAY and how to use it.

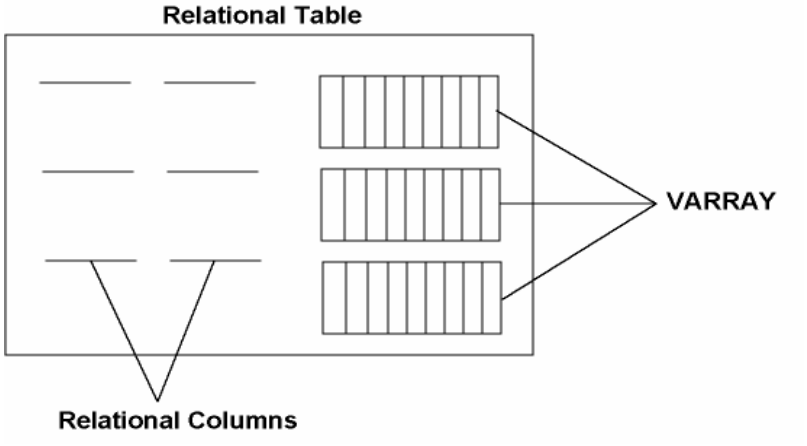
What is a VARRAY?

VARRAY stands for variable-size array. It is an array that can be either manipulated as a whole or individually as elements. It has a maximum size and can contain 0 to any number of elements up to the maximum specified.

VARRAY is stored *in-line*. That means the data of a column of VARRAY type is stored along with the remaining data of the row.

You can use VARRAY data type for:

- A column in a relational table
- A PL/SQL variable
- A PL/SQL parameter of procedure or function
- A PL/SQL function return type
- A data attribute of an object type



Using VARRAY

The following procedure illustrates how to declare and use VARRAY.

The following is the syntax to create a data type of VARRAY.

```
CREATE TYPE array_name AS VARRAY (limit)
       OF data_type
```

Array_name	is the name of the VARRAY data type.
Limit	is the maximum number of elements that the array can have
Data_type	is the type of each element of the array. It can be any standard type or object type.

Example:

First create an object type called PROJECT_TYPE to store the name of a project and the role played by the employee in that project.

```
create type project_type as object
(
  name varchar2(50),
  role varchar2(20)
);
```

Create a VARRAY that can contain up to 5 elements of PROJECT_TYPE.

```
create type projectlist as VARRAY(5) of project_Type;
```

Then create EMP table where column PROJECTS is of type PROJECTLIST.

```
create table emp
( empno number(5),
  ename varchar2(30),
  projects  projectlist
);
```

The following INSERT command will insert a row into EMP table with data for PROJECTS column.

```
insert into emp
 values(1,'Ellison',
           projectlist
           (
               project_type('Telephone Billing', 'System Analyst'),
               project_type('Housing Loans','Oracle DBA')
           )
        );
```

While inserting a row into EMP table use constructor of PROJECTLIST and then the constructor of PROJECT_TYPE for each project the employee is handling. The above INSERT command creates a single row in EMP table where employee is involved in two projects.

The following command displays the details of projects of employee with number 1.

```
SQL> select projects from emp where empno = 1;

PROJECTS(NAME, ROLE)
--------------------------------------------------------------------------------
----------------------
PROJECTLIST(PROJECT_TYPE('Telephone Billing', 'System Analyst'),
PROJECT_TYPE('Housing Loans', 'Oracle DBA'))
```

However, it is not possible to access the details of a single project as follows:

```
SQL>select projects(1) from emp where empno= 1;
       *
ERROR at line 1:
ORA-00904: invalid column name
```

So, a VARRAY is always manipulated as a single value in SQL. The following update is used to change the role of employee 1 in Telephone Billing project to Project Leader.

```
update emp
set projects =
           projectlist
           (
               project_type('Telephone Billing','Project Leader'),
               project_type('Housing Loans','Oracle DBA')
           )
where empno = 1;
```

However, it is possible to handle individual elements using collection methods in PL/SQL. The following function is used to check whether an employee is in the part of the given project.

```
create or replace Function IsPartOfProject(p_empno number, p_projectname
varchar2)
return  Boolean
is
  pl ProjectList;    -- VARRAY type to hold projects of an employee
begin

  /* Get the VARRAY of projects */
  select  projects  into    pl
  from    emp
  where   empno = p_empno;

  /*check whether the given project name is existing
       in the list of projects*/

     for idx in  1 .. pl.count
     loop

        /* check whether project name is same as the parameter */

        if pl(idx).name = p_projectname then
           -- exit the function by returning true
           return  true;
        end if;

    end loop;

    /* Return false if project name is not found
              in the projects of the employee */

    return  false;
end;
```

The above function starts by declaring a variable – PL – as of type PROJECTLIST. Then we read value of column PROJECTS, which is of type PROJECTLIST, into variable PL.

COUNT method of collection is used to return the number of elements in the collection. We used it to set the loop that varies IDX from 1 to count of elements. As it possible to access individual elements in PL/SQL, we used IDX as subscript to access each element of array PL.

The expression *PL(IDX).NAME* returns the name of the project from the element at the given position. If it is equal to project name parameter then the function returns true and function is terminated. Otherwise after the loop is terminated the function return false.

Note: *Currently, you cannot reference the individual elements of a varray in an* `INSERT`*,* `UPDATE`*, or* `DELETE` *statement. However, it is possible to add manipulate data using PL/SQL.*

The following procedure is used to add a new project. It takes employee number, project name and role of the employee in the project.

```
create or replace procedure new_project
          (p_empno number, p_projname varchar2, p_role varchar2)
as
 pl projectlist;
begin
   -- get varray of the employee
   select projects into pl from emp2
      where empno = p_empno;

   pl.extend;  -- make room for new project

   -- place an object of project_type at the end of the varray
   pl(pl.last) := project_type (p_projname, p_role);

   -- change the data in the table
   update emp2 set projects = pl
      where empno = p_empno;
end;
/
```

Nested Table

Nested table is a table that is stored in database as the data of a column of the table. Nested table is like an Index-By table, but the main difference is that a nested table can be stored in the database and an Index-by table cannot.

Nested table extends Index-by table by allowing the operations such as SELECT, DELETE, UPDATE and INSERT to be performed on nested table.

The following example illustrates steps related to creating and using nested table.

```
create type project_type as object
(
  name varchar2(50),
  role varchar2(20)
);
```

Now create a TABLE data type as follows:

```
create type ProjectTable as Table of Project_type;
```

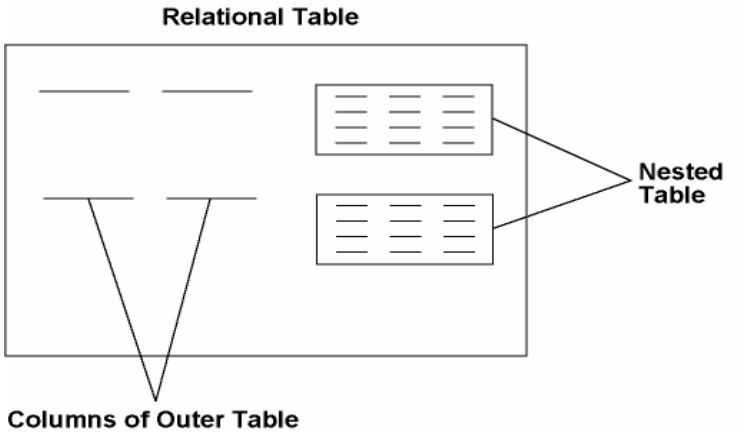
Finally we use PROJECTTABLE type to create a column in EMP table as follows:

```
create table emp
( empno number(5),
  ename varchar2(30),
  projects  projecttable
)
nested table projects store as projects_nt;
```

Table EMP contains PROJECTS, which contains a table of PROJECTTABLE type for each row.

NESTED TABLE option is required as we have a nested table column in the table. NESTED TABLE clause specifies the name of the table in which Oracle stores the data of the nested table. In this example PROJECTS_NT is created by Oracle and maintained by Oracle. It contains the data of PROJECTS column.

Note: *The data of VARRAY is stored as part of the row, but the data of nested table is stored out-of-row in the table whose name is given in NESTED TABLE option.*



The following INSERT command inserts a row into EMP table. It stores two rows into nested table.

```
insert into emp
 values(1,'Ellison',
           projecttable
           (
               project_type('Telephone Billing','System Analyst'),
               project_type('Housing Loans','Oracle DBA')
           )
       );
```

As you observed, the way we insert row into nested table and VARRAY is same. However there are many differences when it comes to data retrieval and updation.

Note: *If the table constructor is called without any parameter then an empty table is created. An empty table is not equal to NULL.*

Using DML commands with Nested Table

DML commands can treat a nested table as one of the following.

- As a table and deal with individual rows of the table.
- As an atomic value.

DML treating inner table as atomic value

The following DML commands operate on outer table treating inner table as a single value.

The following PL/SQL block will assign the projects of employee 1 to employee 3.

```
declare
     pl   projecttable;
begin
     select projects into  pl
     from   emp
     where  empno = 1;

     update emp set projects = pl
     where  empno = 3;

     commit;
end;
```

```
The following DELETE command will delete the details of where PROJECTS column
is NULL.

delete from emp where projects is null;
```

DML treating nested table as table

In the previous section we have seen how to manipulate inner table by treating it as an atomic value. Now in this section we will see how to manipulate inner table as a table.

To manipulate the inner table we have to use a new operator TABLE. This operator takes a single operand that is of type nested table or varray.

The following command displays the details of projects of employee 1.

```
select * from table ( select projects from emp where empno =1);

NAME                                               ROLE
-------------------------------------------------- --------------------
Telephone Billing                                  System Analyst
Housing Loans                                      Oracle DBA
```

Subquery retrieves PROJECTS column from EMP table. TABLE operator expects a Nested Table or VARRAY column. It then supplies the data of the column in the form a rows to outer query, which displays the data as it is.

In order to displays the names of the projects in which employee 1 is Oracle DBA:

```
select name from table ( select projects from emp where empno =1 )
where  role = 'Oracle DBA';

NAME
--------------------------------------------------
Housing Loans
```

It is also possible to insert, delete and update individual rows of nested table. Remember that is not permitted for VARRAY.

The following INSERT command adds a new row to projects table of employee 1.

```
insert into table (select projects from  emp where empno = 1)
values ('BillsOnline.com', 'Web Developer');
```

Similarly, it is possible to change the role of employee 1 in project Telephone Billing to Project Leader.

```
update table ( select projects from emp where empno = 1)
  Set role = 'Project Leader'
where name= 'Telephone Billing';
```

To delete the project BillsOnline.com from employee 1, enter:

```
delete from table ( select projects from emp where empno = 1)
where  name = 'BillsOnline.com'
```

Collection Methods

Oracle provides a set of methods that can be used with collections. These methods can be used only in PL/SQL and not in SQL.

The general syntax of these methods is:

```
collection_name.method_name[(parameters)]
```

Collection_name	Is the name of the collection object
Method_name	Is one of the methods listed in the table below
Parameters	Are the parameters that are to be sent to method (if required).

The following is the list of collection methods and their meaning.

Method	Meaning
EXISTS(n)	Returns true if n^{th} element is existing in the collection.
COUNT	Returns the number of elements that a collection currently contains.
LIMIT	For Nested table it returns NULL. For VARRAY it returns the maximum number of elements specified.
FIRST	Returns the smallest index of the collection. If collection is empty then return NULL. For VARRAY it always returns 1. But for nested table, it may return 1 or if first item is deleted then it will be more than 1.

LAST	Same as FIRST, but returns largest index. For VARRAY LAST and COUNT are same but for Nested Tables, it may be more than COUNT, if any items are deleted from Nested table.
PRIOR(n)	Returns the index number that precedes the given index. If no index is available then it returns NULL. This method ignores null values.
NEXT(n)	Returns the index number that follows the given index. If no index is available then it returns NULL. This method ignores null values. PRIOR and NEXT are useful to traverse a nested table in which some items are deleted.
EXTEND	Appends one null element to collection.
EXTEND(n)	Appends specified number of items.
TRIM(n)	Removes one or more elements from the end of the collection.
DELETE	Removes all elements from the collection.
DELETE(n)	Removes n^{th} elements.
DELETE(m,n)	Removes elements between m and n.

The following examples will show you how to use collection methods on the collections.

```
declare
-- declare a variable of  projecttable type
pl projecttable  := projecttable ( project_type('payroll','designer'));

-- procedure to display the values of the table
procedure displist(pl projecttable)
is
begin
   dbms_output.put_line('Available items');
   -- set the loop to varray from first element to last element
   for i in  pl.first..pl.last
   loop
        if  pl.exists(i) then  -- if item exists then display
               dbms_output.put_line( i || ' - ' ||  pl(i).name);
        end if;
   end loop;
end;

begin  -- beginning of the main block
   displist(pl);

   -- add two new elements to the collection
   pl.extend(2);

   -- set values for two new elements
   pl(2) := project_type('inventory','network adm');
   pl(3) := project_type('law.com','web developer');

   displist(pl);

   pl.delete(2);  -- delete second  item

   displist(pl);
   pl.trim(1);   -- remove the last item
   displist(pl);
end;
```

Summary

A collection is a set of value of same type. Oracle provides VARRAYS, Index-by tables and nested tables.

VARRAY (variable-size array) is used to an array that contains a maximum limit and contains varying number of elements. Oracle doesn't provide much flexibility on VARRAYS. For instance, it is not possible to manipulate individual elements of VARRAY.

Nested table is a table within another table It allows better control on the elements of the table. The data in the nested table is not stored as part of the main table and instead stored separately in a table created by Oracle.

TABLE operator is used to perform data manipulation on individual rows of nested table. It takes a column of nested table or VARRAY type and allows you to treat that as a collection of rows.

Collection methods are used to provide information and manage collections in PL/SQL. They cannot be used in SQL but a collection can be changed by these methods and the result can be put back to table.

Exercises

1. _____ method is used to remove an item of the collection from the given position.
2. _____ is the command used to VARRAY type.
3. Is it possible to use VARRAY column as operand of TABLE operator.
4. Is it possible for COUNT and LAST methods to return different values? If so why.
5. What is the purpose of NESTED TABLE clause in CREATE TABLE command?
6. What are the difference between VARRAY and Nested Table?
7. Create a table to store the details of STUDENTS along with the marks obtained by students in 10 subjects. Assume that the subjects are having fixed order and may vary from 5 to 10.
8. Write a function to take students number and subject number and return marks.