Oracle PL/SQL - How to create a simple array variable?

Ask Question

109

41

I'd like to create an in-memory array variable that can be used in my PL/SQL code. I can't find any collections in Oracle PL/SQL that uses pure memory, they all seem to be associated with tables. I'm looking to do something like this in my PL/SQL (C# syntax):

```
string[] arrayvalues = new string[3] {"Matt", "Joanne", "Robert"};
```

Edit: Oracle: 9i

oracle plsql oracle9i

edited Aug 10 '11 at 16:07

user272735
7,699 6 47 76

asked Aug 10 '11 at 14:41

contactmatt
8,085 29 100 153

2 See: PL/SQL Collections and Records – user272735 Aug 10 '11 at 14:51

1 The "table" reference tends to be a hangover from the old PL/SQL tables naming. VARRAYs, Associative Arrays and Declared nested tables are all in-memory array types. – Ollie Aug 10 '11 at 16:00

read this link orafaq.com/wiki/VARRAY and dba-oracle.com/tips_oracle_varray.htm – zloctb Jul 29 '15 at 5:45

4 Answers

212

You can use VARRAY for a fixed-size array:

```
declare
   type array_t is varray(3) of varch
   array array_t := array_t('Matt', '
begin
   for i in 1..array.count loop
       dbms_output.put_line(array(i))
```

```
   end loop;
end;
```

Or TABLE for an unbounded array:

```
...
   type array_t is table of varchar2(
...
```

The word "table" here has nothing to do with database tables, confusingly. Both methods create in-memory arrays.

With either of these you need to both initialise and extend the collection before adding elements:

```
declare
   type array_t is varray(3) of varch
   array array_t := array_t(); -- Ini
begin
   for i in 1..3 loop
      array.extend(); -- Extend it
      array(i) := 'x';
   end loop;
end;
```

The first index is 1 not 0.

edited Jun 15 '16 at 16:15

answered Aug 10 '11 at 14:48

Tony Andrews
108k 17 190 236

50 "confusingly" just about sums up Oracle – m.edmondson Nov 17 '15 at 14:43

1 @Abdul See updated answer. – Tony Andrews Jun 15 '16 at 16:15

2 @Abdul, no it doesn't. I never use VARRAYs normally but when testing the above code I checked what happens if you try to extend a `varray(3)` 4 times - you get a "subscript out of limit" error. – Tony Andrews Jun 15 '16 at 16:29

2 Wish I coud up vote this answer multiple times @TonyAndrews since you covered the `array.extend()` . Every where I looked did not show this and it was the most important part to being able to add more than one item (from my understanding of it, still new to arrays in SQL). – Jonathan Van Dam Jul 7 '17 at 21:17

1 i logged in to vote +1 – yılmaz Jul 17 '18 at 16:50

57

You could just declare a DBMS_SQL.VARCHAR2_TABLE to hold an in-memory variable length array indexed by a BINARY_INTEGER:

```
DECLARE
   name_array dbms_sql.varchar2_table;
BEGIN
   name_array(1) := 'Tim';
   name_array(2) := 'Daisy';
   name_array(3) := 'Mike';
   name_array(4) := 'Marsha';
   --
   FOR i IN name_array.FIRST .. name_a
   LOOP
      -- Do something
   END LOOP;
END;
```

You could use an associative array (used to be called PL/SQL tables) as they are an in-memory array.

```
DECLARE
   TYPE employee_arraytype IS TABLE OF
        INDEX BY PLS_INTEGER;
   employee_array employee_arraytype;
BEGIN
   SELECT *
     BULK COLLECT INTO employee_array
     FROM employee
    WHERE department = 10;
   --
   FOR i IN employee_array.FIRST .. em
   LOOP
      -- Do something
   END LOOP;
END;
```

The associative array can hold any make up of record types.

Hope it helps, Ollie.

answered Aug 10 '11 at 15:58

Ollie
14.1k 6 36 55

4 This is in addition to Tony's answer above which is a good answer... – Ollie Aug 10 '11 at 16:02

15 The iteration condition raises `VALUE_ERROR` when the collection is empty. I would suggest to rather use `FOR i IN 1 .. employee_array.COUNT` in this case – unziberla Jul 22 '14 at 14:14

j-chomel's version (stackoverflow.com/a/40579334/1915

920) based on `sys.odcivarchar2list` below has the advantage, that you also have a constructor at hand, e.g. for function param default initialization: `sys.odcivarchar2list('val1','val2')` – Andreas Dietrich Jun 28 '18 at 5:29

11

Another solution is to use an Oracle Collection as a Hashmap:

```
declare
-- create a type for your "Array" - it
  type hash_map is table of varchar2(1
  my_hmap hash_map ;
-- i will be your iterator: it must be
  i varchar2(30);
begin
  my_hmap('a') := 'apple';
  my_hmap('b') := 'box';
  my_hmap('c') := 'crow';
-- then how you use it:

  dbms_output.put_line (my_hmap('c'))

-- or to loop on every element - it's
  i := my_hmap.FIRST;

  while (i is not null)  loop
    dbms_output.put_line(my_hmap(i));
    i := my_hmap.NEXT(i);
  end loop;

end;
```

answered Apr 10 '17 at 9:14

J. Chomel
6,074 13 32 51

9

You can also use an ***oracle defined collection***

```
DECLARE
  arrayvalues sys.odcivarchar2list;
BEGIN
  arrayvalues := sys.odcivarchar2list(
  FOR x IN ( SELECT m.column_value m_v
               FROM table(arrayvalues)
  LOOP
    dbms_output.put_line (x.m_value||'
  END LOOP;
END;
```

I would use in-memory array. But with the `.COUNT` improvement suggested by uziberia:

```
DECLARE
  TYPE t_people IS TABLE OF varchar2(1
  arrayvalues t_people;
BEGIN
  SELECT *
   BULK COLLECT INTO arrayvalues
   FROM (select 'Matt' m_value from du
         select 'Joanne'       from du
         select 'Robert'       from du
    )
  ;
  --
  FOR i IN 1 .. arrayvalues.COUNT
  LOOP
    dbms_output.put_line(arrayvalues(i
  END LOOP;
END;
```

Another solution would be to use a Hashmap like @Jchomel did here.

NB:

With Oracle 12c you can even query arrays directly now!

edited Oct 17 '18 at 11:39

Thomas Flinkow
2,345 3 13 41

answered Nov 13 '16 at 22:07

Jika
302 4 13