Arrays and Strings in PL/SQL

Overview

- Arrays
    - Varray Data Structure
    - Creating Varrays
    - Examples
- Strings
    - Types
    - Declaration
    - Functions and Operators
    - Examples

Strings

- Sequence of characters with optional size specifications
- Can be variable or literal.
- A string literal is enclosed within quotation marks.

Types of Strings

- **Fixed-length strings** − size needs to be specified before hand. Later string is right-padded with spaces to the length so specified.
- **Variable-length strings** − In such strings, a maximum length up to 32,767, for the string is specified and no padding takes place.
- **Character large objects (CLOBs)** − These are variable-length strings that can be up to 128 terabytes.

Declaring Strings

- Oracle database provides numerous string data types, such as CHAR, NCHAR, VARCHAR2, NVARCHAR2, CLOB, and NCLOB.
- Ex:
- **DECLARE**

```
DECLARE
   name varchar2(20);
   company varchar2(30);
   introduction clob;
   choice char(1);
```

String Functions and Operators

PL/SQL offers the concatenation operator **‘||’** for joining two strings.

ASCII(x): Returns the ASCII value of the character x.

CHR(x): Returns the character with the ASCII value of x.

CONCAT(x, y): Concatenates the strings x and y and returns the appended string.

LENGTH(x): Returns the number of characters in x.

LOWER(x): Converts the letters in x to lowercase and returns that string.

UPPER(x): Converts the letters in x to uppercase and returns that string.

INSTR(x, find_string [, start] [, occurrence]): Searches for find_string in x and returns the position at which it occurs.

SUBSTR(x, start [, length]): Returns a substring of x that begins at the position specified by start. An optional length for the substring may be supplied.

TRIM([trim_char FROM) x): Trims characters from the left and right of x.

```
DECLARE
   greetings varchar2(11) := 'hello world';
BEGIN
   dbms_output.put_line(UPPER(greetings));                       --HELLO WORLD
   dbms_output.put_line(LOWER(greetings));                       --hello world
   dbms_output.put_line(INITCAP(greetings));                     --Hello World
   dbms_output.put_line ( SUBSTR (greetings, 1, 1));             --h
   dbms_output.put_line ( SUBSTR (greetings, -1, 1));            --d
   dbms_output.put_line ( SUBSTR (greetings, 7, 5));             --world
   dbms_output.put_line ( SUBSTR (greetings, 2));                --ello world
   dbms_output.put_line ( INSTR (greetings, 'e'));               --2
END;
/
```

Arrays

- The PL/SQL programming language provides a data structure called the **VARRAY.**
- A varray is used to store an ordered collection of data of the same type.
- All varrays consist of contiguous memory locations.
- starting index for varrays is always 1.
- The lowest address corresponds to the first element and the highest address to the last element.
- 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

- At schema level:

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

- In a PL/SQL block:

  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.

```
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;
/
```

Elements of a varray could also be a %ROWTYPE of any database table or %TYPE of any database table field.

Select * from customers;

```
+----+----------+-----+-----------+----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY  |
+----+----------+-----+-----------+----------+----------+
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;
/
```