Exam Paper

16 June 2020 20:34

Quiz

A PL/SQL block must consist of the following three sections:

- A Declarative section, which begins with the keyword DECLARE and ends when the executable section starts.
- An Executable section, which begins with the keyword BEGIN and ends with END.
- An Exception handling section, which begins with the keyword EXCEPTION and is nested within the executable section.

a. True
b. False

A PL/SQL block consists of three sections:

- **Declarative (optional):** The optional declarative section begins with the keyword `DECLARE` and ends when the executable section starts.
- **Executable (required):** The required executable section begins with the keyword `BEGIN` and ends with `END`. This section essentially needs to have at least one statement. Observe that `END` is terminated with a semicolon. The executable section of a PL/SQL block can, in turn, include any number of PL/SQL blocks.
- **Exception handling (optional):** The optional exception section is nested within the executable section. This section begins with the keyword `EXCEPTION`.

```
SET  SERVER OUTPUT ON

DECLARE
    v_event VARCHAR2(15);
BEGIN
   v_event  := q'!Father's day!';
  DBMS_OUTPUT.PUT_LINE('3rd Sunday in June is :
   '|| v_event );
   v_event  := q'[Mother's day]';
  DBMS_OUTPUT.PUT_LINE('2nd Sunday in May is :
   '|| v_event );
END;
```

Base Scalar Data Types

- CHAR [(maximum_length)]
- VARCHAR2 (maximum_length)
- NUMBER [(precision, scale)]
- BINARY_INTEGER
- PLS_INTEGER
- BOOLEAN
- BINARY_FLOAT
- BINARY_DOUBLE
- DATE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH

- INTERVAL DAY TO SECOND

```
DECLARE
  v_emp_job                VARCHAR2(9);
  v_count_loop            BINARY_INTEGER := 0;
  v_dept_total_sal    NUMBER(9,2) := 0;
  v_orderdate            DATE := SYSDATE + 7;
  c_tax_rate             CONSTANT NUMBER(3,2) := 8.25;
  v_valid                BOOLEAN NOT NULL := TRUE;
  ...
```

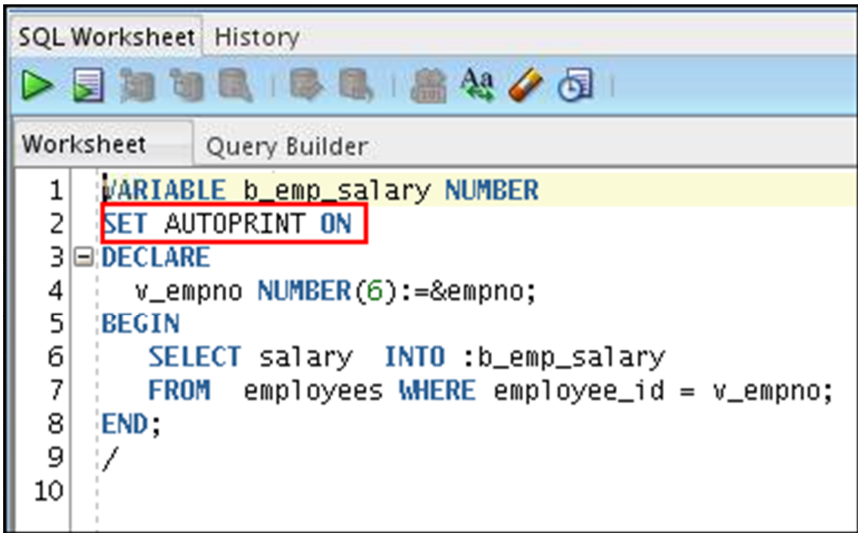
Declaring Variables with the %TYPE Attribute

```
v_emp_lname      employees.last_name%TYPE;

v_balance        NUMBER(7,2);
v_min_balance    v_balance%TYPE := 1000;
```

Referencing Bind Variables

```
VARIABLE b_emp_salary NUMBER
BEGIN
   SELECT salary  INTO :b_emp_salary
   FROM  employees WHERE employee_id = 178;
END;
/
PRINT b_emp_salary
SELECT first_name, last_name
FROM employees
WHERE salary=:b_emp_salary;
```



Quiz

The %TYPE attribute:

a. Is used to declare a variable according to a database column definition (True)
b. Is used to declare a variable according to a collection of columns in a database table or view.(False)
c. Is used to declare a variable according to the definition of another declared variable.(True)
ç. Is prefixed with the database table and column name or the name of the declared variable.(True)

The %TYPE Attribute

PL/SQL variables are usually declared to hold and manipulate data stored in a database. When you declare PL/SQL variables to hold column values, you must ensure that the variable is of the correct data type and precision. If it is not, a PL/SQL error occurs during execution. If you have to design large subprograms, this can be time consuming and error prone.

Rather than hard-coding the data type and precision of a variable, you can use the %TYPE attribute to declare a variable according to another previously declared variable or database column. The %TYPE attribute is most often used when the value stored in the variable is derived from a table in the database. When you use the %TYPE attribute to declare a variable, you should prefix it with the database table and column name. If you refer to a previously declared variable, prefix the variable name of the previously declared variable to the variable being declared. The benefit of %TYPE is that you do not have to change the variable if the column is altered. Also, if the variable is used in any calculations, you need not worry about its precision.

The %ROWTYPE Attribute

The %ROWTYPE attribute is used to declare a record that can hold an entire row of a table or view. You learn about this attribute in the lesson titled “Working with Composite Data Types.”

```
DECLARE
```

```
 v_fname VARCHAR2(25);
BEGIN
 SELECT first_name INTO v_fname
 FROM employees WHERE employee_id=200;
 DBMS_OUTPUT.PUT_LINE(' First Name is : '||v_fname);
END;
```

Inserting Data: Example

```
BEGIN
 INSERT INTO employees
  (employee_id, first_name, last_name, email,
   hire_date, job_id, salary)
   VALUES(employees_seq.NEXTVAL, 'Ruth', 'Cores',
   'RCORES',CURRENT_DATE, 'AD_ASST', 4000);
END;
/
```

Updating Data: Example

```
DECLARE
    sal_increase   employees.salary%TYPE := 800;
BEGIN
UPDATE    employees
    SET      salary = salary + sal_increase
    WHERE    job_id = 'ST_CLERK';
END;
/
```

Deleting Data: Example

```
DECLARE
  deptno   employees.department_id%TYPE := 10;
BEGIN
  DELETE FROM   employees
  WHERE  department_id = deptno;
END;
/
```

Merging Rows

```
BEGIN
MERGE INTO copy_emp c
```

```
    USING employees e
    ON (e.employee_id = c.empno)
  WHEN MATCHED THEN
    UPDATE SET
      c.first_name     = e.first_name,
      c.last_name      = e.last_name,
      c.email          = e.email,
      . . .
  WHEN NOT MATCHED THEN
    INSERT VALUES(e.employee_id, e.first_name,
e.last_name,
          . . .,e.department_id);
END;
/
```

SQL Cursor Attributes for Implicit Cursors

```
DECLARE
  v_rows_deleted VARCHAR2(30)
  v_empno employees.employee_id%TYPE := 176;
BEGIN
  DELETE FROM  employees
  WHERE employee_id = v_empno;
  v_rows_deleted := (SQL%ROWCOUNT ||
                     ' row deleted.');
  DBMS_OUTPUT.PUT_LINE (v_rows_deleted);
END;
```

Quiz

When using the `SELECT` statement in PL/SQL, the `INTO` clause is required and queries can return one or more rows.

a. True
b. False

IF

```
DECLARE
  v_myage number:=31;
BEGIN
  IF v_myage  < 11 THEN
       DBMS_OUTPUT.PUT_LINE(' I am a child ');
    ELSIF v_myage  < 20 THEN
       DBMS_OUTPUT.PUT_LINE(' I am young ');
    ELSIF v_myage  < 30 THEN
       DBMS_OUTPUT.PUT_LINE(' I am in my
twenties');
    ELSIF v_myage  < 40 THEN
       DBMS_OUTPUT.PUT_LINE(' I am in my
thirties');
    ELSE
       DBMS_OUTPUT.PUT_LINE(' I am always young
');
  END IF;
END;
```

CASE Expressions: Example

SET VERIFY OFF // değişkeni içeren her satırın görüntülerinden önce ve sonra görüntülenip görüntülenmeyeceğini denetler.

```
DECLARE
   v_grade  CHAR(1) := UPPER('&grade');
   v_appraisal VARCHAR2(20);
BEGIN
   v_appraisal := CASE v_grade
         WHEN 'A' THEN 'Excellent'
         WHEN 'B' THEN 'Very Good'
         WHEN 'C' THEN 'Good'
         ELSE 'No such grade'
      END;
DBMS_OUTPUT.PUT_LINE ('Grade: '|| v_grade  ||
                      ' Appraisal ' || v_appraisal);
END;
```

Searched CASE Expressions

```
DECLARE
   v_grade  CHAR(1) := UPPER('&grade');
   v_appraisal VARCHAR2(20);
BEGIN
    v_appraisal := CASE
         WHEN v_grade  = 'A' THEN 'Excellent'
         WHEN v_grade  IN ('B','C') THEN 'Good'
         ELSE 'No such grade'
     END;
   DBMS_OUTPUT.PUT_LINE ('Grade: '|| v_grade  ||
                ' Appraisal ' || v_appraisal);
END;
```

CASE Statement

```
DECLARE
   v_deptid NUMBER;
   v_deptname VARCHAR2(20);
   v_emps NUMBER;
   v_mngid NUMBER:= 108;
BEGIN
  CASE  v_mngid
   WHEN  108 THEN
    SELECT department_id, department_name
     INTO v_deptid, v_deptname FROM departments
     WHERE manager_id=108;
    SELECT count(*) INTO v_emps FROM employees
     WHERE department_id=v_deptid;
   WHEN  200 THEN
    ...
 END CASE;
DBMS_OUTPUT.PUT_LINE ('You are working in the
'|| v_deptname||
' department. There are '||v_emps ||' employees
in this
department');
END
```

Basic Loop: Example

```
DECLARE
  v_countryid    locations.country_id%TYPE :=
'CA';
  v_loc_id       locations.location_id%TYPE;
  v_counter          NUMBER(2) := 1;
  v_new_city     locations.city%TYPE :=
'Montreal';
BEGIN
  SELECT MAX(location_id) INTO v_loc_id FROM
locations
  WHERE country_id = v_countryid;
  LOOP
    INSERT INTO locations(location_id, city,
country_id)
    VALUES((v_loc_id + v_counter), v_new_city,
v_countryid);
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 3;
  END LOOP;
```

```
END;
```

WHILE Loops: Example

```
DECLARE
  v_countryid   locations.country_id%TYPE :=
'CA';
  v_loc_id      locations.location_id%TYPE;
  v_new_city    locations.city%TYPE :=
'Montreal';
  v_counter     NUMBER := 1;
BEGIN
  SELECT MAX(location_id) INTO v_loc_id FROM
locations
  WHERE country_id = v_countryid;
  WHILE v_counter <= 3 LOOP
    INSERT INTO locations(location_id, city,
country_id)
    VALUES((v_loc_id + v_counter), v_new_city,
v_countryid);
    v_counter := v_counter + 1;
  END LOOP;
END;
```

FOR Loops: Example

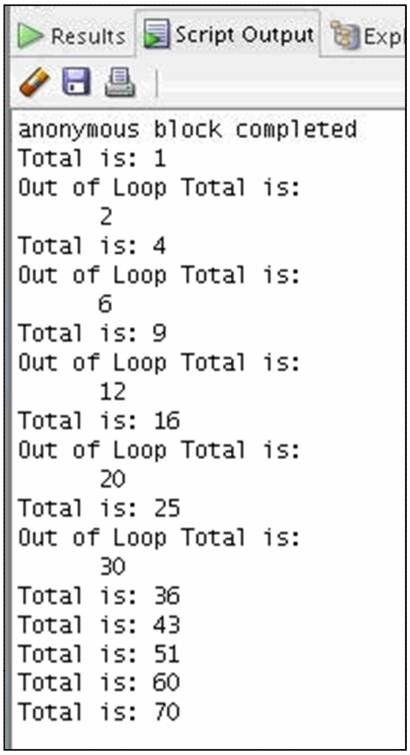
```
DECLARE
  v_countryid   locations.country_id%TYPE :=
'CA';
  v_loc_id      locations.location_id%TYPE;
  v_new_city    locations.city%TYPE :=
'Montreal';
BEGIN
  SELECT MAX(location_id) INTO v_loc_id
    FROM locations
    WHERE country_id = v_countryid;
  FOR i IN 1..3 LOOP
    INSERT INTO locations(location_id, city,
country_id)
    VALUES((v_loc_id + i), v_new_city,
v_countryid );
  END LOOP;
END;
/
```

Nested Loops and Labels: Example

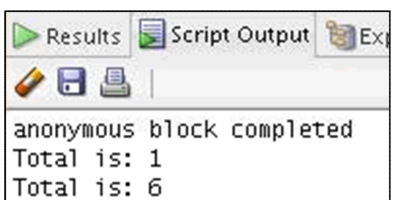
```
BEGIN
  <<Outer_loop>>
  LOOP
    v_counter := v_counter+1;
  EXIT WHEN v_counter>10;
    <<Inner_loop>>
    LOOP
      ...
      EXIT Outer_loop WHEN total_done = 'YES';
      -- Leave both loops
      EXIT WHEN inner_done = 'YES';
      -- Leave inner loop only
      ...
    END LOOP Inner_loop;
    ...
  END LOOP Outer_loop;
END;
```

PL/SQL CONTINUE Statement: Example 1

```
DECLARE
  v_total SIMPLE_INTEGER := 0;
BEGIN
  FOR i IN 1..10 LOOP
    v_total := v_total + i;
    dbms_output.put_line
     ('Total is: '|| v_total);
    CONTINUE WHEN i > 5;
    v_total := v_total + i;
    dbms_output.put_line
     ('Out of Loop Total is:
      '|| v_total);
  END LOOP;
END;
/
```




```
DECLARE
 v_total NUMBER := 0;
BEGIN
 <<BeforeTopLoop>>
 FOR i IN 1..10 LOOP
   v_total := v_total + 1;
   dbms_output.put_line
```



```
 FOR i IN 1..10 LOOP
   v_total := v_total + 1;
   dbms_output.put_line
     ('Total is: ' || v_total);
   FOR j IN 1..10 LOOP
     CONTINUE BeforeTopLoop WHEN i + j > 5;
     v_total := v_total + 1;
   END LOOP;
 END LOOP;
END two_loop;
```

```
anonymous block completed
Total is: 1
Total is: 6
Total is: 10
Total is: 13
Total is: 15
Total is: 16
Total is: 17
Total is: 18
Total is: 19
Total is: 20
```

Quiz

There are three types of loops: basic, `FOR`, and `WHILE`.

a. True

b. False