Database Programming with PL/SQL

2-1

Using Variables in PL/SQL
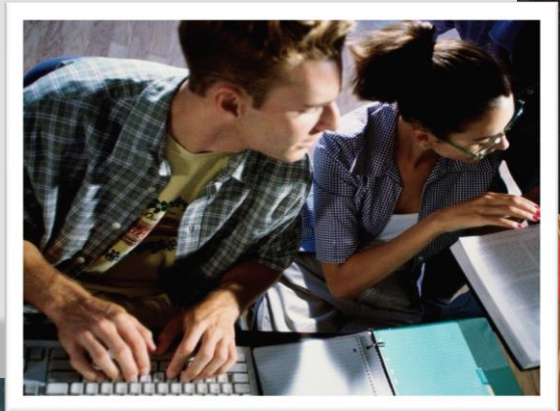
ORACLE
Academy

Objectives

- This lesson covers the following objectives:
  - List the uses of variables in PL/SQL
  - Identify the syntax for variables in PL/SQL
  - Declare and initialize variables in PL/SQL
  - Assign new values to variables in PL/SQL
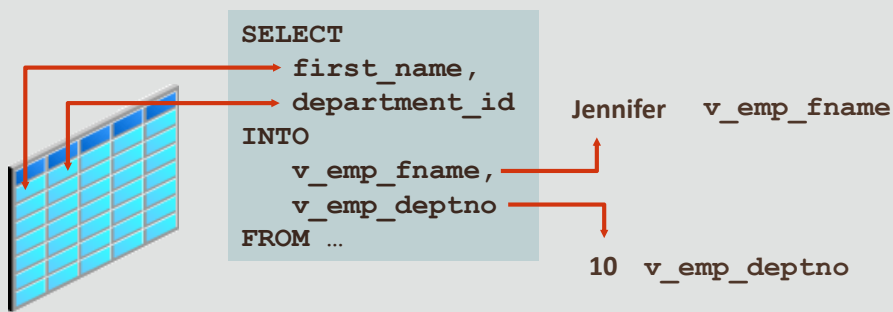
ORACLE
Academy

3

3

Purpose

- You use variables to store and manipulate data
- In this lesson, you learn how to declare and initialize variables in the declarative section of a PL/SQL block
- With PL/SQL, you can declare variables and then use them in both SQL and procedural statements
- Variables can be thought of as storage containers that hold something until it is needed

4

4

Use of Variables

- Variables are expressions that stand for something of value (in the equation $x + y = 45$, x and y are variables that stand for two numbers that add up to 45)

- When defining a variable in a PL/SQL declaration section, you label a memory location, assign a datatype, and, if needed, assign a starting value for the variable

- A variable can represent a number, character string, boolean (true/false value), or other datatypes

- Throughout the PL/SQL code, variable values can be changed by the assignment operator (:=)

ORACLE
Academy

5

Use of Variables

- Use variables for:
  - Temporary storage of data
  - Manipulation of stored values
  - Reusability


```
SELECT
    first_name,
    department_id              Jennifer   v_emp_fname
INTO
    v_emp_fname,
    v_emp_deptno
FROM ...
                               10  v_emp_deptno
```

Variables are mainly used for the storage of data and manipulation of those stored values. Consider the SQL statement shown in the slide. The statement is retrieving the first_name and department_id from the table. If you have to manipulate the first_name or the department_id, then you have to store the retrieved value. Variables are used to temporarily store the values. You can use the value stored in these variables for processing or manipulating the data.

Handling Variables in PL/SQL

- Variables are:
  - Declared and initialized in the declarative section
  - Used and assigned new values in the executable section
- Variables can be:
  - Passed as parameters to PL/SQL subprograms
  - Assigned to hold the output of a PL/SQL subprogram

Declaring Variables

- All PL/SQL variables must be declared in the declaration section before referencing them in the PL/SQL block

- The purpose of a declaration is to allocate storage space for a value, specify its data type, and name the storage location so that you can reference it

- You can declare variables in the declarative part of any PL/SQL block, subprogram, or package

A variable is a name or label that points to a value stored at a particular location in the computer's memory.

Declaring Variables: Syntax

- The identifier is the name of the variable
- It and the datatype are the minimum elements required

```
identifier [CONSTANT] datatype [NOT NULL]
   [:= expr | DEFAULT expr];
```

In addition to variables, you also can declare cursors and exceptions in the declarative section. You will learn how to declare cursors and exceptions later in the course.

Initializing Variables

- Variables are assigned a memory location inside the DECLARE section
- Variables can be assigned a value at the same time
- This process is called initialization
- The value in a variable also can be modified by reinitializing the variable in the executable section

```
DECLARE
   v_counter INTEGER := 0;
BEGIN
   v_counter := v_counter + 1;
   DBMS_OUTPUT.PUT_LINE(v_counter);
END;
```

10

Declaring and Initializing Variables Example 1

- This example shows the declaration of several variables of various datatypes using syntax that sets constraints, defaults, and initial values
- You will learn more about the different syntax as the course progresses

```
DECLARE
   fam_birthdate DATE;
   fam_size          NUMBER(2) NOT NULL := 10;
   fam_location      VARCHAR2(13) := 'Florida';
   fam_bank          CONSTANT NUMBER := 50000;
   fam_population    INTEGER;
   fam_name          VARCHAR2(20) DEFAULT 'Roberts';
   fam_party_size    CONSTANT PLS_INTEGER := 20;
```

ORACLE
Academy

PLSQL 2-1
Using Variables in PL/SQL

11

Declaring and Initializing Variables Example 2

- This example shows the convention of beginning variables with v_ and variables that are configured as constants with c_

```
DECLARE
   v_emp_hiredate      DATE;
   v_emp_deptno      NUMBER(2) NOT NULL := 10;
   v_location        VARCHAR2(13) := 'Atlanta';
   c_comm            CONSTANT NUMBER := 1400;
   v_population      INTEGER;
   v_book_type          VARCHAR2(20) DEFAULT 'fiction';
   v_artist_name     VARCHAR2(50);
   v_firstname          VARCHAR2(20):='Rajiv';
   v_lastname           VARCHAR2(20) DEFAULT 'Kumar';
   c_display_no      CONSTANT PLS_INTEGER := 20;
```

- The defining of data types and data structures using a standard in a programming language is a significant aid to readability

ORACLE
Academy

PLSQL 2-1
Using Variables in PL/SQL

12

Assigning Values in the Executable Section Example 1

• After a variable is declared, you can use it in the executable section of a PL/SQL block

• For example, in the following block, the variable v_myname is declared in the declarative section of the block

```
DECLARE
  v_myname  VARCHAR2(20);
BEGIN
  DBMS_OUTPUT.PUT_LINE('My name is: '||v_myname);
  v_myname := 'John';
  DBMS_OUTPUT.PUT_LINE('My name is: '||v_myname);
END;
```

ORACLE
Academy

Assigning Values in the Executable Section Example 1

- You can access this variable in the executable section of the same block
- What do you think the block will print?

```
DECLARE
   v_myname   VARCHAR2(20);
BEGIN
   DBMS_OUTPUT.PUT_LINE('My name is: '||v_myname);
   v_myname := 'John';
   DBMS_OUTPUT.PUT_LINE('My name is: '||v_myname);
END;
```

The code results in the following output:

My name is:

My name is: John

Statement processed.

Assigning Values in the Executable Section Example 1

- In this example, the variable has no value when the first PUT_LINE is executed, but then the value John is assigned to the variable before the second PUT_LINE

- The value of the variable is then concatenated with the string My name is:

- The output is:

```
My name is:
My name is:   John

Statement process.
```

- A non-initialized variable contains a NULL value until a non-null value is explicitly assigned to it

ORACLE
Academy

Assigning Values in the Executable Section Example 2

- In this block, the variable v_myname is declared and initialized
- It begins with the value John, but the value is then manipulated in the executable section of the block

```
DECLARE
   v_myname   VARCHAR2(20):= 'John';
BEGIN
   v_myname   := 'Steven';
   DBMS_OUTPUT.PUT_LINE('My name is: '|| v_myname);
END;
```

- The output is:

```
My name is:  Steven

Statement processed.
```

ORACLE
Academy

Passing Variables as Parameters to PL/SQL Subprograms

- Parameters are values passed to a subprogram by the user or by another program

- The subprogram uses the value in the parameter when it runs

- The subprogram may also return a parameter to the calling environment. In PL/SQL, subprograms are generally known as procedures or functions

- You will learn more about procedures and functions as the course progresses

ORACLE
Academy

PLSQL 2-1
Using Variables in PL/SQL

17

Passing Variables as Parameters to PL/SQL Subprograms

- In the following example, the parameter v_date is being passed to the procedure PUT_LINE, which is part of the package DBMS_OUTPUT

```
DECLARE
   v_date     VARCHAR2(30);
BEGIN
   SELECT TO_CHAR(SYSDATE) INTO v_date FROM DUAL;
   DBMS_OUTPUT.PUT_LINE(v_date);
END;
```

ORACLE
Academy

PLSQL 2-1
Using Variables in PL/SQL

18

The syntax in this slide will be explained in more detail in later lessons. For now, you simply need to understand that in PL/SQL, a variable can be passed from one program to another program, and we call these variables parameters.

Assigning Variables to PL/SQL Subprogram Output

- You can use variables to hold values that are returned by a function (see function definition below and a call to this function on the following slide)

```
CREATE FUNCTION num_characters (p_string IN VARCHAR2)
RETURN INTEGER IS
   v_num_characters INTEGER;
BEGIN
   SELECT LENGTH(p_string) INTO v_num_characters
      FROM DUAL;
   RETURN v_num_characters;
END;
```

- The concept, creation, and use of functions will be covered later in this course

ORACLE
Academy

Assigning Variables to PL/SQL Subprogram Output

- In the call to the function num_characters, the value returned by the function will be stored in the variable v_length_of_string

```
DECLARE
   v_length_of_string   INTEGER;
BEGIN
   v_length_of_string := num_characters('Oracle
Corporation');
   DBMS_OUTPUT.PUT_LINE(v_length_of_string);
END;
```

Terminology

- Key terms used in this lesson included:
  - Parameters
  - Variables

- Parameters – values passed to a program by a user or by another program
- Variables – used for temporary storage and manipulation of data

Summary

- In this lesson, you should have learned how to:
  - List the uses of variables in PL/SQL
  - Identify the syntax for variables in PL/SQL
  - Declare and initialize variables in PL/SQL
  - Assign new values to variables in PL/SQL

PLSQL 2-1
Using Variables in PL/SQL

22