Capgemini

PLSQL

Lesson 01: PLSQL Basics, Datatypes

Lesson Objectives

To understand the following topics:

- Introduction and Need for PLSQL
- Datatypes
- Scalar
- Composite Variables

1.1: Need for PL/SQL
Overview

PL/SQL is a procedural extension to SQL.

The “data manipulation” capabilities of “SQL” are combined with the “processing capabilities” of a “procedural language”.

PL/SQL provides features like conditional execution, looping and branching.

PL/SQL supports subroutines, as well.

PL/SQL program is of block type, which can be “sequential” or “nested” (one inside the other).

1.1: Introduction to PL/SQL
Features of PL/SQL

PL/SQL provides the following features:

- Tight Integration with SQL
- Better performance
- Several SQL statements can be bundled together into one PL/SQL block and sent to the server as a single unit.
- Standard and portable language
- Although there are a number of alternatives when it comes to writing software to run against the Oracle Database, it is easier to run highly efficient code in PL/SQL, to access the Oracle Database, than in any other language.

1.1: Introduction to PL/SQL
PL/SQL Block Structure

A PL/SQL block comprises of the following structures:

- DECLARE – Optional

Variables, cursors, user-defined exceptions

- BEGIN – Mandatory

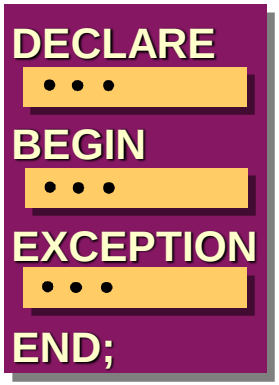
SQL statements

- PL/SQL statements

EXCEPTION – Optional

Actions to perform when errors occur

- END; – Mandatory



1.1: Introduction to PL/SQL
Block Types

There are three types of blocks in PL/SQL:

Anonymous

Named:

Procedure

Function

Anonymous

```
[DECLARE]

BEGIN
  --statements

[EXCEPTION]

END;
```

Procedure

```
PROCEDURE
name
IS

BEGIN
  --statements

[EXCEPTION]

END;
```

Function

```
FUNCTION
name
RETURN
datatype
IS
BEGIN
  --statements
  RETURN value;
[EXCEPTION]

END;
```

1.2: Data Types
Declaring PL/SQL variables

Syntax

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

Example

```
DECLARE
    v_hiredate      DATE;
    v_deptno        NUMBER(2) NOT NULL := 10;
    v_location      VARCHAR2(13) := 'Atlanta';
    c_comm          CONSTANT NUMBER := 1400;
```

Base Scalar Data Types

Base Scalar Datatypes:

- Given below is a list of Base Scalar Datatypes:
  - VARCHAR2 (maximum_length)
  - NUMBER [(precision, scale)]
  - DATE
  - CHAR [(maximum_length)]
  - LONG
  - LONG RAW
  - BOOLEAN
  - BINARY_INTEGER
  - PLS_INTEGER

1.2: Data Types
Base Scalar Data Types - Example

Here are a few examples of Base Scalar Datatypes:

```
v_job         VARCHAR2(9);
v_count       BINARY_INTEGER := 0;
v_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;
```

1.2: Data Types
Declaring Datatype with %TYPE Attribute

While using the %TYPE Attribute:

- Declare a variable according to:
- a database column definition
- another previously declared variable
- Prefix %TYPE with:
- the database table and column
- the previously declared variable name

1.2: Data Types
Declaring Datatype with %TYPE Attribute (Contd…)

Example:

```
...
v_name          staff_master.staff_name%TYPE;
v_balance       NUMBER(7,2);
v_min_balance   v_balance%TYPE := 10;
...
```

1.2: Data Types
Declaring Datatype by using %ROWTYPE

Example:

```
DECLARE
    nRecord staff_master%rowtype;
BEGIN
    SELECT * into nrecord
            FROM staff_master
            WHERE staff_code = 100001;

            UPDATE staff_master
            SET staff_sal = staff_sal + 101
            WHERE emp_code = 100001;
END;
```

1.2: Data Types
User-defined SUBTYPES

User-defined SUBTYPES:

User-defined SUBTYPES are subtypes based on an existing type.

They can be used to give an alternate name to a type.

Syntax:

```
SUBTYPE New_Type IS original_type;
```

```
SUBTYPE T_Counter IS NUMBER;
V_Counter T_Counter;
SUBTYPE T_Emp_Record IS  EMP%ROWTYPE;
```

It can be a predefined type, subtype, or %type reference.

1.2: Data Types
User-defined SUBTYPES (Contd…)

- It is illegal to constrain a subtype.

```
SUBTYPE  T_Counter  IS NUMBER(4)    -- Illegal
```

- Possible solutions:

```
V_Dummy   NUMBER(4);
SUBTYPE T_Counter   IS        V_Dummy%TYPE;
V_Counter    T_Counter ;
SUBTYPE    T_Numeric  IS      NUMBER;
V_Counter  IS  T_Numeric(5);
```

1.2: Data Types
Composite Data Types

Composite Datatypes in PL/SQL:

- Two composite datatypes are available in PL/SQL:
- records
- tables

A composite type contains components within it. A variable of a composite type contains one or more scalar variables.

1.2: Data Types
Record Data Types

Record Datatype:

- A record is a collection of individual fields that represents a row in the table.
- They are unique and each has its own name and datatype.
- The record as a whole does not have value.

Defining and declaring records:

- Define a RECORD type, then declare records of that type.
- Define in the declarative part of any block, subprogram, or package.

1.2: Data Types
Record Data Types (Contd…)

Syntax:

```
TYPE type_name IS RECORD (field_declaration [,field_
declaration] ...);
```

1.2: Data Types
Record Data Types - Example

Here is an example for declaring Record datatype:

```
DECLARE
TYPE DeptRec IS RECORD (
Dept_id
    department_master.dept_code%TYPE,
Dept_name          varchar2(15),
```

1.2: Data Types
Record Data Types - Example (Contd…)

Here is an example for declaring and using Record datatype:

```
DECLARE
      TYPE recname is RECORD
      (customer_id number,
       customer_name varchar2(20));
      var_rec recname;
BEGIN
      var_rec.customer_id:=20;
      var_rec.customer_name:='Smith';
      dbms_output.put_line(var_rec.customer_id||'
'||var_rec.customer_name);
END;
```

1.2: Data Types
Table Data Type

A PL/SQL table is:

- a one-dimensional, unbounded, sparse collection of homogeneous elements
- indexed by integers
- In technical terms, a PL/SQL table:
  - is like an array
  - is like a SQL table; yet it is not precisely the same as either of those data structures
  - is one type of collection structure
  - is PL/SQL's way of providing arrays

Table Data Type (Contd…)

Declaring a PL/SQL table:

- There are two steps to declare a PL/SQL table:
  - Declare a TABLE type.
  - Declare PL/SQL tables of that type.

```
TYPE type_name is TABLE OF
{Column_type | table.column%type} [NOT NULL]
INDEX BY BINARY_INTEGER;
```

- If the column is defined as NOT NULL, then PL/SQL table will reject NULLs.

1.2: Data Types
Table Data Type - Examples

Example 1:

- To create a PL/SQL table named as “student_table” of char column.

```
DECLARE
TYPE student_table is table of char(10)
INDEX BY BINARY_INTEGER;
```

Example 2:

- To create “student_table” based on the existing column of “student_name” of EMP table.

```
DECLARE
TYPE student_table is table of student_master.student_name%type
INDEX BY BINARY_INTEGER;
```

1.2: Data Types
Table Data Type - Examples (Contd…)

After defining type emp_table, define the PL/SQL tables of that type.

- For example:

```
Student_tab  student_table;
```

- These tables are unconstrained tables.
- You cannot initialize a PL/SQL table in its declaration.
  - For example:

```
Student_tab :=('SMITH','JONES','BLAKE');          --Illegal
```

1.2: Data Types
Referencing PL/SQL Tables

Here is an example of referencing PL/SQL tables:

```
DECLARE
    TYPE staff_table is table of
     staff_master.staff_name%type
     INDEX BY BINARY_INTEGER;
staff_tab staff_table;
BEGIN
     staff_tab(1) := 'Smith'; --update Smith's
salary
     UPDATE staff_master
     SET staff_sal = 1.1 * staff_sal
     WHERE staff_name = staff_tab(1);
END;
```

1.2: Data Types
Referencing PL/SQL Tables - Examples

- To assign values to specific rows, the following syntax is used:

```
PLSQL_table_name(primary_key_value) := PLSQL expression;
```

- From ORACLE 7.3, the PL/SQL tables allow records as their columns.

SUMMARY

In this lesson, you have learnt:

- Introduction and Need for PLSQL
- Datatypes
- Scalar
- Composite Variables

Review Question

Question 1: User-defined SUBTYPES are subtypes based on an existing type.

- True / False

Question 2: A record is a collection of individual fields that represents a row in the table.

- True/ False

Review Question

Question 3: %ROWTYPE is used to declare a variable with the same datatype as a column of a specific table.

- True / False

Question 4: PL/SQL tables use a primary key to give you array-like access to rows.

- True / False