UNIT V

PL/SQL:PL/SQL Composite Data Types: Records–Tables–Varrays. PL/SQLNamed Blocks: Procedures–Functions–Packages-Triggers – Data Dictionary Views.

Text Book: "DATABASE SYSTEMS USING ORACLE"-NILESH SHAH,2nd Edition, PHI

Prepared by Dr.P.Radha

PL/SQL:PL/SQL COMPOSITE DATA TYPES

- Composite data types are like scalar data types. Scalar data types are atomic, because they do not consist of a group.
- Composite data types, on the other hand, are groups, or “collections.”
- Examples of composite data types are RECORD, TABLE, nested TABLE, and VARRAY

PL/SQL RECORDS

- PL/SQL records are similar in structure to a row in a database table.
- A record consists of components of any scalar, PL/SQL record, or PL/SQL table type.
- These components are known as fields, and they have their own values

PL/SQL RECORDS

- The record does not have a value as a whole; instead, it enables you to access these components as a group.
- It makes your life easier by transferring the entire row into a record rather than each column into a variable separately.
- A PL/SQL record is based on a cursor, a table’s row, or a user-defined record type.

Creating a PL/SQL Record

- You create a RECORD type first, and then you declare a record with that RECORD type.
- The general syntax is

TYPE recordtypename IS RECORD

(fieldname1 datatype | variable%TYPE | table.column%TYPE | table%ROWTYPE [[NOT NULL] := | DEFAULT Expression]

[, fieldname2 . . .

, FieldName3);

recordname recordtypename;

Cont…

Referencing Fields in a Record

- A field in a record has a name that is given in the RECORD-type definition.
- You cannot reference a field by its name only; you must use the record name as a qualifier:

recordname.fieldname

employee_rec.e_sal

Cont…

Working with Records

- A record is known in the block where it is declared.
- When the block ends, the record no longer exists. You can assign values to a record from columns in a row by using the SELECT statement or the FETCH statement.
- The order of fields in a record must match the order of columns in the row. A record can be assigned to another record if both records have the same structure.

Cont…

Nested Records

- You can create a nested record by including a record into another record as a field.
- The record that contains another record as a field is called the **enclosing record**

PL/SQL TABLES

- A table, like a record, is a composite data structure in PL/SQL.
- A PL/SQL table is a single-dimensional structure with a collection of elements that store the same type of value.
- In other words, it is like an array in other programming languages.
- A table is a dynamic structure that is not constrained, whereas an array is not dynamic in most computer languages.

Declaring a PL/SQL Table

- A PL/SQL TABLE declaration is done in two steps, like a record declaration:
- Declare a PL/SQL table type with a TYPE statement.
- The structure could use any of the scalar data types.
- Declare an actual table based on the type declared in the previous step.

Cont…

The general syntax is

TYPE tabletypename IS TABLE OF

datatype | variablename%TYPE | tablename.columnname%TYPE [NOT NULL] INDEX BY BINARY_INTEGER;

Cont…

Referencing Table Elements/Rows

- The rows in a table are referenced in the same way that an element in an array is referenced.
- You cannot reference a table by its name only.
- You must use the primary key value in a pair of parentheses as its subscript or index:

tablename (primarykeyvalue)

Cont…

Assigning Values to Rows in a PL/SQL Table

You can assign values to the rows in a table in three ways:

- Direct assignment.
- Assignment in a loop.
- Aggregate assignment.

Cont…

Direct Assignment.

- You can assign a value to a row with an assignment statement, as you already learned in the previous topic.
- This is preferable if only a few assignments are to be made.
- If an entire database table's values are to be assigned to a table, however, a looping method is preferable.

Cont…

Assignment in a Loop

You can use any of the three PL/SQL loops to as- sign values to rows in a table.

Aggregate Assignment

- You can assign a table's values to another table. The data types of both tables must be compatible.
- When you assign a table's values to another table, the table receiving those values loses all its previous primary key values as well as its data column values.
- If you assign an empty table with no rows to another table with rows, the recipient table is cleared

Cont…

Built-In Table Methods

- The built-in table methods are procedures or functions that provide information about a PL/SQL table. The general syntax is

tablename.methodname [(index1 [, index2])]

TABLE OF RECORDS

- The PL/SQL table type is declared with a data type.
- The %ROWTYPE declaration attribute can be used to define the record type.
- When a table is based on a record, the record must consist of fields with scalar data types.
- The record must not contain a nested record

PL/SQL VARRAYS

- A **VARRAY** is another composite data type or collection type in PL/SQL.
- Varray stands for variable-size array. They are single-dimensional, bounded collections of elements with the same data type.
- They retain their ordering and subscripts when stored in and retrieved from a database table.
- They are similar to a PL/SQL table, and each element is assigned a subscript/index starting with 1.

Cont…

- A PL/SQL VARRAY declaration is done in two steps, like a table declaration:
- Declare a PL/SQL VARRAY type with a TYPE statement.
- The TYPE declaration includes a size to set the upper bound of a Varray. The lower bound is always one.
- Declare an actual Varray based on the type declared in the previous step.

Cont…

The general syntax is

DECLARE

TYPE varraytypename IS VARRAY (size) OF ElementType *[NOT NULL]**; varrayname varraytypename;***

Cont…

For example,

```
DECLARE
TYPE Lname_varray_type IS VARRAY(5) OF employee.LName%TYPE;
Lname_varray Lname_varray_type := Lname_varray_type( );
```

NAMED BLOCKS

- Procedures
- Functions
- Packages
- Triggers

PROCEDURES

- A procedure is a named PL/SQL program block that can perform one or more tasks.
- A procedure is the building block of modular programming.

Cont..

The general syntax of a procedure is

CREATE [OR REPLACE] PROCEDURE procedurename [(parameter1 [, parameter2 . . .])]

IS

[constant/variable declarations]

BEGIN

executable statements [EXCEPTION

exception handling statements] END [procedurename];

Cont..

Calling a Procedure

- A call to the procedure is made through an executable PL/SQL statement.
- The procedure is called by specifying its name along with the list of parameters (if any) in parentheses.

The general syntax is

procedurename [(parameter1, . . .)];

Cont..

For example,

```
monthly_salary(v_salary);
calculate_net(v_monthly_salary, 0.28);
display_messages;
```

Cont..

Procedure Header

- The procedure definition that comes before the reserved word IS is called the procedure header.
- The procedure header contains the name of the procedure and the parameter list with data types (if any).

Cont…

For example

```
CREATE OR REPLACE PROCEDURE
   monthly_salary (v_salary_in IN
   employee.Salary%TYPE)
CREATE OR REPLACE PROCEDURE
   calculate_net (v_monthly_salary_in IN
   employee.Salary%TYPE, v_taxrate_in IN
   NUMBER)
CREATE OR REPLACE PROCEDURE
   display_messages
```

Cont …

Procedure Body

- The procedure body contains declaration, executable, and exception-handling sections.
- The declaration and exception-handling sections are optional.
- The executable section contains action statements, and it must contain at least one.

Cont…

Parameters

- Parameters are used to pass values back and forth from the calling environment to the Oracle server.
- The values passed are processed and/or returned with a procedure execution.
- There are three types of parameters: IN, OUT, and IN OUT

Cont…

Actual and Formal Parameters

- The parameters passed in a call statement are called the **actual parameters**.
- The parameter names in the header of a module are called the **formal parameters**.
- The actual parameters and their matching formal parameters must have the same data types.
- In a procedure call, the parameters are passed without data types.
- The procedure header contains formal parameters with data types, but the size of the data type is not required

Cont…

Matching Actual and Formal Parameters

- There are two different ways in PL/SQL to link formal and actual parameters:
- In ***positional notation***, the formal parameter is linked with an actual parameter implicitly by position . Positional notation is more commonly used for parameter matching.
- In ***named notation***, the formal parameter is linked with an actual parameter explicitly by name.

The general syntax is

formalparametername => argumentvalue

FUNCTIONS

- A function, like a procedure, is a named PL/SQL block.
- Like a procedure, it is also a stored block.
- The main difference between a function and a procedure is that a function always returns a value to the calling block.
- A function is characterized as follows:

A function is characterized as follows:

- A function can be passed zero or more parameters.
- A function must have an explicit RETURN statement in the executable section to return a value.
- The data type of the return value must be declared in the function’s header.
- A function cannot be executed as a stand-alone program

- A function may have parameters of the IN, OUT, and IN OUT types, but the primary use of a function is to return a value with an explicit RETURN statement.
- The use of OUT and IN OUT parameter types in functions is rare—and considered to be a bad practice

Cont…

The general syntax is

```
CREATE [ OR REPLACE ] FUNCTION functionname [ (parameter1 [, parameter2    ]) ]
RETURN DataType
IS BEGIN

   [ constant | variable declarations ] executable statements
RETURN returnvalue

   [ EXCEPTION
exception-handling statements RETURN returnvalue ]
END [ functionname ];
```

Cont..

Function Header

- The function header comes before the reserved word IS.
- The header contains the name of the function, the list of parameters (if any), and the RETURN data type.

Cont…

Function Body

- The body of a function must contain at least one executable statement.
- If there is no declaration, the reserved word BEGIN follows IS.
- If there is no exception handler, you can omit the word EXCEPTION.
- The function name label next to END is optional. There can be more than one return statement, but only one RETURN is executed in a function call.

Cont…

RETURN Data Types

- A function can return a value with a scalar data type, such as VARCHAR2, NUM- BER, BINARY_INTEGER, or BOOLEAN.
- It can also return a composite or complex data type, such as a PL/SQL table, a PL/SQL record, a nested table, VARRAY, or LOB.

Cont…

Calling a Function

- A function call is similar to a procedure call. You call a function by mentioning its name along with its parameters (if any).
- The parameter list is enclosed within parentheses.
- A procedure does not have an explicit RETURN statement, so a procedure call can be an independent statement on a separate line.
- A function does return a value, so the function call is made via an executable statement, such as an assignment, selection, or output statement.

Cont…

Calling a Function from an SQL Statement

- A stored function block can be called from an SQL statement, such as SELECT.

For example,

SELECT get_deptname(10) FROM dual;

PACKAGES

A package is a collection of PL/SQL objects. The objects in a package are grouped within BEGIN and END blocks. A package may contain objects from the following list:

- Cursors.
- Scalar variables.
- Composite variables.
- Constants.
- Exception names.
- TYPE declarations for records and tables.
- Procedures.
- Functions

Cont…

- The objects in a package can be declared as public objects, which can be referenced from outside, or as private objects, which are known only to the package.
- You can restrict access to a package to its specification only and hide the actual programming aspect.
- A package follows some rules of object-oriented programming, and it gives programmers some object-oriented capabilities.
- A package compiles successfully even without a body if the specification compiles.

Cont…

Structure of a Package

- A package provides an extra layer to a module. A module has a header and a body, whereas a package has a specification and a body.
- A module's header specifies the name and the parameters, which tell us how to call that module.
- Similarly, the pack- age specification tells us how to call different modules within a package.

Cont…

Package Specification

- A package specification does not contain any code, but it does contain information about the elements of the package.
- It contains definitions of functions and procedures, declarations of global or public variables, and anything else that can be declared in a PL/SQL block’s declaration section.
- The objects in the specification section of a package are called **public objects**.

Cont…

The general syntax is

CREATE [OR REPLACE] PACKAGE packagename IS [constant, variable and type declarations] [exception declarations]

[cursor specifications]

[function specifications]

[procedure specifications]

END [packagename];

Cont…

Package Body

- A package body contains actual programming code for the modules described in the specification section.
- It also contains code for the modules not described in the specification section.
- The module code in the body without a description in the specification is called a **private module**, or a **hidden module**, and it is not visible outside the body of the package.

Cont…

The general syntax of a package body is

PACKAGE BODY packagename

IS

[variable and type declarations]

[cursor specifications and SELECT queries] [header and body of functions]

[header and body of procedures]

[BEGIN

executable statements]

[EXCEPTION

Exception handlers]

END[packagename];

TRIGGERS

- A **database trigger**, known simply as a **trigger**, is a PL/SQL block.
- It is stored in the database and is called automatically when a triggering event occurs.
- A user cannot call a trigger explicitly.
- The triggering event is based on a Data Manipulation Language (DML) statement, such as INSERT, UPDATE, or DELETE.
- A trigger can be created to fire before or after the triggering event.

Cont…

- The execution of a trigger is also known as **firing the trigger**.

The general syntax is

CREATE [OR REPLACE] TRIGGER triggername

BEFORE | AFTER | INSTEAD OF triggeringevent ON table|view [FOR EACH ROW]

[WHEN condition]

DECLARE

Declaration statements

BEGIN

Executable statements

EXCEPTION

Exception-handling statements

END;

Cont…

```
SQL> /* Anonymous block calls function HAS_PREREQ
DOC> and function FIND_PREREQ in package COURSE_INFO */ SQL> DECLARE
V_FLAG BOOLEAN;
V_COURSEID COURSE.COURSEID%TYPE := ’&P_COURSEID’;
V_TITLE VARCHAR2(30);
BEGIN
V_COURSEID := UPPER(V_COURSEID);
V_FLAG := COURSE_INFO.HAS_PREREQ(V_COURSEID);
IF V_FLAG = TRUE THEN
V_TITLE := COURSE_INFO.FIND_PREREQ(V_COURSEID);
DBMS_OUTPUT.PUT_LINE(’Course: ’ || V_COURSEID);
DBMS_OUTPUT.PUT_LINE(’Pre-Requisite - ’ || V_COURSEID);
END IF;
END; 14 /
Enter value for p_courseid: CIS265 Course: CIS265
Pre-Requisite - CI5253
PL/SQL procedure successfully completed. SQL> /
Enter value for p_courseid: CIS253 No prerequisite
PL/SQL procedure successfully completed.
SQL> /
Enter value for p_courseid: CIS999 Course: CIS999 does not exist
PL/SQL procedure successfully completed.
 SQL>
```

Cont…

BEFORE Triggers

- The BEFORE trigger is fired before execution of a DML statement.
- The BEFORE trigger is useful when you want to plug into some values in a new row, insert a calculated column into a new row, or validate a value in the INSERT query with a lookup in another table

Cont…

```
SQL> CREATE OR REPLACE TRIGGER EMPLOYEE_BI_TRIGGER
BEFORE INSERT ON EMPLOYEE
FOR EACH ROW
DECLARE
V_EMPID EMPLOYEE.EMPLOYEEID%TYPE;
BEGIN
SELECT EMPLOYEE_EMPLOYEEID_SEQ.NEXTVAL
INTO V_EMPID FROM DUAL;
:NEW.EMPLOYEEID := V_EMPID;
:NEW.HIREDATE := SYSDATE;
END;
12 /
Trigger created.
SQL>
```

Cont…

AFTER Triggers

- An AFTER trigger fires after a DML statement is executed.
- It utilizes the built-in Boolean functions INSERTING, UPDATING, and DELETING.
- If the triggering event is one of the three DML statements, the function related to the DML statement returns TRUE and the other two return FALSE.

Cont…

```
SQL> CREATE OR REPLACE TRIGGER EMPLOYEE_ADU_TRIGGER
AFTER DELETE OR UPDATE ON EMPLOYEE
DECLARE
V_TRANSTYPE VARCHAR2(6);
BEGIN
IF DELETING THEN
V_TRANSTYPE := ’DELETE’;
ELSIF UPDATING THEN
V_TRANSTYPE := ’UPDATE’;
END IF;
INSERT INTO TRANSHISTORY
VALUES (’EMPLOYEE’, V_TRANSTYPE, USER, SYSDATE);
END;
14 /
Trigger created.
SQL>
```

DATA DICTIONARY VIEWS

- Oracle maintains a very informative Data Dictionary.
- A few Data Dictionary views are useful for getting information about stored PL/SQL blocks.
- The following are examples of queries to USER_PROCEDURES (for named blocks), USER_TRIGGERS (for triggers only), USER_SOURCE (for all source codes), USER_OBJECTS(for any object), and USER_ERRORS (for current errors) views:

Cont…

```
SELECT Object_Name, Procedure_Name
   FROM USER_PROCEDURES;
SELECT Trigger_Name, Trigger_Type,
   Triggering_Event, Table_Name, Trigger_Body
   FROM USER_TRIGGERS;
SELECT Name, Type, Line, Text FROM
   USER_SOURCE; SELECT Object_Name,
   Object_Type FROM USER_OBJECTS;
SELECT Name, Type, Sequence, Line, Position
   FROM USER_ERRORS;
```

- These views can provide information ranging from the name of an object to the entire source code.
- Use the DESCRIBE command to find out the names of columns in each Data Dictionary view, and issue SELECT queries according to the information desired.