GALGOTIAS UNIVERSITY

School of Computing Science and Engineering

Program: BCA - IOP

Course Code: BCAS3031

Course Name: PL/SQL & Cursors and Triggers

Dr. T. Poongodi

Associate Professor

PL/SQL CASE Statement

- The PL/SQL CASE statement allows to execute a sequence of statements based on a selector.
- A selector can be anything such as variable, function, or expression that the CASE statement evaluates to a Boolean value.
- Use almost any PL/SQL data types as a selector except BLOB, BFILE and composite types.
- Unlike the PL/SQL IF statement, PL/SQL CASE statement uses a selector instead of using a combination of multiple Boolean expressions.

```
[<<label_name>>]
CASE [TRUE | selector]
        WHEN expression1 THEN
                sequence_of_statements1;
        WHEN expression2 THEN
                sequence_of_statements2;
...
        WHEN expressionN THEN
                sequence_of_statementsN;
        [ELSE sequence_of_statementsN+1;]
END CASE [label_name];
```

- Followed by the keyword CASE is a selector. The PL/SQL CASE statement evaluates the selector only once to decide which sequence of statements to execute.
- Followed by the selector is any number of the WHEN clauses. If the selector value is equal to expression in the WHEN clause, the corresponding sequence of statement after the THEN keyword is executed.
- If the selector's value is not one of the choices covered by WHEN clause, the sequence of statements in the ELSE clause will be executed. The ELSE clause is optional so if you omit it. PL/SQL will add the following implicit ELSE clause:

```
ELSE RAISE CASE_NOT_FOUND;
```

If you use an implicit ELSE clause in the PL/SQL CASE statement, an CASE_NOT_FOUND exception is raised and can be handled in the exception handling section of the PL/SQL block as usual.

The END CASE clause is used to terminate the CASE statement.

```
SET SERVEROUTPUT ON;
DECLARE
        n_pct employees.commission_pct%TYPE;
        v_eval varchar2(10);
        n_emp_id employees.employee_id%TYPE := 145;
BEGIN
        -- get commission percentage
        SELECT commission_pct
        INTO n_pct
        FROM employees
        WHERE employee_id = n_emp_id;
        -- evalutate commission percentage
```

```
CASE n_pct
          WHEN 0 THEN
                    v_eval := 'N/A’;
          WHEN 0.1 THEN
                    v_eval := 'Low’;
          WHEN 0.4 THEN
                    v_eval := 'High’;
          ELSE
                    v_eval := 'Fair';
END CASE;
-- print commission evaluation
DBMS_OUTPUT.PUT_LINE('Employee ' || n_emp_id ||
                              ' commission ' || TO_CHAR(n_pct) ||
                              ' which is ' || v_eval);
END;
/
```

PL/SQL searched CASE statement

PL/SQL provides a special CASE statement called *searched CASE statement*.
The syntax of the PL/SQL searched CASE statement is as follows:

```
[<<label_name>>]
CASE
      WHEN search_condition_1 THEN
            sequence_of_statements_1;
      WHEN search_condition_2 THEN
            sequence_of_statements_2;
      ...
      WHEN search_condition_N THEN
            sequence_of_statements_N;
      [ELSE sequence_of_statements_N+1;]
END CASE [label_name];
```

- The searched CASE statement has no selector. Each WHEN clause in the searched CASE statement contains a search condition that returns a Boolean value.
- The search condition is evaluated sequentially from top to bottom. If a search condition evaluates to TRUE, the sequence of statements in the corresponding WHEN clause is executed and the control is passed to the next statement, therefore, the subsequent search conditions are ignored.
- If no search condition evaluates to TRUE, the sequence of statements in the ELSE clause will be executed.
- The following is an example of using PL/SQL searched CASE statement:

```
SET SERVEROUTPUT ON;
DECLARE
        n_salary EMPLOYEES.SALARY%TYPE;
        n_emp_id EMPLOYEES.EMPLOYEE_ID%TYPE := 200;
        v_msg VARCHAR(20);
BEGIN
SELECT salary
INTO n_salary
FROM employees
WHERE employee_id = n_emp_id;

CASE
        WHEN n_salary < 2000 THEN
                v_msg := 'Low';
        WHEN n_salary >= 2000 and n_salary <=3000 THEN
                v_msg := 'Fair';
        WHEN n_salary >= 3000 THEN v_msg := 'High';
END CASE;
        DBMS_OUTPUT.PUT_LINE(v_msg);
END;
/
```

CASE	**Similar to IF-THEN-ELSIF statement. A 'SELECTOR' is used to choose the alternatives instead of Boolean expression.**	**Used to select from several alternatives using 'SELECTOR'**
SEARCHED CASE	CASE statement with no actual 'SELECTOR'. Instead, it contains the actual condition (which evaluates to TRUE/FALSE) that will select the alternatives.	Used to choose from more than two alternatives mostly.

Case Statement

- Like the IF statement, the **CASE statement** selects one sequence of statements to execute.
- However, to select the sequence, the CASE statement uses a selector rather than multiple Boolean expressions.
- A selector is an expression whose value is used to select one of several alternatives.

Searched CASE statement

- The searched CASE statement **has no selector**, and it's WHEN clauses contain search conditions that yield Boolean values.

```
DECLARE
        a NUMBER :=55;
        b NUMBER :=5;
        arth_operation VARCHAR2(20) :='MULTIPLY';
BEGIN
        dbms_output.put_line('Program started.' );
        CASE (arth_operation)
                WHEN 'ADD' THEN dbms_output.put_line('Addition of the numbers are: '|| a
                        +b );
                WHEN 'SUBTRACT' THEN dbms_output.put_line('Subtraction of the numbers
                        are: '||a-b );
                WHEN 'MULTIPLY' THEN dbms_output.put_line('Multiplication of the numbe
                        rs are: '|| a*b);
                WHEN 'DIVIDE' THEN dbms_output.put_line('Division of the numbers are:'||
                        a/b);
                ELSE dbms_output.put_line('No operation action defined. Invalid operation';
        END CASE;
                dbms_output.put_line('Program completed.' );
END;
/
```

Output:

```
Program started.
Multiplication of the numbers are: 275
Program completed.
```

```
DECLARE
        a NUMBER :=55;
        b NUMBER :=5;
        arth_operation VARCHAR2(20) :='DIVIDE';
BEGIN
        dbms_output.put_line(‘Program started.' );
        CASE
                WHEN arth_operation = 'ADD’
                        THEN dbms_output.put_line(‘Addition of the numbers are: '||a+b );
                WHEN arth_operation = ‘SUBTRACT’
                        THEN dbms_output.put_line(‘Subtraction of the numbers are: '|| a-b);
                WHEN arth_operation = ‘MULTIPLY’
                        THEN dbms_output.put_line(‘Multiplication of the numbers are: '|| a*b );
                WHEN arth_operation = ’DIVIDE’
                THEN dbms_output.put_line(‘Division of the numbers are: '|| a/b ):
                ELSE dbms_output.put_line(‘No operation action defined. Invalid operation');
        END CASE;
        dbms_output.put_line(‘Program completed.' );
END;
/
```

```
Program started.
Division of the numbers are: 11
Program completed.
```

Thank You