PL/SQL Constructs

- Jayendra Khatod

Objectives

- Construct an IF statement
- Construct and identify different loop statements
- Basic Loop
- For Loop
- While Loop

IF Statements

Syntax

```
IF condition THEN
  statements;
[ELSIF condition THEN
  statements;]
[ELSE
  statements;]
END IF;
```

Simple IF statement:

Set the manager ID to 22 if the employee name is Osborne.

```
IF v_ename = 'OSBORNE' THEN
  v_mgr := 22;
END IF;
```

Simple IF Statements

- **Set the job title to Salesman, the department number to 35, and the commission to 20% of the current salary if the last name is Smith.**
- **Example**

```
. . .
IF v_ename       = ‘Smith' THEN
   v_job        := 'SALESMAN';
   v_deptno     := 35;
   v_new_comm   := sal * 0.20;
END IF;
. . .
```

Compound IF Statements

- **If the last name is Vargas and the salary is more than 6500:**

  Set department number to 60.

```
. . .
IF v_ename = 'Vargas' AND salary > 6500 THEN
v_deptno := 60;
END IF;
. . .
```

IF-THEN-ELSE Statements

- **Set a Boolean flag to TRUE if the hire date is greater than five years; otherwise, set the Boolean flag to FALSE.**

```
DECLARE
     v_hire_date DATE := '12-Dec-1990';
     v_five_years BOOLEAN;
BEGIN
. . .
     IF MONTHS_BETWEEN(SYSDATE,v_hire_date)/12 > 5
     THEN
          v_five_years := TRUE;
     ELSE
          v_five_years := FALSE;
     END IF;
...
```

IF-THEN-ELSIF Statements

- For a given value, calculate a percentage of that value based on a condition.
- Example

```
. . .
IF    v_start > 100 THEN
      v_start := 2 * v_start;
ELSIF v_start >= 50 THEN
      v_start := .5 * v_start;
ELSE
      v_start := .1 * v_start;
END IF;
. . .
```

Case Expressions

```
DECLARE
    v_grade CHAR(1) := 'B' ;
    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;
```

Handling NULLs

When working with nulls, you can avoid some common mistakes by keeping in mind the following rules:

- Simple comparisons involving nulls always yield NULL.
- Applying the logical operator NOT to a null yields NULL.
- In conditional control statements, if the condition yields NULL, its associated sequence of statements is not executed.

Iterative Control: LOOP Statements

- **Loops repeat a statement or sequence of statements multiple times.**
- **There are three loop types:**
  - **Basic loop**
  - **FOR loop**
  - **WHILE loop**

Basic Loop

- **Syntax**

```
LOOP                                   -- delimiter
   statement1;
   . . .                               -- statements
   EXIT [WHEN condition];              -- EXIT statement
END LOOP;                              -- delimiter
```

```
where:  condition          is a Boolean variable or
                           expression (TRUE, FALSE,
                           or NULL);
```

Basic Loop

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

FOR Loop

- **Syntax**

```
FOR counter in [REVERSE]
        lower_bound..upper_bound LOOP
    statement1;
    statement2;
    . . .
END LOOP;
```

  - **Use a FOR loop to shortcut the test for the number of iterations.**
  - **Do not declare the counter; it is declared implicitly.**

FOR Loop

- **Insert the first 10 new line items for order number 601.**
- **Example**

```
BEGIN
  FOR i IN 1..10 LOOP
    dbms_output.put_line(i);
  END LOOP;
END;
```

WHILE Loop

- **Syntax**

```
WHILE condition LOOP
   statement1;
   statement2;
   . . .
END LOOP;
```

Condition is evaluated at the beginning of each iteration.

- **Use the WHILE loop to repeat statements while a condition is TRUE.**

WHILE Loop

- **Example**

```
DECLARE
v_count   NUMBER(2) := 1;
BEGIN
  WHILE v_count <= 10
  LOOP
    dbms_output.put_line (v_count);
    v_count := v_count + 1;
  END LOOP;
  COMMIT;
END;
/
```

Summary

- Change the logical flow of statements by using control structures.
  - Conditional (IF statement)
  - Loops:
    - Basic loop
    - FOR loop
    - WHILE loop
    - EXIT statement

Thank You !