Any substantial program always contains some conditional logic or looping. Oftentimes, both looping and logic are combined to make powerful solutions. The recipes in this chapter will show you some examples using basic conditional logic. Once you've mastered the art of conditional logic, then you will learn how to perform all the loop types that are available in PL/SQL. Lastly, you will see some useful examples that put these concepts into action.

For the purposes of this chapter, it is important to note that a condition is any variable or expression that evaluates to a boolean. Conditions can contain one or more variables or expressions, but they must always evaluate to either TRUE, FALSE, or NULL.

3-1. Choosing When to Execute Code

Problem
Your code contains a condition, and you are interested in executing code to perform specific actions if the condition evaluates to TRUE, FALSE, or NULL.

Solution
Use an IF-THEN statement to evaluate an expression (or condition) and determine which code to execute as a result.

The following example depicts a very simple IF-THEN statement that evaluates one variable to see whether it contains a larger value than another variable. If so, then the statements contained within the IF-THEN statement are executed; otherwise, they are ignored.

```sql
DECLARE
  value_one   NUMBER := &value_one;
  value_two   NUMBER := &value_two;
BEGIN
  IF value_one > value_two THEN
    DBMS_OUTPUT.PUT_LINE('value_one is greater than value_two');
  END IF;
END;
```

As you can see from the example, if `value_one` is greater than `value_two`, a line of output will be printed stating so. Otherwise, the IF statement is bypassed, and processing continues.

How It Works
As shown in the solution, the general format for the IF-THEN statement is as follows:
IF condition THEN
    Statements to be executed

END IF;

The IF-THEN statement is one of the most frequently used conditional statements. If a given condition evaluates to TRUE, then the code contained within the IF-THEN statement is executed. If the condition evaluates to FALSE or NULL, then the statement is exited. However, it is possible to incorporate a different set of statements if the condition is not satisfied. Please see Recipe 3-2 for an example.

Any number of IF-THEN statements can be nested within one another. The statements within the IF-THEN will be executed if the condition that is specified evaluates to TRUE.

3-2. Choosing Between Two Mutually Exclusive Conditions

Problem
You have two conditions that are mutually exclusive. You want to execute one set of statements if the first condition evaluates to TRUE. Otherwise, if the first condition is FALSE or NULL, then execute a different set of statements.

Solution
Use an IF-ELSE statement to evaluate the condition and execute the statements that correspond to it if the condition evaluates to TRUE. In the following example, a given employee ID is used to query the EMPLOYEES table. If that employee exists, then the employee record will be retrieved. If not found, then a message will be displayed stating that no match was found.

DECLARE
    employee                employees%ROWTYPE;
    emp_count               number := 0;
BEGIN
    SELECT count(*)
    INTO emp_count
    FROM employees
    WHERE employee_id = 100;

    IF emp_count > 0  THEN
        SELECT *
        INTO employee
        FROM employees
        WHERE employee_id = 100;

        IF employee.manager_id IS NOT NULL THEN
            DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name ||
                ' has an assigned manager.');
        ELSE
            DBMS_OUTPUT.PUT_LINE(employee.first_name || ' ' || employee.last_name ||
                ' does not have an assigned manager.');
        END IF;
    END IF;