

4

Writing Control Structures

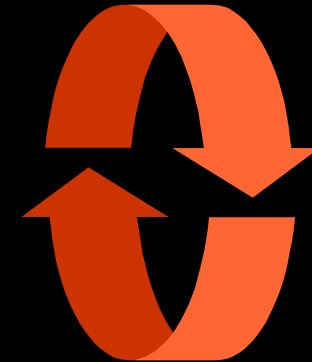
Objectives

After completing this lesson, you should be able to do the following:

- **Identify the uses and types of control structures**
- **Construct an IF statement**
- **Use CASE expressions**
- **Construct and identify different loop statements**
- **Use logic tables**
- **Control block flow using nested loops and labels**

Controlling PL/SQL Flow of Execution

- You can change the logical execution of statements using conditional `IF` statements and loop control structures.
- Conditional `IF` statements:
 - `IF-THEN-END IF`
 - `IF-THEN-ELSE-END IF`
 - `IF-THEN-ELSIF-END IF`



IF Statements

Syntax:

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

If the employee name is Gietz, set the Manager ID to 102.

```
IF UPPER(v_last_name) = 'GIETZ' THEN
    v_mgr := 102;
END IF;
```

Simple IF Statements

If the last name is Vargas:

- Set job ID to SA_REP
- Set department number to 80

```
. . .  
IF v_ename      = 'Vargas' THEN  
    v_job       := 'SA_REP';  
    v_deptno    := 80;  
END IF;  
. . .
```

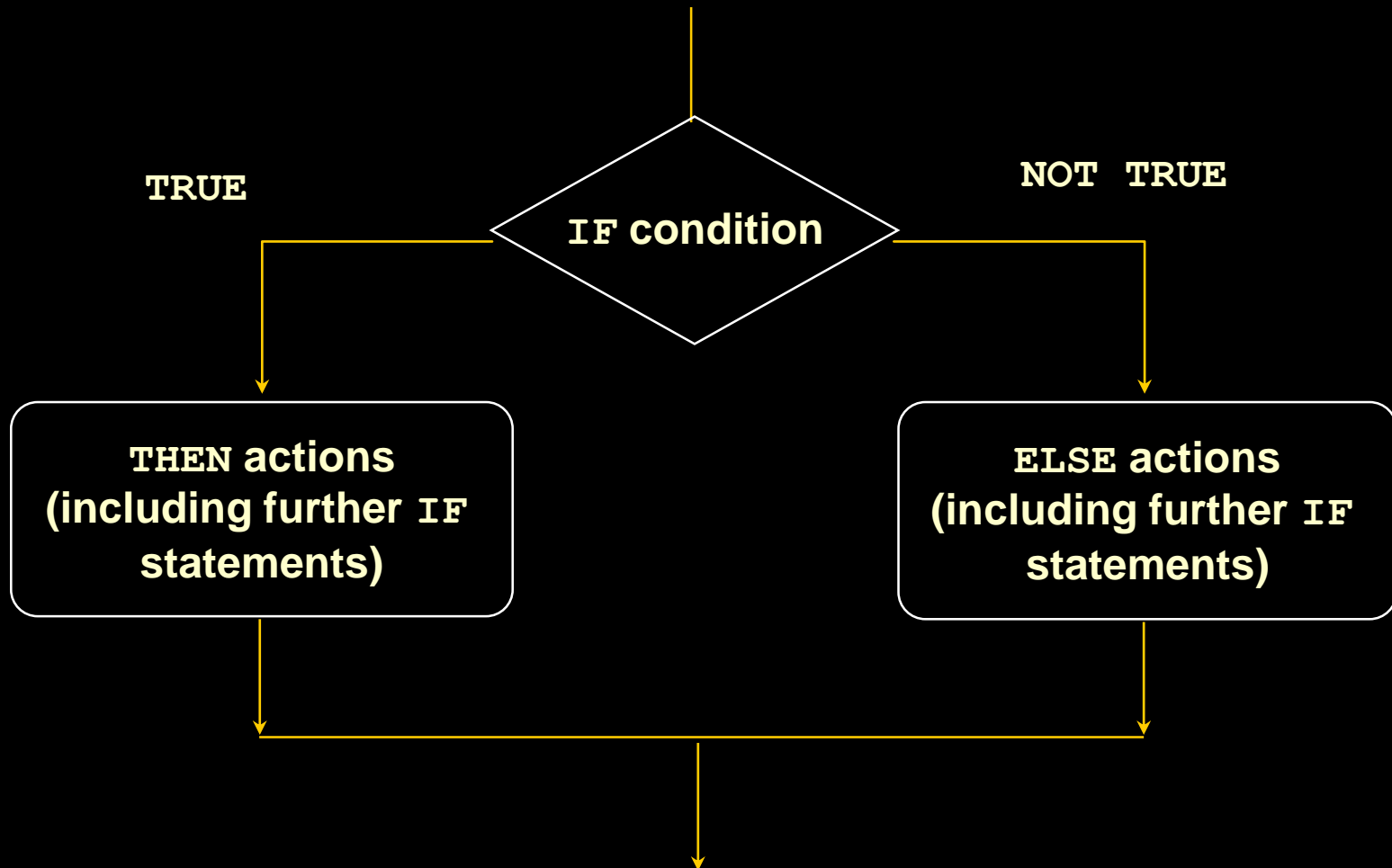
Compound IF Statements

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 Statement Execution Flow

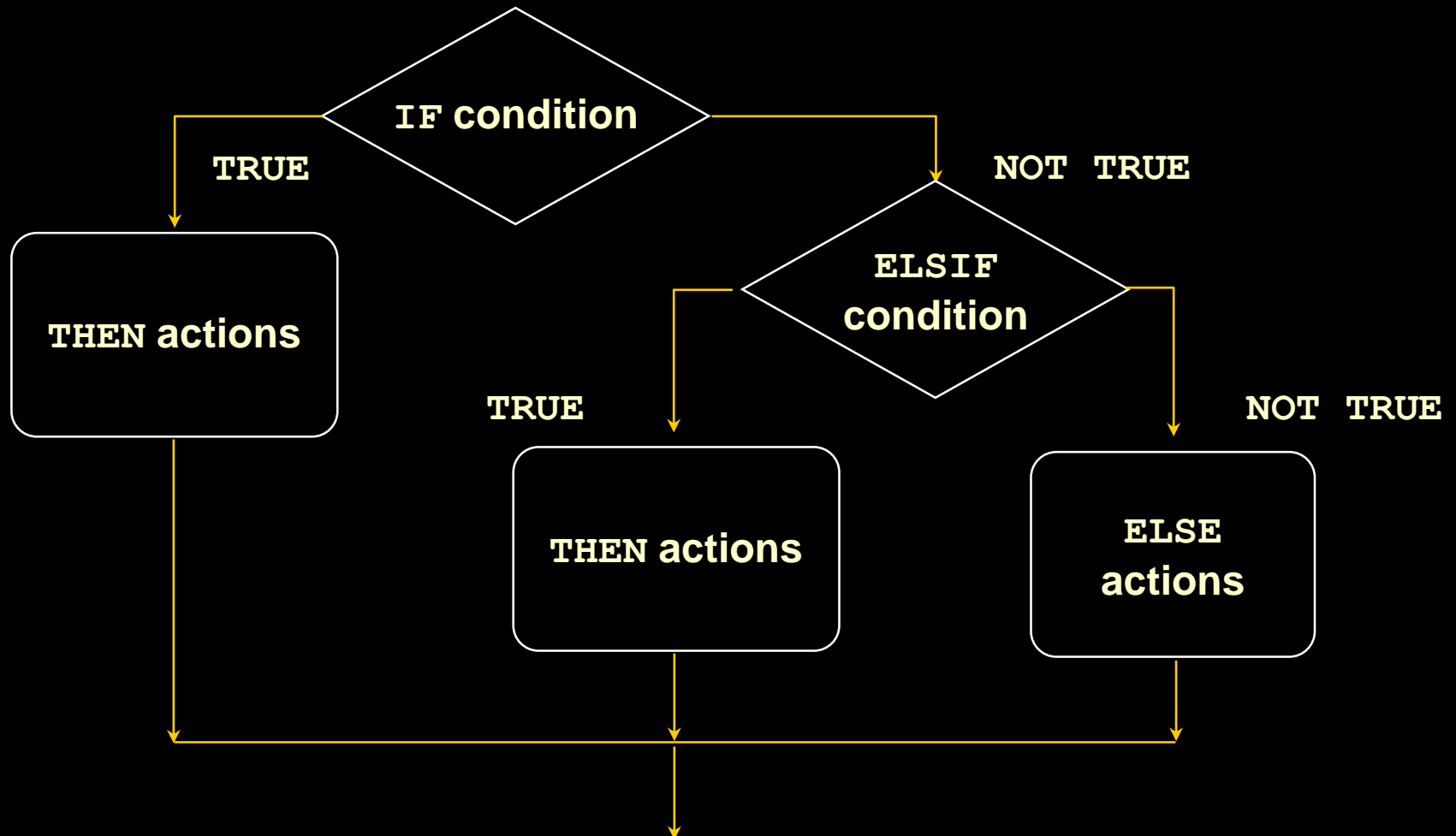


IF-THEN-ELSE Statements

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 Statement Execution Flow



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 := 0.2 * v_start;  
ELSIF v_start >= 50 THEN  
        v_start := 0.5 * v_start;  
ELSE  
        v_start := 0.1 * v_start;  
END IF;  
. . .
```

CASE Expressions

- A CASE expression selects a result and returns it.
- To select the result, the CASE expression uses an expression whose value is used to select one of several alternatives.

```
CASE selector
  WHEN expression1 THEN result1
  WHEN expression2 THEN result2
  ...
  WHEN expressionN THEN resultN
[ELSE resultN+1;]
END;
```

CASE Expressions: Example

```
SET SERVEROUTPUT ON
DECLARE
    v_grade CHAR(1) := UPPER('&p_grade');
    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.**

Logic Tables

Build a simple Boolean condition with a comparison operator.

| AND | <i>TRUE</i> | <i>FALSE</i> | <i>NULL</i> | OR | <i>TRUE</i> | <i>FALSE</i> | <i>NULL</i> | NOT | |
|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------------|--------------|-------|
| <i>TRUE</i> | TRUE | FALSE | NULL | <i>TRUE</i> | TRUE | TRUE | TRUE | <i>TRUE</i> | FALSE |
| <i>FALSE</i> | FALSE | FALSE | FALSE | <i>FALSE</i> | TRUE | FALSE | NULL | <i>FALSE</i> | TRUE |
| <i>NULL</i> | NULL | FALSE | NULL | <i>NULL</i> | TRUE | NULL | NULL | <i>NULL</i> | NULL |

Boolean Conditions

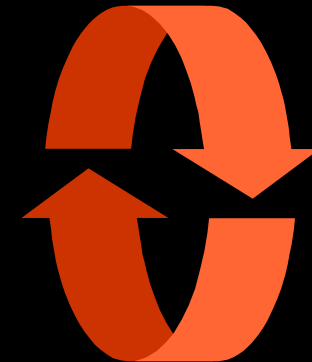
What is the value of **V_FLAG** in each case?

```
v_flag := v_reorder_flag AND v_available_flag;
```

| V_REORDER_FLAG | V_AVAILABLE_FLAG | V_FLAG |
|----------------|------------------|--------|
| TRUE | TRUE | ? |
| TRUE | FALSE | ? |
| NULL | TRUE | ? |
| NULL | FALSE | ? |

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 Loops

Syntax:

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

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

Basic Loops

Example:

```
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;
/
```

WHILE Loops

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 Loops

Example:

```
DECLARE
  v_country_id      locations.country_id%TYPE := 'CA';
  v_location_id     locations.location_id%TYPE;
  v_city            locations.city%TYPE := 'Montreal';
  v_counter          NUMBER := 1;
BEGIN
  SELECT MAX(location_id) INTO v_location_id FROM locations
  WHERE country_id = v_country_id;
  WHILE v_counter <= 3 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;
  END LOOP;
END;
/
```

FOR Loops

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.
- '*lower_bound* .. *upper_bound*' is required syntax.

FOR Loops

Insert three new locations IDs for the country code of CA and the city of Montreal.

```
DECLARE
    v_country_id    locations.country_id%TYPE := 'CA';
    v_location_id    locations.location_id%TYPE;
    v_city           locations.city%TYPE := 'Montreal';
BEGIN
    SELECT MAX(location_id) INTO v_location_id
        FROM locations
        WHERE country_id = v_country_id;
    FOR i IN 1..3 LOOP
        INSERT INTO locations(location_id, city, country_id)
        VALUES((v_location_id + i), v_city, v_country_id );
    END LOOP;
END;
/
```

FOR Loops

Guidelines

- **Reference the counter within the loop only; it is undefined outside the loop.**
- **Do *not* reference the counter as the target of an assignment.**

Guidelines While Using Loops

- Use the basic loop when the statements inside the loop must execute at least once.
- Use the `WHILE` loop if the condition has to be evaluated at the start of each iteration.
- Use a `FOR` loop if the number of iterations is known.

Nested Loops and Labels

- Nest loops to multiple levels.
- Use labels to distinguish between blocks and loops.
- Exit the outer loop with the `EXIT` statement that references the label.

Nested Loops and Labels

```
...  
BEGIN  
  <<Outer_loop>>  
  LOOP  
    v_counter := v_counter+1;  
    EXIT WHEN v_counter>10;  
    <<Inner_loop>>  
    LOOP  
      ...  
      EXIT Outer_loop WHEN total_done = 'YES';  
      -- Leave both loops  
      EXIT WHEN inner_done = 'YES';  
      -- Leave inner loop only  
      ...  
    END LOOP Inner_loop;  
    ...  
  END LOOP Outer_loop;  
END;
```

Summary

In this lesson you should have learned to:

Change the logical flow of statements by using control structures.

- **Conditional (IF statement)**
- **CASE Expressions**
- **Loops:**
 - **Basic loop**
 - **FOR loop**
 - **WHILE loop**
- **EXIT statements**

Practice 4 Overview

This practice covers the following topics:

- **Performing conditional actions using the IF statement**
- **Performing iterative steps using the loop structure**