19

Writing Control Structures

Objectives

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

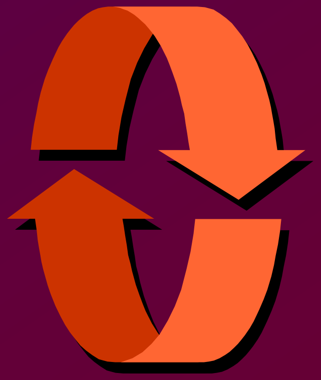
- Identify the uses and types of control structures
- Construct an IF statement
- Construct and identify different loop statements
- Use logic tables
- Control block flow using nested loops and labels

Controlling PL/SQL Flow of Execution

You can change the logical flow 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;
```

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

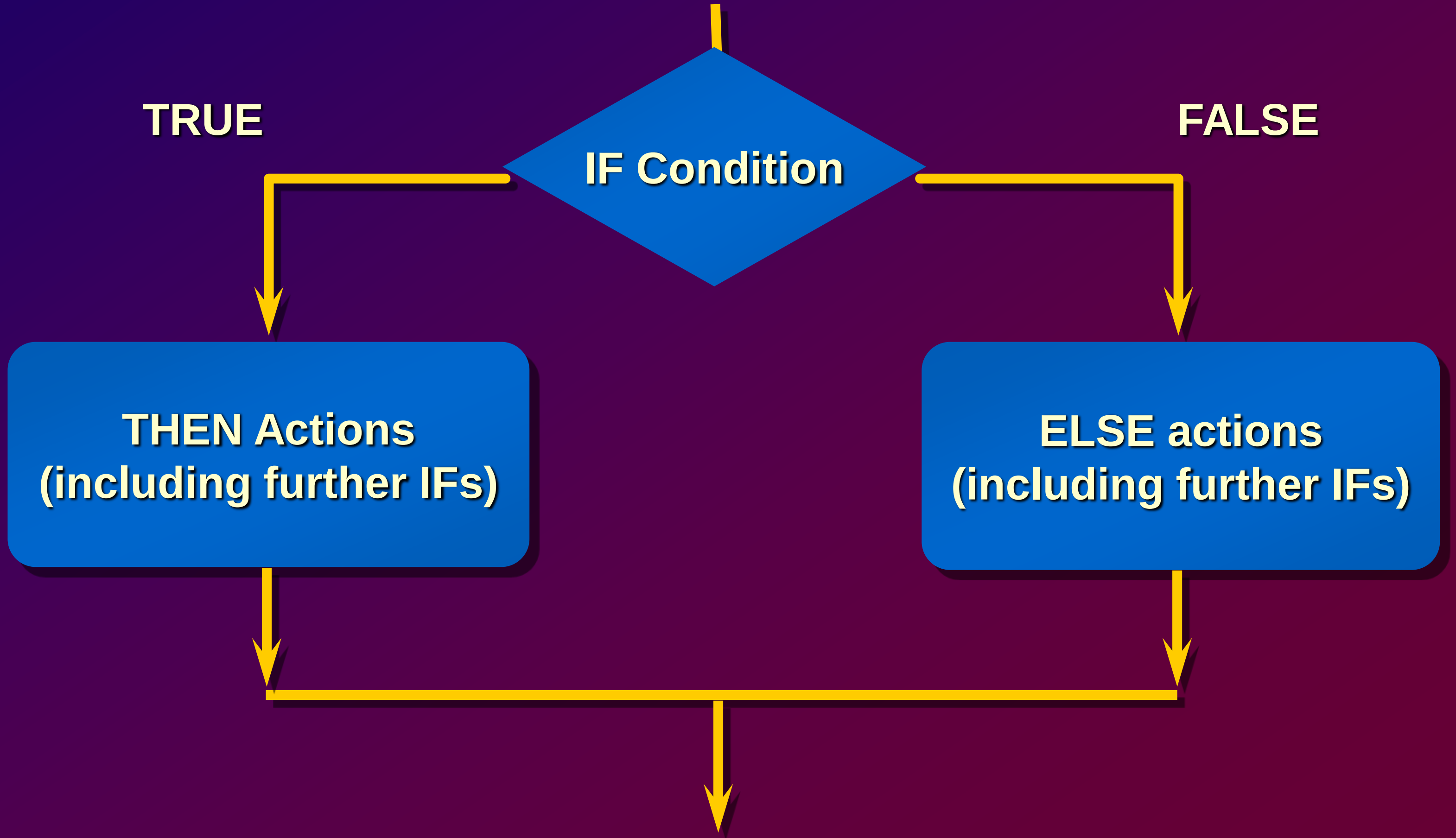
Set the job title to Salesman, the department number to 35, and the commission to 20%of the current salary if the last name is Miller.

Example

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

ORACLE®

IF-THEN-ELSE Statement Execution Flow



TRUE
IF Condition
FALSE
THEN Actions (including further IFs)
ELSE actions (including further IFs)

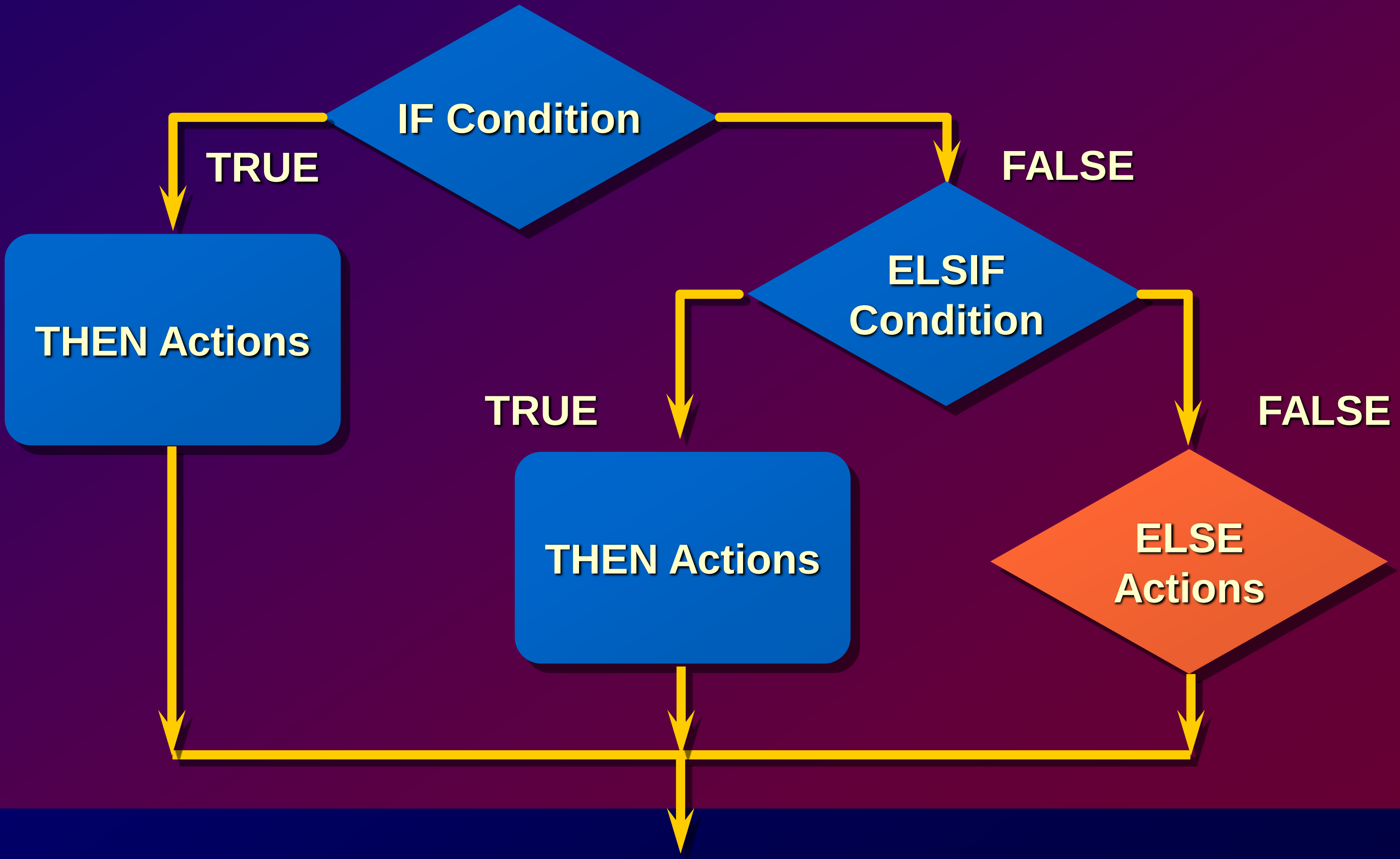
IF-THEN-ELSE Statements

Set a flag for orders where there are fewer than 5 days between order date and ship date.

Example

```
...
IF v_shipdate - v_orderdate < 5 THEN
  v_ship_flag := 'Acceptable';
ELSE
  v_ship_flag := 'Unacceptable';
END IF;
...
```

IF-THEN-ELSIF Statement Execution Flow



IF-THEN-ELSIF Statements

For a given value entered, return a calculated value.

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;
. . .
```

Building Logical Conditions

- You can handle null values with the IS NULL operator.
- Any expression containing a null value evaluates to NULL.
- Concatenated expressions with null values treat null values as an empty string.

Logic Tables

Build a simple Boolean condition with a comparison operator.

AND	*TRUE*	*FALSE*	*NULL*
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NULL	NULL	FALSE	NULL

OR	*TRUE*	*FALSE*	*NULL*
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

NOT	
TRUE	FALSE
FALSE	TRUE
NULL	NULL

Boolean Conditions

What is the value of V_FLAG in each case?

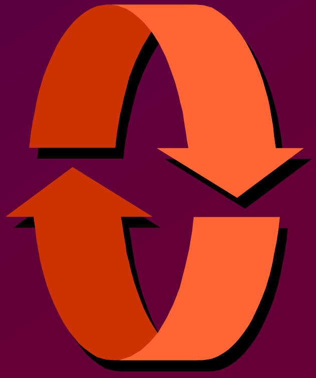
```
v_flag := v_reorder_flag AND v_available_flag;
```

V_REORDER_FLAG	V_AVAILABLE_FLAG	V_FLAG
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
NULL	TRUE	NULL
NULL	FALSE	FALSE

ORACLE®

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);
```

ORACLE®

Basic Loop

Example

```
DECLARE
  v_ordid     item.ordid%TYPE := 101;
  v_counter   NUMBER(2) := 1;
BEGIN
  LOOP
    INSERT INTO item(ordid, itemid)
      VALUES(v_ordid, v_counter);
    v_counter := v_counter + 1;
    EXIT WHEN v_counter > 10;
  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 index; it is declared implicitly.

FOR Loop

Guidelines

- Reference the counter within the loop only; it is undefined outside the loop.
- Use an expression to reference the existing value of a counter.
- Do *not* reference the counter as the target of an assignment.

FOR Loop

Insert the first 10 new line items for order number 101.

Example

```
DECLARE
  v_ordid     item.ordid%TYPE := 101;
BEGIN
  FOR i IN 1..10 LOOP
    INSERT INTO item(ordid, itemid)
      VALUES(v_ordid, i);
  END LOOP;
END;
```

ORACLE®

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

```
ACCEPT p_price PROMPT 'Enter the price of the item: '
ACCEPT p_itemtot PROMPT 'Enter the maximum total for
                         purchase of item: '
DECLARE
...
v_qty                   NUMBER(8) := 1;
v_running_total         NUMBER(7,2) := 0;
BEGIN
  ...
  WHILE v_running_total < &p_itemtot LOOP
    ...
  v_qty := v_qty + 1;
  v_running_total := v_qty * &p_price;
  END LOOP;
...
```

ORACLE®

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 referencing 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

Change the logical flow of statements by using control structures.

- Conditional (IF statement)
- Loops
  - Basic loop
  - FOR loop
  - WHILE loop
  - EXIT statement

Practice Overview

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

ORACLE®