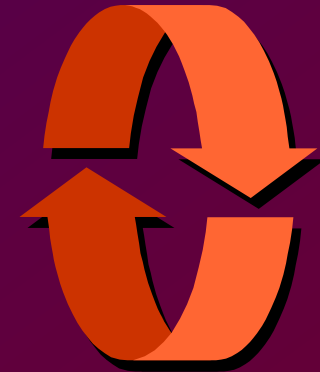


# Controlling PL/SQL Flow of Execution

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;
```

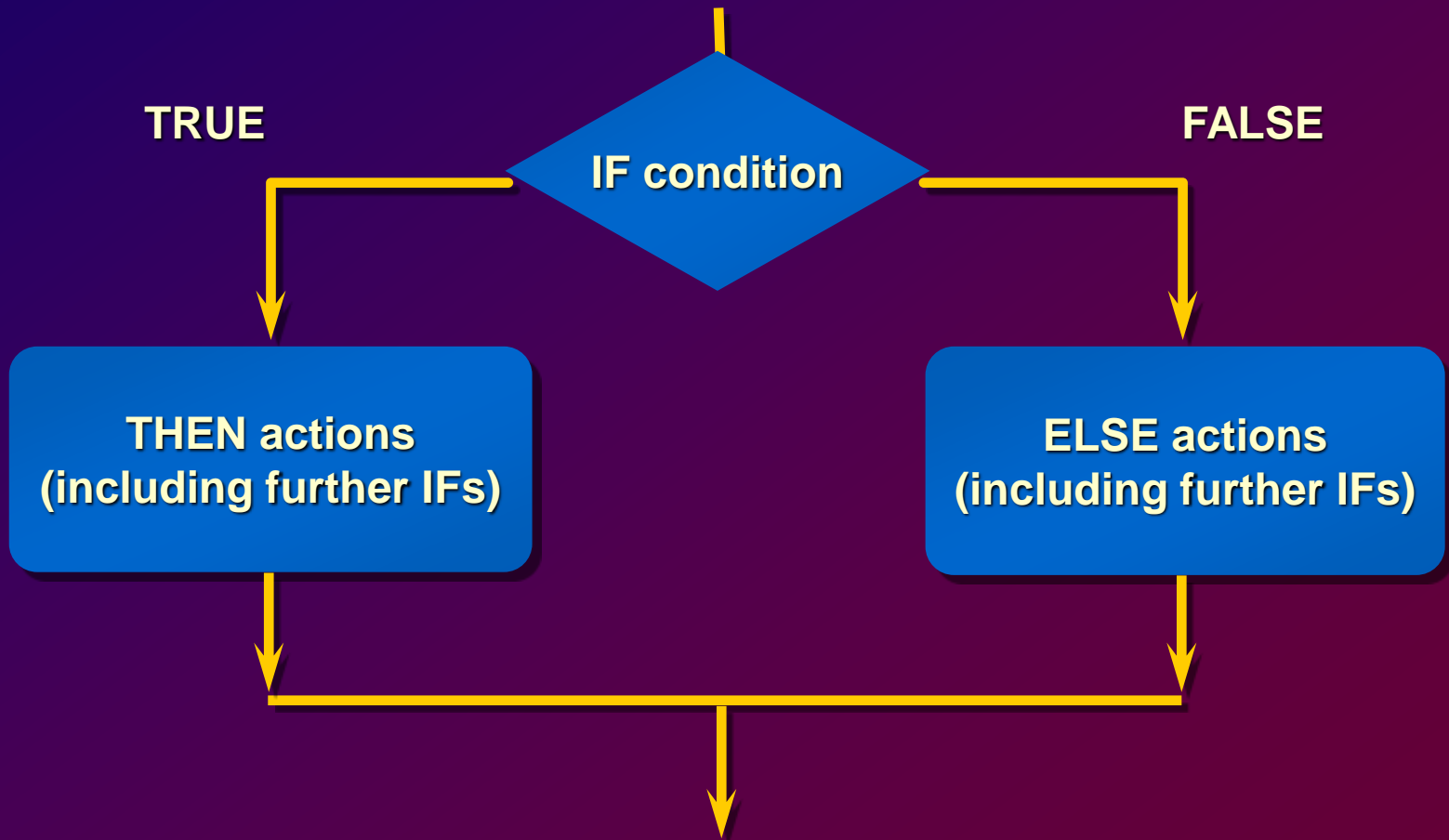
# Simple IF Statements

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;  
. . .
```

# IF-THEN-ELSE Statement Execution Flow



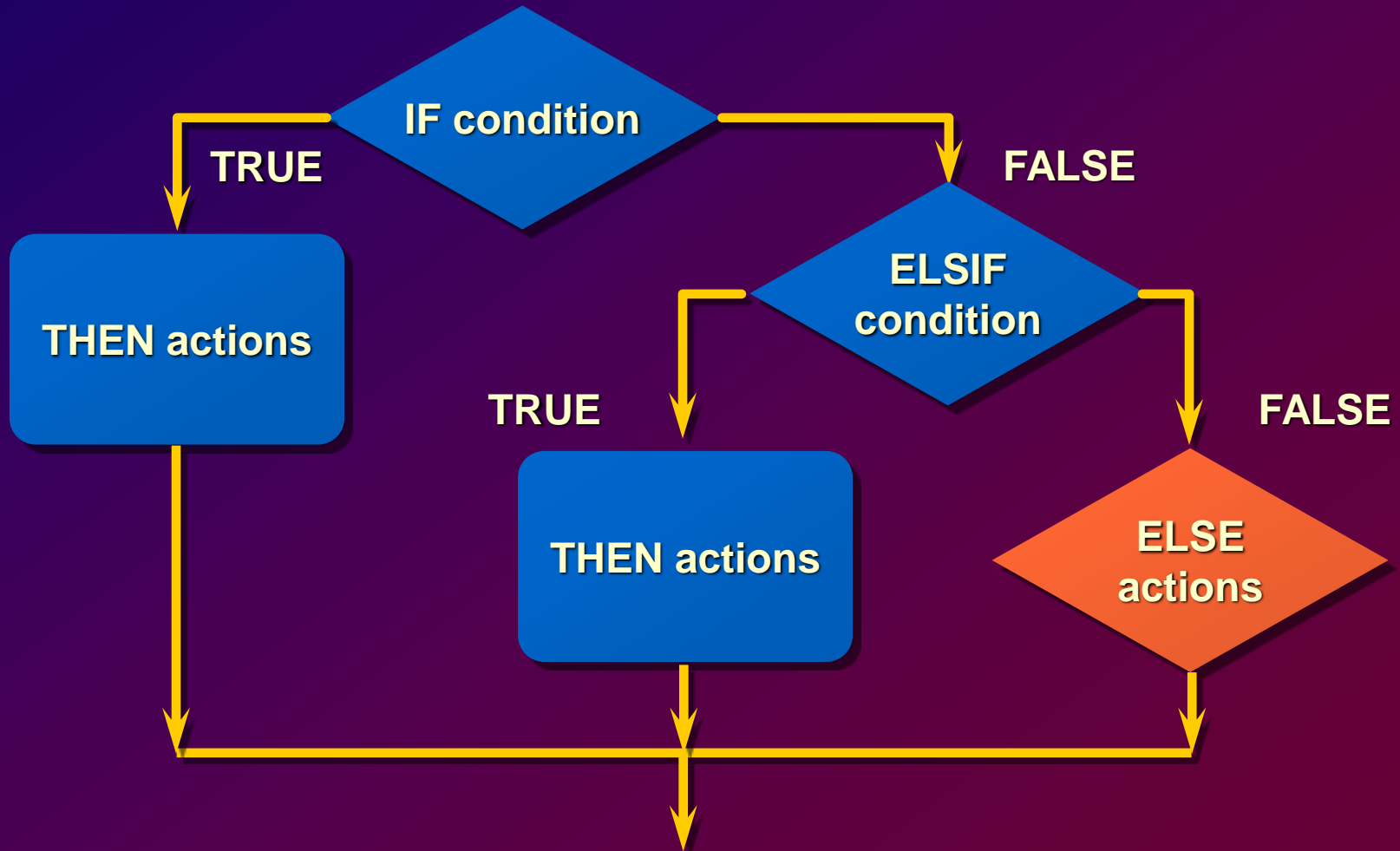
# IF-THEN-ELSE Statements

Set a flag for orders where there are fewer than five 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, 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;  
. . .
```

# Building Logical Conditions

- You can handle null values with the IS NULL operator.
- Any arithmetic 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.**

<b>AND</b>	<i>TRUE</i>	<i>FALSE</i>	<i>NULL</i>	<b>OR</b>	<i>TRUE</i>	<i>FALSE</i>	<i>NULL</i>	<b>NOT</b>	
<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

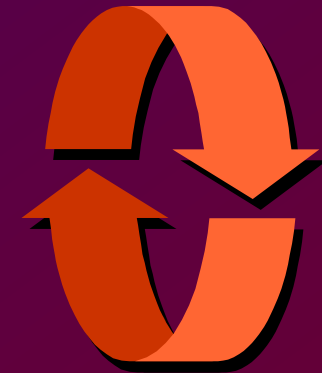
What is the value of V\_FLAG in each case?

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

V_REORDER_FLAG	V_AVAILABLE_FLAG	V_FLAG
TRUE	TRUE	<b>TRUE</b>
TRUE	FALSE	<b>FALSE</b>
NULL	TRUE	<b>NULL</b>
NULL	FALSE	<b>FALSE</b>

# 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

## Example

```
DECLARE
  v_ordid      item.ordid%TYPE := 601;
  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 601.

## Example


```
DECLARE
  v_ordid      item.ordid%TYPE := 601;
BEGIN
  FOR i IN 1..10 LOOP
    INSERT INTO item(ordid, itemid)
      VALUES (v_ordid, 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

```
ACCEPT p_new_order PROMPT 'Enter the order number: '  
ACCEPT p_items -  
    PROMPT 'Enter the number of items in this order: '  
DECLARE  
v_count      NUMBER(2) := 1;  
BEGIN  
    WHILE v_count <= &p_items LOOP  
        INSERT INTO item (ordid, itemid)  
        VALUES (&p_new_order, v_count);  
        v_count := v_count + 1;  
    END LOOP;  
    COMMIT;  
END;  
/
```

# 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;
```