

# 5

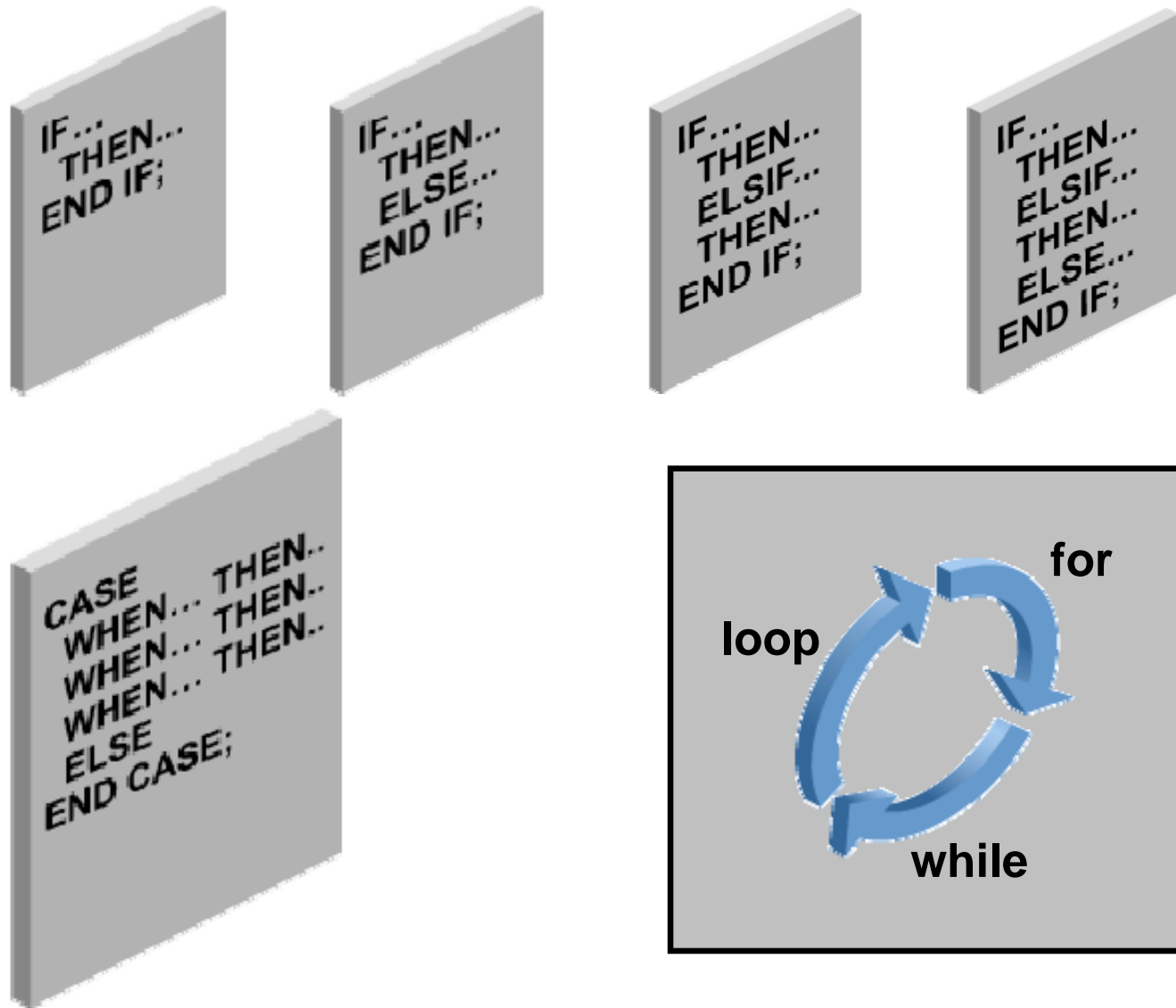
## Writing Control Structures

# 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` statements and `CASE` expressions
- Construct and identify different loop statements
- Use guidelines when using conditional control structures

# Controlling Flow of Execution



# IF Statements

Syntax:

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

# Simple IF Statement

```
DECLARE
  myage number:=31;
BEGIN
  IF myage < 11
  THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
  END IF;
END;
/
```

```
anonymous block completed
```

# IF THEN ELSE Statement

```
SET SERVEROUTPUT ON
DECLARE
myage number:=31;
BEGIN
IF myage < 11
  THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
  ELSE
    DBMS_OUTPUT.PUT_LINE(' I am not a child ');
END IF;
END;
/
```

```
anonymous block completed
I am not a child
```

# IF ELSIF ELSE Clause

```
DECLARE
myage number:=31;
BEGIN
IF myage < 11
THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
ELSIF myage < 20
THEN
    DBMS_OUTPUT.PUT_LINE(' I am young ');
ELSIF myage < 30
THEN
    DBMS_OUTPUT.PUT_LINE(' I am in my twenties');
ELSIF myage < 40
THEN
    DBMS_OUTPUT.PUT_LINE(' I am in my thirties');
ELSE
    DBMS_OUTPUT.PUT_LINE(' I am always young ');
END IF;
END;
/
```

```
anonymous block completed
I am in my thirties
```

# NULL Values in IF Statements

```
DECLARE
myage number;
BEGIN
IF myage < 11
  THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
  ELSE
    DBMS_OUTPUT.PUT_LINE(' I am not a child ');
END IF;
END;
/
```

```
anonymous block completed
I am not a child
```



# CASE Expressions

- A CASE expression selects a result and returns it.
- To select the result, the CASE expression uses expressions. The value returned by these expressions 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
SET VERIFY OFF
DECLARE
    grade CHAR(1) := UPPER('&grade');
    appraisal VARCHAR2(20);
BEGIN
    appraisal :=
        CASE grade
            WHEN 'A' THEN 'Excellent'
            WHEN 'B' THEN 'Very Good'
            WHEN 'C' THEN 'Good'
            ELSE 'No such grade'
        END;
    DBMS_OUTPUT.PUT_LINE ('Grade: ' || grade || '
                          Appraisal ' || appraisal);
END;
/
```

# Searched CASE Expressions

```
DECLARE
    grade CHAR(1) := UPPER('&grade');
    appraisal VARCHAR2(20);
BEGIN
    appraisal :=
        CASE
            WHEN grade = 'A' THEN 'Excellent'
            WHEN grade IN ('B','C') THEN 'Good'
            ELSE 'No such grade'
        END;
    DBMS_OUTPUT.PUT_LINE ('Grade: ' || grade || '
                            Appraisal ' || appraisal);
END;
/
```

# CASE Statement

```
DECLARE
    deptid NUMBER;
    deptname VARCHAR2(20);
    emps NUMBER;
    mngid NUMBER:= 108;
BEGIN
    CASE mngid
        WHEN 108 THEN
            SELECT department_id, department_name
            INTO deptid, deptname FROM departments
            WHERE manager_id=108;
            SELECT count(*) INTO emps FROM employees
            WHERE department_id=deptid;
        WHEN 200 THEN
            ...
        END CASE;
    DBMS_OUTPUT.PUT_LINE ('You are working in the ' || deptname ||
    ' department. There are ' || emps || ' employees in this
    department');
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`.
- If the condition yields `NULL` in conditional control statements, 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 `flag` in each case?

```
flag := reorder_flag AND available_flag;
```

REORDER_FLAG	AVAILABLE_FLAG	FLAG
TRUE	TRUE	? (1)
TRUE	FALSE	? (2)
NULL	TRUE	? (3)
NULL	FALSE	? (4)

# 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
  statement1;
  . . .
  EXIT [WHEN condition];
END LOOP;
```

# Basic Loops

## Example

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

# WHILE Loops

Syntax:

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

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

# WHILE Loops

## Example

```
DECLARE
  countryid    locations.country_id%TYPE := 'CA';
  loc_id       locations.location_id%TYPE;
  new_city     locations.city%TYPE := 'Montreal';
  counter      NUMBER := 1;
BEGIN
  SELECT MAX(location_id) INTO loc_id FROM locations
  WHERE country_id = countryid;
  WHILE counter <= 3 LOOP
    INSERT INTO locations(location_id, city, country_id)
    VALUES((loc_id + counter), new_city, countryid);
    counter := counter + 1;
  END LOOP;
END;
/
```

# FOR Loops

- 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 counter IN [REVERSE]
    lower_bound..upper_bound LOOP
    statement1;
    statement2;
    . . .
END LOOP;
```

# FOR Loops

Example:

```
DECLARE
  countryid    locations.country_id%TYPE := 'CA';
  loc_id       locations.location_id%TYPE;
  new_city     locations.city%TYPE := 'Montreal';
BEGIN
  SELECT MAX(location_id) INTO loc_id
    FROM locations
   WHERE country_id = countryid;
  FOR i IN 1..3 LOOP
    INSERT INTO locations(location_id, city, country_id)
      VALUES((loc_id + i), new_city, countryid );
  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.
- Neither loop bound should be NULL.

# Guidelines for Loops

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



# Nested Loops and Labels

- You can 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  
    counter := counter+1;  
    EXIT WHEN 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 how to change the logical flow of statements by using the following control structures:

- Conditional (IF statement)
- CASE expressions and CASE statements
- Loops:
  - Basic loop
  - FOR loop
  - WHILE loop
- EXIT statements

# Practice 5: Overview

This practice covers the following topics:

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