

FIAP GRADUAÇÃO

# Tecnologia em Análise e Desenvolvimento de Sistemas

Application Development For Databases

Prof. Alan Reis

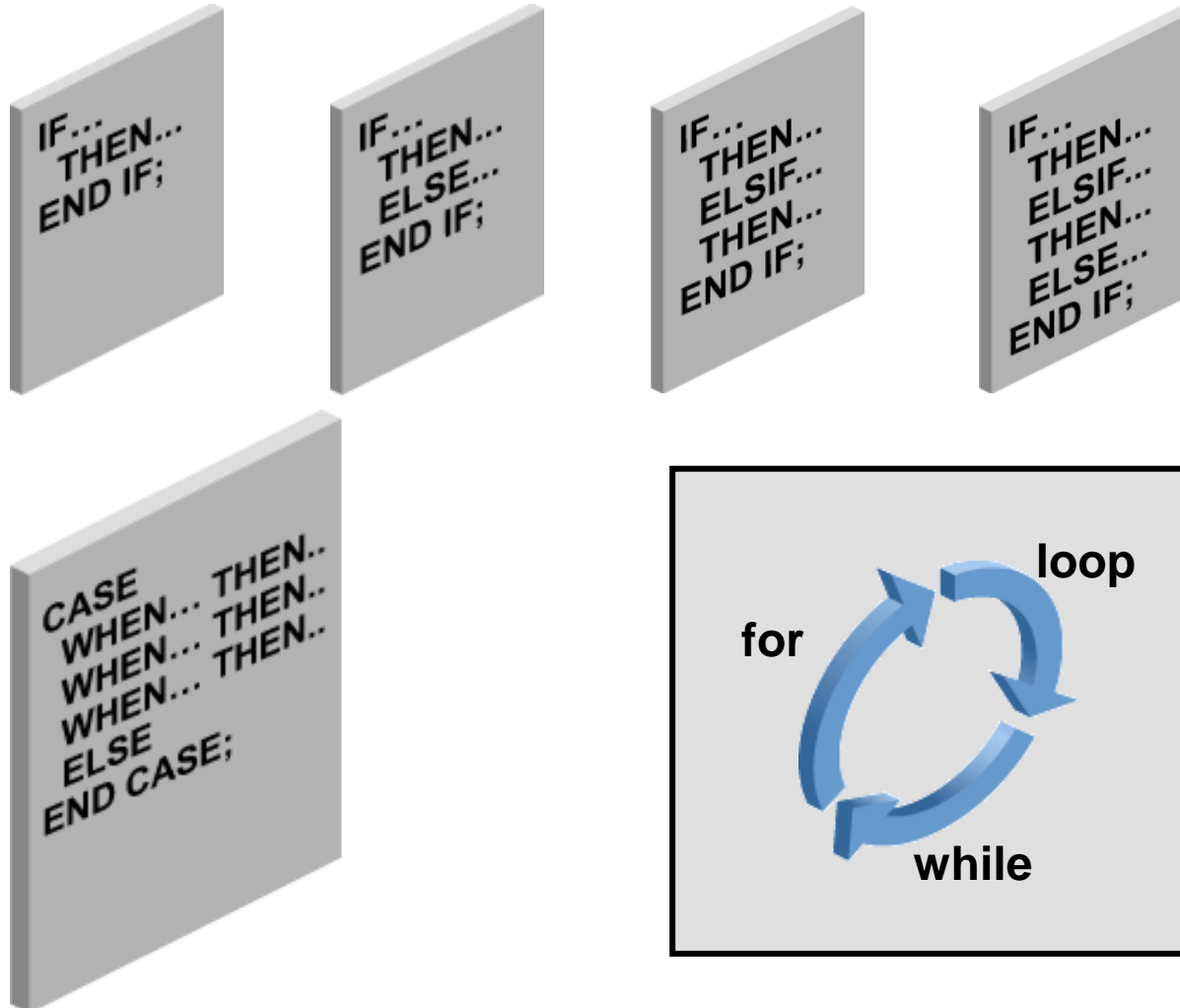
# **Criando Estruturas de Controle**

# Objetivos

Ao concluir esta lição, você será capaz de:

- Identificar os usos e os tipos de estruturas de controle
- Construir uma instrução `IF`
- Usar instruções `CASE` e expressões `CASE`
- Construir e identificar instruções de loop
- Usar diretrizes ao utilizar estruturas de controle condicional

# Controlando o Fluxo de Execução



# Agenda

- Usando instruções `IF`
- Usando instruções `CASE` e expressões `CASE`
- Construindo e identificando instruções de loop

# Instrução IF

Sintaxe:

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

# Instrução IF Simples

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

anonymous block completed



# Instrução IF THEN ELSE

```
DECLARE
  v_myage number:=31;
BEGIN
  IF v_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
```

# Cláusula IF ELIF ELSE

```
DECLARE
  v_myage number:=31;
BEGIN
  IF v_myage < 11 THEN
    DBMS_OUTPUT.PUT_LINE(' I am a child ');
  ELIF v_myage < 20 THEN
    DBMS_OUTPUT.PUT_LINE(' I am young ');
  ELIF v_myage < 30 THEN
    DBMS_OUTPUT.PUT_LINE(' I am in my twenties');
  ELIF v_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
```

# Valor NULL na Instrução IF

```
DECLARE
  v_myage  number;
BEGIN
  IF v_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
```

# Agenda

- Usando instruções `IF`
- Usando instruções `CASE` e expressões `CASE`
- Construindo e identificando instruções de loop

# Expressões CASE

- A expressão CASE seleciona e retorna um resultado.
- Para selecionar o resultado, a expressão CASE usa expressões. O valor retornado por essas expressões é usado para selecionar uma das várias alternativas.

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

# Expressões CASE: Exemplo

```
SET VERIFY OFF
DECLARE
    v_grade CHAR(1) := UPPER('&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;
/
```

# Expressões CASE Pesquisadas

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

# Instrução CASE

```
DECLARE
    v_deptid NUMBER;
    v_deptname VARCHAR2(20);
    v_emps NUMBER;
    v_mngid NUMBER:= 108;
BEGIN
    CASE v_mngid
    WHEN 108 THEN
        SELECT department_id, department_name
        INTO v_deptid, v_deptname FROM departments
        WHERE manager_id=108;
        SELECT count(*) INTO v_emps FROM employees
        WHERE department_id=v_deptid;
    WHEN 200 THEN
        ...
    END CASE;
    DBMS_OUTPUT.PUT_LINE ('You are working in the ' || v_deptname ||
    ' department. There are ' || v_emps || ' employees in this
    department');
END;
/
```



# Tratando Valores Nulos

Ao trabalhar com nulos, é possível evitar alguns erros comuns seguindo estas regras:

- Comparações simples envolvendo nulos sempre retornam `NULL`.
- A aplicação do operador lógico `NOT` a um nulo retorna `NULL`.
- Se a condição retornar `NULL` em instruções de controle condicional, a sequência associada de instruções não será executada.

# Tabelas Lógicas

Construa uma condição booleana simples com um operador de comparação.

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

# Expressões Booleanas ou Expressão Lógica?

Qual é o valor do `flag` em cada caso?

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

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

# Agenda

- Usando instruções `IF`
- Usando instruções `CASE` e expressões `CASE`
- Construindo e identificando instruções de loop

# Controle Iterativo: Instruções LOOP

- Os loops repetem uma instrução (ou uma sequência de instruções) várias vezes.
- Há três tipos de loop:
  - Loop básico
  - Loop `FOR`
  - Loop `WHILE`



# Loops Básicos

Sintaxe:

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

# Loop Básico: Exemplo

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

# Loops WHILE

Sintaxe:

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

Use o loop WHILE para repetir instruções enquanto uma condição for TRUE.



# Loops WHILE: Exemplo

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

# Loops FOR

- Use um loop `FOR` para abreviar o teste diminuindo o número de iterações.
- Não declare o contador; ele é declarado implicitamente.

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

# Loops FOR: Exemplo

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

# Regras do Loop FOR

- Faça referência ao contador apenas dentro do loop; fora do loop ele é indefinido.
- Não faça referência ao contador como o destino de uma designação.
- Nenhum limite de loop pode ser `NULL`.

# Sugestão de Utilização de Loops

- Use o loop básico quando as instruções dentro do loop precisarem ser executadas pelo menos uma vez.
- Use o loop `WHILE` se a condição precisar ser avaliada no início de cada iteração.
- Use um loop `FOR` se o número de iterações for conhecido.

# Loops Aninhados e Labels

- É possível aninhar loops em vários níveis.
- Use labels para diferenciar blocos e loops.
- Saia do loop externo com a instrução `EXIT` que faz referência ao label.

# Loops Aninhados e Labels: Exemplo

```
...  
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;  
/
```

# Instrução CONTINUE do Código PL/SQL FIAP


- Definição
  - Adiciona a funcionalidade para iniciar a próxima iteração de loop
  - Permite que programadores possam transferir o controle para a próxima iteração de um loop
  - Usa estrutura e semântica paralela na instrução EXIT
- Vantagens
  - Facilita o processo de programação
  - Pode oferecer um pequeno aperfeiçoamento do desempenho em relação às soluções de programação anteriores para simular a instrução CONTINUE





# Instrução CONTINUE do Código PL/SQL: Exemplo 1

```
DECLARE
  v_total SIMPLE_INTEGER := 0;
BEGIN
  FOR i IN 1..10 LOOP
    ① v_total := v_total + i;
      dbms_output.put_line
        ('Total is: ' || v_total);
      CONTINUE WHEN i > 5;
    ② v_total := v_total + i;
      dbms_output.put_line
        ('Out of Loop Total is:
          ' || v_total);
    END LOOP;
  END;
/
```

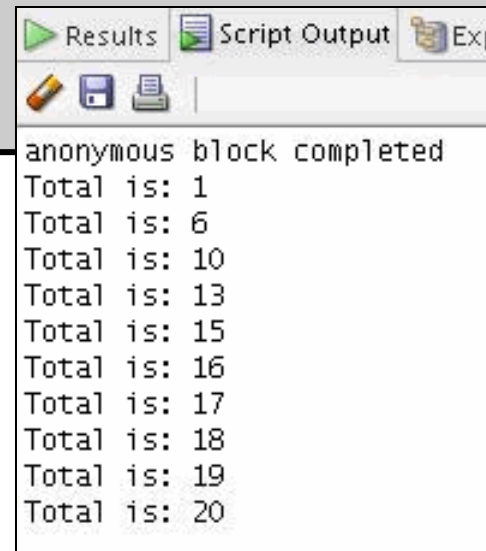


Results Script Output Exp

anonymous block completed  
Total is: 1  
Out of Loop Total is:  
2  
Total is: 4  
Out of Loop Total is:  
6  
Total is: 9  
Out of Loop Total is:  
12  
Total is: 16  
Out of Loop Total is:  
20  
Total is: 25  
Out of Loop Total is:  
30  
Total is: 36  
Total is: 43  
Total is: 51  
Total is: 60  
Total is: 70

# Instrução CONTINUE do Código PL/SQL: Exemplo 2

```
DECLARE
  v_total NUMBER := 0;
BEGIN
  <<BeforeTopLoop>>
  FOR i IN 1..10 LOOP
    v_total := v_total + 1;
    dbms_output.put_line
      ('Total is: ' || v_total);
    FOR j IN 1..10 LOOP
      CONTINUE BeforeTopLoop WHEN i + j > 5;
      v_total := v_total + 1;
    END LOOP;
  END LOOP;
END two_loop;
```



Results Script Output Exp

anonymous block completed

Total is: 1  
Total is: 6  
Total is: 10  
Total is: 13  
Total is: 15  
Total is: 16  
Total is: 17  
Total is: 18  
Total is: 19  
Total is: 20

# Questionário

Existem três tipos de loops: básicos, `FOR` e `WHILE`.

- a. Verdadeiro
- b. Falso

# Sumário

Nesta lição, você aprendeu a alterar o fluxo lógico das instruções usando as seguintes estruturas de controle:

- Condicional (instrução `IF`)
- Expressões `CASE` e instruções `CASE`
- Loops:
  - Loop básico
  - Loop `FOR`
  - Loop `WHILE`
- Instrução `EXIT`
- Instrução `CONTINUE`

## Exercício 5: Visão Geral

- 1)Escreva um bloco PL/SQL que faça a verificação de um numero se é positivo, negativo ou ele é 0.
- 2)Crie um bloco PL/SQL que faça a verificação das notas de 1 a 5 respondendo para (A, B, C, D, E).
- 3)Crie um bloco PL/SQL que imprima os números de 1 a 10 utilizando Loop Simples.
- 4)Faça um bloco PL/SQL que utilize for e imprima um contador até 10 dizendo se é um número par ou impar.
- 5)Faça um bloco PL/SQL que imprima o campo HIRE\_DATE de um select na tabela EMPLOYEES utilizando while.

## Referências:

GONÇALVES, E. PL/SQL. Domine a Linguagem do Banco de Dados Oracle. 1º Edição – Casa do Código, 2015.