
FIAP

Sistemas de Informação

Prof. Me. Alexandre Barcelos
profalexandre.barcelos@fiap.com.br

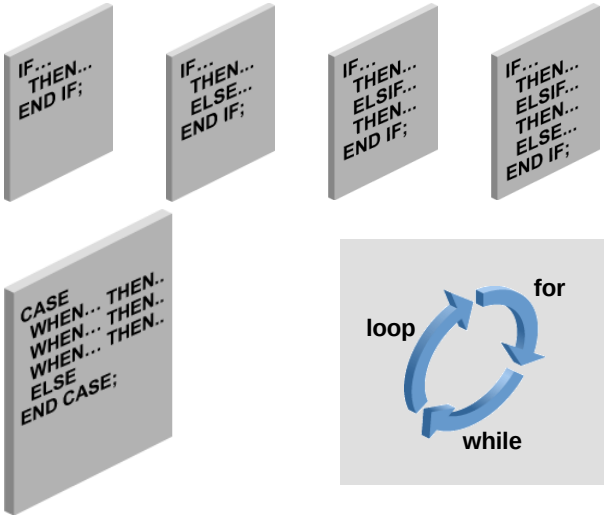
Estruturas de Controle

Objetivos

- **Identificar os usos e tipos de estruturas de controle**
- **Construção da instrução IF**
- **Utilizar as instruções e expressões CASE**
- **Construir e identificar diferentes instruções de loop**

Objetivos:
Controle condicional dentro do bloco PL/SQL usando loops e instruções IF.

Controle do Fluxo de Execução



1-5

Você pode alterar o fluxo lógico de instruções dentro do bloco PL/SQL com diversas *estruturas para controle*. Esta lição aborda os dois tipos de estruturas para controle do PL/SQL: construções condicionais com a instrução IF e estruturas para controle LOOP

Existem três formatos de instruções IF:

- `IF-THEN-END IF`
- `IF-THEN-ELSE-END IF`
- `IF-THEN-ELSIF-END IF`

Declarações IF

Sintaxe:

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

1-6

Instruções IF

A estrutura da instrução IF do PL/SQL é semelhante à estrutura das instruções IF em outras linguagens procedurais. Ela permite que o PL/SQL execute ações de modo seletivo com base em condições.

Na sintaxe:

- *condição* é uma expressão ou variável Booleana (TRUE, FALSE ou NULL) (Ela está associada a uma sequência de instruções, que será executada somente se a expressão produzir TRUE.)
- THEN é uma cláusula que associa a expressão Booleana que a precede com a sequência de instruções posterior
- *instruções* pode ser uma ou mais instruções SQL ou PL/SQL. (Elas podem incluir mais instruções IF contendo diversos IFs, ELSEs e ELSIFs aninhados.)
- ELSIF é uma palavra-chave que introduz uma expressão Booleana. (Se a primeira condição produzir FALSE ou NULL, a palavra-chave ELSIF introduzirá condições adicionais.)
- ELSE é uma palavra-chave que se for atingida pelo controle, executará a sequência de instruções que segue a palavra-chave

Declaração IF Simples

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

PL/SQL procedure successfully completed.

1-7

Instruções IF Simples

No exemplo do slide, o PL/SQL executará essas três ações (definindo as variáveis v_job, v_deptno e v_new_comm) somente se a condição for TRUE. Se a condição for FALSE ou NULL, o PL/SQL as ignorará. Em ambos os casos, o controle será reiniciado na próxima instrução do programa após END IF.

Diretrizes

- Você pode executar ações de maneira seletiva com base nas condições atendidas.
- Ao criar um código, lembre-se da grafia das palavras-chave:
  - ELSIF é uma palavra.
  - END IF são duas palavras.
- Se a condição Booleana para controle for TRUE, a seqüência de instruções associada será executada; se ela for FALSE ou NULL, a seqüência de instruções associadas será ignorada. É permitido qualquer quantidade de cláusulas ELSIF.
- Pode haver no máximo uma cláusula ELSE.
- Endente instruções executadas condicionalmente para clareza.

Declaração IF THEN ELSE

```
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;
/
```

```
I am not a child
PL/SQL procedure successfully completed.
```

1-8

Fluxo de Execução da Instrução IF-THEN-ELSE

Se a condição for FALSE ou NULL, você poderá usar a cláusula ELSE para executar outras ações.

Assim como com a instrução IF simples, o controle reiniciará no programa a partir de END IF. Por exemplo:

```
IF condição1 THEN
        instrução1;
ELSE
        instrução2;
END IF;
```

Instruções IF Aninhadas

Os dois conjuntos de ações do resultado da primeira instrução IF podem incluir mais instruções IF antes que as ações específicas sejam executadas. As cláusulas THEN e ELSE podem incluir instruções IF. Cada instrução IF aninhada deve ser terminada com um END IF correspondente.

```
IF condição1 THEN
        instrução1;
ELSE
        IF condição2 THEN
                instrução2;
        END IF;
END IF;
```

Claúsula IF ELSIF ELSE

```
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;
/
```

```
I am in my thirties
PL/SQL procedure successfully completed.
```

1-9

Instruções IF-THEN-ELSIF

Quando possível, use a cláusula ELSIF em vez de aninhar instruções IF. O código fica mais fácil de ler e entender e a lógica é identificada claramente. Se a ação na cláusula ELSE consistir puramente de outra instrução IF, será mais conveniente usar a cláusula ELSIF. Isso torna o código mais claro pois elimina a necessidade de END IFs aninhadas ao final de cada conjunto de condições e ações futuras.

Exemplo

```
IF condição1 THEN
        instrução1;
ELSIF condição2 THEN
        instrução2;
ELSIF condição3 THEN
        instrução3;
END IF;
```

A instrução IF-THEN-ELSIF de exemplo acima é definida ainda mais da seguinte forma:

Para um determinado valor, calcule um percentual do valor original. Se o valor for superior a 100, o valor calculado será o dobro do valor inicial. Se o valor estiver entre 50 e 100, o valor calculado será 50% do valor inicial. Se o valor informado for menor que 50, o valor calculado será 10% do valor inicial.

Observação: Qualquer expressão aritmética contendo valores nulos será avaliada como nula.

Valores NULL em Declarações IF

```
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;
/
```

```
I am not a child
PL/SQL procedure successfully completed.
```

1-10

Desenvolvendo Condições Lógicas

Você pode desenvolver uma condição Booleana simples combinando expressões de data, números ou caracteres com um operador de comparação. Normalmente, manipule valores nulos com o operador IS NULL.

Null em Expressões e Comparações

• A condição IS NULL será avaliada como TRUE somente se a variável que ela estiver verificando for NULL.

• Qualquer expressão contendo um valor nulo é avaliada como NULL, com exceção de uma expressão concatenada, que trata os valores nulos como uma string vazia.

Exemplos

Essas duas expressões serão avaliadas como NULL se v_sal for NULL.

v_sal > 1000

v_sal * 1.1

No exemplo a seguir, a string não será avaliada como NULL se v_string for NULL.

'PL'||v_string||'SQL'

Expressões CASE

- **Uma expressão CASE seleciona um resultado e o retorna.**
- **Para selecionar o resultado, a expressão CASE utiliza expressões. O valor retornado por essas expressões é utilizado para selecionar uma de várias alternativas.**

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

1-11

Expressões CASE

Uma expressão CASE retorna um resultado com base em uma ou mais alternativas. Para retornar o resultado, a expressão CASE usa um seletor, que é uma expressão cujo valor é usado para retornar uma das várias alternativas.

O seletor é seguido por uma ou mais cláusulas WHEN que são verificadas sequencialmente.

O valor do seletor determina qual resultado é retornado. Se o valor do seletor for igual ao valor de uma expressão da cláusula WHEN, essa cláusula WHEN será executada e esse resultado será retornado.

O PL / SQL também fornece uma expressão CASE pesquisada, que tem o formato:

```
CASE
  WHEN search_condition1 THEN result1
  WHEN search_condition2 THEN result2
  ...
  WHEN search_conditionN THEN resultN
 [ELSE resultN+1]
END;
```

Uma expressão CASE pesquisada não possui seletor. Além disso, suas cláusulas WHEN contêm condições de pesquisa que geram um valor booleano, em vez de expressões que podem gerar um valor de qualquer tipo.

Expressões CASE : Exemplo

```
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;
/
```

1-12

Expressões CASE: Exemplo

No exemplo do slide, a expressão CASE usa o valor na variável GRADE como expressão. Este valor é aceito pelo usuário usando uma variável de substituição. Com base no valor inserido pelo usuário, a expressão CASE retorna o valor da variável GRADE com base no valor do valor da nota. A saída do exemplo é a seguinte quando você digita a ou A para a nota:

```
Grade: A Appraisal Excellent
PL/SQL procedure successfully completed.
```

Pesquisas com Expressões CASE

```
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;
/
```


Expressões CASE pesquisadas

No exemplo anterior, você viu uma única expressão de teste que era a variável GRADE. A cláusula WHEN comparou um valor com essa expressão de teste. Nas instruções CASE pesquisadas, você não tem uma expressão de teste. Em vez disso, a cláusula WHEN contém uma expressão que resulta em um valor booleano.

O mesmo exemplo é reescrito neste slide para mostrar as instruções CASE pesquisadas.

Instrução CASE

```
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;
/
```

1-14

Instrução CASE

Lembre-se do uso da instrução IF. Você pode incluir n número de instruções PL / SQL na cláusula THEN e também na cláusula ELSE. Da mesma forma, você pode incluir instruções na instrução CASE. A instrução CASE é mais legível em comparação com várias instruções IF e ELSIF.

Qual é a diferença entre uma instrução CASE e uma expressão CASE?

Uma expressão CASE avalia a condição e retorna um valor.

Por outro lado, uma instrução CASE avalia a condição e executa uma ação.

Uma instrução CASE pode ser um bloco PL / SQL completo.

As instruções CASE terminam com END CASE; mas as expressões CASE terminam com END;

Controle iterativo: instruções LOOP

- **Loops repetem uma instrução ou seqüência de instruções várias vezes.**
- **Existem três tipos de loop:**
  - **Basic loop**
  - **`FOR` loop**
  - **`WHILE` loop**

1-15

Controle Iterativo: Instruções LOOP

O PL/SQL oferece diversos recursos para estruturar loops para repetirem uma instrução ou sequência de instruções várias vezes.

As construções em loop são o segundo tipo de estrutura para controle:

- Loop básico para fornecer ações repetitivas sem condições gerais
- Loops FOR para fornecer controle iterativo para ações com base em uma contagem
- Loops WHILE para fornecer controle iterativo para ações com base em uma condição
- Instrução EXIT para terminar loops

Observação: Outro tipo de loop FOR, loop FOR de cursor, será discutido em uma lição subsequente.

Loops Simples

Sintaxe:

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

1-16

Loop Basico

O formato mais simples da instrução LOOP é o loop básico (ou infinito), que delimita uma sequência de instruções entre as palavras-chave LOOP e END LOOP. Sempre que o fluxo de execução atinge a instrução END LOOP, o controle retorna à instrução LOOP correspondente acima. Um loop básico permite a execução de sua instrução pelo menos uma vez, mesmo que a condição já esteja atendida no momento em que o loop foi informado. Sem a instrução EXIT, o loop seria infinito.

A instrução EXIT

Você pode terminar um loop usando a instrução EXIT. O controle passa para a próxima instrução após a instrução END LOOP. Pode-se emitir EXIT como uma ação dentro de uma instrução IF ou como uma instrução independente dentro do loop. A instrução EXIT deve ser colocada dentro de um loop. No segundo caso, você pode anexar uma cláusula WHEN para permitir a terminação condicional do loop. Quando a instrução EXIT é encontrada, a condição na cláusula WHEN é avaliada. Se a condição produzir TRUE, o loop finalizará e o controle passará para a próxima instrução após o loop. Um loop básico pode conter várias instruções EXIT..

Loops Simples

Exemplo:

```
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;
/
```

1-17

Loop Básico (continuação)

O loop básico de exemplo mostrado no slide está definido como se segue: Inserir os 10 primeiros novos itens de linha para o número do pedido 601.

Observação: O loop básico permite a execução de sua instrução pelo menos uma vez, mesmo que a condição já esteja atendida ao entrar com o loop, contanto que a condição esteja colocada no loop de forma a ser verificada somente após essas instruções. Entretanto, se a condição exit for colocada no início do loop, antes de qualquer outra instrução executável, e ela for true, ocorrerá a saída do loop e as instruções jamais serão executadas.

Loops WHILE

Sintaxe:

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

Utilize o loop WHILE para repetir as declarações enquanto a codinção for TRUE.

1-18

Loop WHILE

Você pode usar o loop WHILE para repetir uma seqüência de instruções até a condição para controle não ser mais TRUE. A condição é avaliada ao início de cada iteração. O loop terminará quando a condição for FALSE. Se a condição for FALSE no início do loop, nenhuma iteração futura será executada.

Na sintaxe:

condição é uma expressão ou variável Booleana (TRUE, FALSE, ou NULL)

instrução pode ser uma ou mais instruções SQL ou PL/SQL

Se as variáveis envolvidas nas condições não se alterarem no curso do corpo do loop, a condição permanecerá TRUE e o loop não terminará.

Observação: Se a condição produzir NULL, o loop será ignorado e o controle passará para a próxima instrução.

Loops WHILE

Exemplo:

```
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;
/
```

1-19

Loops While (continuação)

No exemplo do slide, três novos IDs de locais para o código de país da CA e a cidade de Montreal são adicionados.

A cada iteração através do loop WHILE, um contador (contador) é incrementado. Se o número de iterações for menor ou igual ao número 3, o código no loop será executado e uma linha será inserida na tabela de locais. Depois que o contador exceder o número de novos locais para essa cidade e país, a condição que controla o loop é avaliada como FALSE e o loop termina.

Loops FOR

- **Use um loop FOR para testar 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;
```

1-20

Loop FOR

Os loops FOR têm a mesma estrutura geral do loop básico. Além disso, eles têm uma instrução para controle no início da palavra-chave LOOP para determinar o número de iterações que o PL/SQL executa.

Na sintaxe:

- *contador* é um inteiro declarado implicitamente cujo valor aumenta ou diminui automaticamente (diminuirá se a palavra-chave REVERSE for usada) em 1 cada iteração do loop até o limite superior ou inferior a ser alcançado
- REVERSE faz o contador decrescer a cada iteração a partir do limite superior até o limite inferior. (Note que o limite inferior ainda é referenciado primeiro.)
- *limite_inferior* especifica o limite inferior da faixa de valores do contador
- *limite_superior* especifica o limite superior da faixa de valores do contador

Não declare o contador, ele é declarado implicitamente como um inteiro.

Observação: A sequência de instruções é executada sempre que o contador é incrementado, conforme determinado pelos dois limites. Os limites superior e inferior da faixa do loop podem ser literais, variáveis ou expressões, mas devem ser avaliados para inteiros. Se o limite inferior da faixa do loop for avaliado para um inteiro maior do que o limite superior, a sequência de instruções não será executada.

Loop FOR (continuação)

Por exemplo, a instrução1 é executada somente uma vez:

```
FOR i IN 3..3 LOOP
   instrução1;
END LOOP;
```

Observação: Os limites inferior e superior de uma instrução LOOP não precisam ser literais numéricas. Eles podem ser expressões que convertem para valores numéricos.

Exemplo:

```
DECLARE
     v_lower NUMBER := 1;
     v_upper NUMBER := 100;
BEGIN
     FOR i IN v_lower..v_upper LOOP
     ...
     END LOOP;
END;
```

Loops FOR

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;
/
```

1-22

Loops FOR (continuação)

Você já aprendeu como inserir três novos locais para o código de país da CA e a cidade de Montreal usando o loop básico e o loop WHILE. Este slide mostra como obter o mesmo usando o loop FOR.

Loops FOR

Guidelines

- **Referencia o contador somente dentro do loop; Ele é indefinido fora do loop.**
- **Não faça referência ao contador como o destino de uma atribuição.**

1-23

Loops FOR (continuação)

O slide lista as diretrizes a serem seguidas ao gravar um loop FOR. Nota: Os limites inferior e superior de uma instrução LOOP não precisam ser literais numéricos. Eles podem ser expressões que se convertem em valores numéricos. Exemplo:

```
DECLARE
  lower  NUMBER := 1;
  upper  NUMBER := 100;
BEGIN
  FOR i IN lower..upper LOOP
  ...
  END LOOP;
END;
/
```

Guidelines While Using Loops

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

Diretrizes para Loops

Um loop básico permite a execução de sua instrução pelo menos uma vez, mesmo que a condição já seja atendida ao entrar no loop. Sem a instrução EXIT, o loop seria infinito.

Você pode usar o loop WHILE para repetir uma sequência de instruções até que a condição de controle não seja mais TRUE. A condição é avaliada no início de cada iteração. O loop termina quando a condição é FALSE. Se a condição for FALSE no início do loop, nenhuma iteração adicional será executada.

Os loops FOR têm uma instrução de controle antes da palavra-chave LOOP para determinar o número de iterações que o PL / SQL executa. Use um loop FOR se o número de iterações for predeterminado.

Exercícios

OBRIGADO

profalexandre.barcelos@fiap.com.br

https://www.linkedin.com/in/alexandrebarcelos

FIAP


FIAP