Agenda:

- Apresentar os diferentes tipos de laços em PL/SQL
- Demonstrar como usar os laços para controlar o fluxo de execução de um programa
- Praticar a implementação de laços em PL/SQL para resolver problemas específicos

FIAP GRADUAÇÃO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Definição de laços e sua importância na programação

Laços, ou loops, são estruturas de controle fundamentais em programação PL/SQL, assim como em muitas outras linguagens de programação. Eles permitem que você execute um bloco de código repetidamente enquanto uma condição específica for verdadeira ou por um número fixo de vezes

- Tipos de Laços

FOR: Laço para executar um bloco de código um número específico de vezes.

WHILE: Laço para executar um bloco de código enquanto uma condição for verdadeira

LOOP: Laço para executar um bloco de código indefinidamente.

FIAP GRADUAÇÃO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Definição de laços e sua importância na programação

Eficiência: Com os loops, você pode executar a mesma instrução ou bloco de código várias vezes sem repetir o código, o que torna seu programa mais eficiente e economiza tempo de desenvolvimento.

Manipulação de Conjuntos de Dados: Em PL/SQL, é comum lidar com grandes conjuntos de dados, como tabelas de banco de dados. Os loops permitem percorrer esses conjuntos de dados linha por linha, aplicando operações ou processamentos específicos a cada linha.

Automatização de Tarefas: Os loops são úteis para automatizar tarefas repetitivas, como processamento de dados, cálculos complexos, geração de relatórios, entre outros.

Flexibilidade: Os loops oferecem flexibilidade para lidar com diferentes cenários de programação, como iteração sobre uma lista de valores, processamento de registros em uma tabela ou execução de uma sequência de instruções até que uma condição específica seja atendida.

FIAP GRADUAÇÃO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Definição de laços e sua importância na programação

Eficiência: Com os loops, você pode executar a mesma instrução ou bloco de código várias vezes sem repetir o código, o que torna seu programa mais eficiente e economiza tempo de desenvolvimento.

Manipulação de Conjuntos de Dados: Em PL/SQL, é comum lidar com grandes conjuntos de dados, como tabelas de banco de dados. Os loops permitem percorrer esses conjuntos de dados linha por linha, aplicando operações ou processamentos específicos a cada linha.

Automatização de Tarefas: Os loops são úteis para automatizar tarefas repetitivas, como processamento de dados, cálculos complexos, geração de relatórios, entre outros.

Flexibilidade: Os loops oferecem flexibilidade para lidar com diferentes cenários de programação, como iteração sobre uma lista de valores, processamento de registros em uma tabela ou execução de uma sequência de instruções até que uma condição específica seja atendida.

Estrutura de repetição: loop

```
Loop
        < instrução(ões) >
        Exit when < condição >
End loop;
```

Estrutura de repetição: loop

```
DECLARE
        V_CONTADOR NUMBER(2):= 1;
BEGIN
LOOP
        DBMS_OUTPUT.PUT_LINE(V_CONTADOR);
        V_CONTADOR := V_CONTADOR + 1;
        EXIT WHEN V_CONTADOR > 20;
END LOOP;
END;
```

Estrutura de repetição: while

```
WHILE < condição> LOOP
        < instrução(ões) >;
END LOOP;
```

Estrutura de repetição: while

```
DECLARE
V_CONTADOR NUMBER(2):= 1;
BEGIN
WHILE V_CONTADOR <= 20 LOOP
        DBMS_OUTPUT.PUT_LINE(V_CONTADOR);
        V_CONTADOR := V_CONTADOR + 1;
END LOOP;
END;
```

Estrutura de repetição: for

```
FOR < contador> IN <valor inicial> .. <valor final>
LOOP
        < instrução (ões) >;
END LOOP;
```

Estrutura de repetição: for

```
BEGIN
FOR V_CONTADOR IN 1..20 LOOP
        DBMS_OUTPUT.PUT_LINE(V_CONTADOR);
END LOOP;
END;
```

Estrutura de repetição: for - reverse

```
BEGIN
FOR V_CONTADOR IN REVERSE 1..20 LOOP
        DBMS_OUTPUT.PUT_LINE(V_CONTADOR);
END LOOP;
END;
```

Estrutura de repetição: exercícios

1. Montar um bloco de programação que realize o processamento de uma tabuada qualquer, por exemplo a tabuada do número 150.

Gabarito tabuada

```
DECLARE
   v_tabuada number(2) := &tabuada;
            V_CONTADOR NUMBER(2):= 0;
BEGIN
LOOP
            DBMS_OUTPUT.PUT_LINE(V_CONTADOR||' X '||v_tabuada||' = '||v_contador * v_tabuada);
            V_CONTADOR := V_CONTADOR + 1;
            EXIT WHEN V_CONTADOR > 10;
END LOOP;
END;
```

Gabarito tabuada

```
DECLARE
   v_tabuada number(3) := &tabuada;
   V_CONTADOR NUMBER(2):= 0;
BEGIN
WHILE V_CONTADOR <= 10 LOOP
DBMS_OUTPUT.PUT_LINE(V_CONTADOR||' X '||v_tabuada||' = '||v_contador * v_tabuada);
V_CONTADOR := V_CONTADOR + 1;
END LOOP;
END;
```

Gabarito tabuada

```
DECLARE
   V_TABUADA NUMBER(3) := &tabuada;
BEGIN
   FOR V_CONTADOR IN 1..10 LOOP
      DBMS_OUTPUT.PUT_LINE(V_CONTADOR||' X '||v_tabuada||' = '||v_contador * v_tabuada);
   end loop;
END;
```

Estrutura de repetição: exercícios

2. Em um intervalo numérico inteiro, informar quantos números são pares e quantos são ímpares.

2. GABARITO

```
DECLARE
    v_inicio NUMBER := 1; -- Início do intervalo
    v_fim NUMBER := 85; -- Fim do intervalo
    v_qtd_pares NUMBER := 0; -- Variável para armazenar a quantidade de números pares
    v_qtd_impares NUMBER := 0; -- Variável para armazenar a quantidade de números ímpares
    v_contador NUMBER; -- Variável de contador para percorrer o intervalo
BEGIN
    -- Loop para percorrer o intervalo e contar os números pares e ímpares
    FOR v_contador IN v_inicio..v_fim LOOP
        IF MOD(v_contador, 2) = 0 THEN -- Verifica se o número é par
            v_qtd_pares := v_qtd_pares + 1; -- Incrementa a quantidade de números pares
        ELSE
            v_qtd_impares := v_qtd_impares + 1; -- Incrementa a quantidade de números ímpares
        END IF;
    END LOOP;

    -- Exibição dos resultados
    DBMS_OUTPUT.PUT_LINE('Quantidade de números pares: ' || v_qtd_pares);
    DBMS_OUTPUT.PUT_LINE('Quantidade de números ímpares: ' || v_qtd_impares);
END;
/
```

Estrutura de repetição: exercícios

3. Exibir e média dos valores pares em um intervalo numérico e soma dos ímpares.

```
DECLARE
 v_inicio NUMBER := 1; -- Início do intervalo
    v_fim NUMBER := 10; -- Fim do intervalo
    v_soma_impares NUMBER := 0; -- Variável para armazenar a soma dos números ímpares
    v_qtd_impares NUMBER := 0; -- Variável para contar a quantidade de números ímpares
    v_soma_pares NUMBER := 0; -- Variável para armazenar a soma dos números pares
    v_qtd_pares NUMBER := 0; -- Variável para contar a quantidade de números pares
    v_contador NUMBER; -- Variável de contador para percorrer o intervalo
    v_media_pares NUMBER; -- Variável para armazenar a média dos números pares
BEGIN
    -- Loop para percorrer o intervalo e calcular a soma dos ímpares e média dos pares
    FOR v_contador IN v_inicio..v_fim LOOP
        IF MOD(v_contador, 2) = 0 THEN -- Verifica se o número é par
            v_soma_pares := v_soma_pares + v_contador; -- Soma o valor ao total dos pares
            v_qtd_pares := v_qtd_pares + 1; -- Incrementa a quantidade de números pares
        ELSE
            v_soma_impares := v_soma_impares + v_contador; -- Soma o valor ao total dos ímpares
            v_qtd_impares := v_qtd_impares + 1; -- Incrementa a quantidade de números ímpares
        END IF;
    END LOOP;

    -- Calcular a média dos números pares
    IF v_qtd_pares <> 0 THEN
        v_media_pares := v_soma_pares / v_qtd_pares;
    ELSE
        v_media_pares := 0; -- Evita divisão por zero se não houver números pares
    END IF;

    -- Exibição dos resultados
    DBMS_OUTPUT.PUT_LINE('Soma dos números ímpares: ' || v_soma_impares);
    DBMS_OUTPUT.PUT_LINE('Média dos números pares: ' || v_media_pares);
END;
/
```

FIAP GRADUAÇÃO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Lista de exercícios para subir no portal na entrega de trabalhos até dia 24/03/2024

FIAP GRADUAÇÃO

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Tire suas Dúvidas