Laços

A linguagem PL/pgSQL possui alguns tipos de laços de repetição.

LOOP

```
[<<rótulo>>]
LOOP
    instruções
END LOOP;
```

A instrução **LOOP** define um laço incondicional, repetido indefinidamente até ser terminado por uma instrução **EXIT** ou **RETURN**.

```
EXIT [ rótulo ] [ WHEN expressão ];
```

A instrução **EXIT** encerra a execução do laço. Se especificada a cláusula **WHEN**, o laço será encerrado apenas se a expressão for verdadeira. Nos laços aninhados pode ser utilizado um rótulo opcional na instrução **EXIT** para especificar o nível de aninhamento que deve ser terminado.

```
CONTINUE [ rótulo ] [ WHEN expressão ];
```

A instrução **CONTINUE** pula os comandos restantes do laço e volta ao controle do laço para verificar se deve ser realizada uma nova iteração.

LOOP

Exemplo:

```
CREATE FUNCTION fatorial(n INTEGER) RETURNS INTEGER AS $$
  DECLARE
    ret INTEGER;
    i INTEGER;
  BEGIN
    ret = 1;
    i = 1;
    LOOP
      EXIT WHEN i>n;
      ret = ret*i;
      i = i + 1;
    END LOOP;
    RETURN ret;
  END;
$$ LANGUAGE plpgsql;
```

LOOP

Teste:

```
SELECT fatorial(4);
 fatorial
----------
       24
```

WHILE

```
[<<rótulo>>]
WHILE expressão LOOP
    instruções
END LOOP;
```

A instrução WHILE repete uma sequência de instruções enquanto a expressão de condição for avaliada como verdade. A condição é verificada logo antes de cada entrada no corpo do laço.

WHILE

Exemplo:

```
CREATE FUNCTION fatorial(n INTEGER) RETURNS INTEGER AS $$
  DECLARE
    ret INTEGER;
    i INTEGER;
  BEGIN
    ret = 1;
    i = 1;
    WHILE i<=n LOOP
      ret = ret*i;
      i = i + 1;
    END LOOP;
    RETURN ret;
  END;
$$ LANGUAGE plpgsql;
```

Teste:

```
SELECT fatorial(4);
 fatorial
----------
       24
```

FOR

Existem algumas variações do laço **FOR**. Uma das variações cria um laço que interage num intervalo de valores inteiros.

```
[<<rótulo>>]
FOR nome IN [ REVERSE ] expressão .. expressão [ BY expressão ] LOOP
    instruções
END LOOP;
```

A variável de controle do laço é definida automaticamente como sendo do tipo **INTEGER**, e somente existe dentro do laço. As expressões que fornecem o limite inferior e superior do intervalo e o passo são avaliadas somente uma vez, ao entrar no laço. Se não for especificado um passo com a cláusula **BY**, o passo da interação será 1, mas quando **REVERSE** é especificado se torna -1.

FOR

Exemplo:

```
CREATE FUNCTION fatorial(n INTEGER) RETURNS INTEGER AS $$
  DECLARE
    ret INTEGER;
  BEGIN
    ret = 1;
    FOR i IN 1..n LOOP
      ret = ret*i;
    END LOOP;
    RETURN ret;
  END;
$$ LANGUAGE plpgsql;
```

Teste:

```
SELECT fatorial(4);
 fatorial
----------
       24
```

FOR

Outra variação do **FOR** executa um laço com o resultado da consulta.

```
[<<rótulo>>]
FOR registro_ou_linha IN comando LOOP
    instruções
END LOOP;
```

Cada linha de resultado do comando, que deve ser um comando que retorne linhas, é atribuída, sucessivamente, à variável registro ou linha, e o corpo do laço é executado uma vez para cada linha.

FOR

Exemplo:

```
CREATE FUNCTION revenda_lucromedio( cod revenda.codigo%TYPE
) RETURNS venda.valor%TYPE AS $$
  DECLARE
    lucro       venda.valor%TYPE = 0;
    quantidade  INTEGER = 0;
    valores     RECORD;
    ret         venda.valor%TYPE = 0;
  BEGIN
    FOR valores IN SELECT venda.valor, automovel.preco FROM
venda, automovel WHERE revenda=cod AND
automovel.codigo=venda.automovel LOOP
      quantidade = quantidade+1;
      lucro = lucro + COALESCE(valores.valor-
valores.preco,0);
      RAISE NOTICE '% Total %', quantidade, lucro;
    END LOOP;
    ret = lucro/quantidade;
    RETURN ret;
  END;
$$ LANGUAGE plpgsql;
```

FOR

Teste:

```
SELECT revenda_lucromedio('01');
NOTICE:  1 Total 2500.00
NOTICE:  2 Total 5500.00
NOTICE:  3 Total 7500.00
NOTICE:  4 Total 9000.00
 revenda_lucromedio
--------------------
            2250.00
```

FOREACH

O laço **FOREACH** executa um laço com os elementos de um array.

```
[<<rótulo>>]
FOREACH variável [ SLICE número ] IN ARRAY expressão LOOP
    instruções
END LOOP;
```

Cada elemento do array resultante da expressão é atribuída, sucessivamente, à variável e o corpo do laço é executado uma vez para cada elemento.

Os elementos são visitados em ordem de armazenamento, independente do número de dimensões do array.

FOREACH

Exemplo:

```
CREATE FUNCTION somar_array(a INTEGER[]) RETURNS int AS $$
  DECLARE
    ret INTEGER = 0;
    x INTEGER;
  BEGIN
    FOREACH x IN ARRAY a LOOP
      ret = ret + x;
    END LOOP;
    RETURN ret;
  END;
$$ LANGUAGE plpgsql;
```

Teste:

```
SELECT somar_array(ARRAY[[1,2,3],[4,5,6],[7,8,9]]);
 somar_array
-------------
          45
```

FOREACH

Se for atribuído um valor positivo para SLICE, a variável irá receber um array com o número de dimensões indicadas.

FOREACH

Exemplo:

```
CREATE FUNCTION somar_linha(a INTEGER[]) RETURNS VOID AS $$
  DECLARE
    linha INTEGER[];
    soma INTEGER;
    x INTEGER;
  BEGIN
    FOREACH linha SLICE 1 IN ARRAY a LOOP
      soma = 0;
      FOREACH x IN ARRAY linha LOOP
        soma = soma + x;
      END LOOP;
      RAISE NOTICE 'Soma = %', soma;
    END LOOP;
  END;
$$ LANGUAGE plpgsql;
```

FOREACH

Teste:

```
SELECT somar_linha(ARRAY[[1,2,3],[4,5,6],[7,8,9]]);
NOTICE:  Soma = 6
NOTICE:  Soma = 15
NOTICE:  Soma = 24
 somar_linha
-------------
```

Retornando uma Linha

As funções em PL/pgSQL podem retornar uma linha de dados.

Retornando uma Linha

Exemplo:

```
CREATE FUNCTION revenda_maiorvenda( cod revenda.codigo%TYPE
) RETURNS venda AS $$
  DECLARE
    ret          venda;
    valores      venda;
  BEGIN
    FOR valores IN SELECT * FROM venda WHERE revenda=cod
LOOP
      IF ret.valor IS NULL OR valores.valor>ret.valor THEN
        ret = valores;
      END IF;
    END LOOP;
    RETURN ret;
  END;
$$ LANGUAGE plpgsql;
```

Retornando uma Linha

Teste:

```
SELECT * FROM revenda_maiorvenda( '01' );
 cliente | revenda | automovel |    data    |  valor
---------+---------+-----------+------------+----------
 04      | 01      | 11        | 2010-02-23 | 31000.00
```

CREATE TYPE

O comando CREATE TYPE cria um novo tipo de dado para uso no banco de dados corrente.

```
CREATE TYPE nome AS ( nome_do_atributo tipo_de_dado [, . . . ] )
```

Com a criação de tipos definidos pelo usuário, é possível retornar uma linha com qualquer estrutura.

Retornando uma Linha

Exemplo:

```
CREATE TYPE venda_cliente AS ( nome CHAR(10), modelo CHAR(15),
lucro NUMERIC(7,2) );

CREATE FUNCTION revenda_maiorcliente( cod revenda.codigo%TYPE )
RETURNS venda_cliente AS $$
  DECLARE
    ret          venda_cliente;
    valores      RECORD;
  BEGIN
    FOR valores IN SELECT venda.cliente, venda.data, venda.valor-
automovel.preco AS lucro, automovel.modelo FROM venda, automovel
WHERE revenda=cod AND automovel.codigo=venda.automovel LOOP
      IF ret.lucro IS NULL OR valores.lucro>ret.lucro THEN
        ret.modelo = valores.modelo;
        ret.lucro  = valores.lucro;
        SELECT INTO ret.nome nome FROM cliente WHERE
codigo=valores.cliente;
      END IF;
    END LOOP;
    RETURN ret;
  END;
$$ LANGUAGE plpgsql;
```

Retornando uma Linha

Teste:

```
SELECT * FROM revenda_maiorcliente( '01' );
    nome     |      modelo       |  lucro
-------------+-------------------+---------
 Jose        | Corsa Sedan       | 3000.00
```

Retornando uma Linha

Também podem ser usados parâmetros de saída para retornar um registro genérico.

Retornando uma Linha

Exemplo:

```
CREATE FUNCTION revenda_maiorcliente( cod revenda.codigo
%TYPE, OUT r_nome cliente.nome%TYPE, OUT r_modelo
automovel.modelo%TYPE, OUT r_lucro venda.valor%TYPE ) AS $$
  DECLARE
    valores     RECORD;
  BEGIN
    FOR valores IN SELECT venda.cliente, venda.data,
venda.valor-automovel.preco AS lucro, automovel.modelo FROM
venda, automovel WHERE revenda=cod AND
automovel.codigo=venda.automovel LOOP
      IF r_lucro IS NULL OR valores.lucro>r_lucro THEN
        r_modelo = valores.modelo;
        r_lucro  = valores.lucro;
        SELECT INTO r_nome nome FROM cliente WHERE
codigo=valores.cliente;
      END IF;
    END LOOP;
  END;
$$ LANGUAGE plpgsql;
```

Retornando uma Linha

Teste:

```
SELECT * FROM revenda_maiorcliente( '01' );
   r_nome    |     r_modelo      | r_lucro
-------------+-------------------+---------
 Jose        | Corsa Sedan       | 3000.00
```

Retornando uma Linha

Outra forma de retornar uma linha com qualquer estrutura é definir o retorno da função como um registro genérico (RECORD).

Retornando uma Linha

Exemplo:

```
CREATE FUNCTION revenda_maiorcliente( cod revenda.codigo%TYPE )
RETURNS RECORD AS $$
  DECLARE
    ret          RECORD;
    valores      RECORD;
    primeiro     BOOLEAN := TRUE;
  BEGIN
    FOR valores IN SELECT cliente.nome AS cliente, venda.data,
venda.valor-automovel.preco AS lucro, automovel.modelo FROM venda,
automovel, cliente WHERE revenda=cod AND
automovel.codigo=venda.automovel AND cliente.codigo=venda.cliente
LOOP
      IF primeiro THEN
        ret = valores;
        primeiro = FALSE;
      ELSIF valores.lucro>ret.lucro THEN
        ret = valores;
      END IF;
    END LOOP;
    RETURN ret;
  END;
$$ LANGUAGE plpgsql;
```

Retornando uma Linha

Teste:

```
SELECT revenda_maiorcliente( '01' );
                revenda_maiorcliente
-----------------------------------------------------
 ("Jose      ",2010-01-21,3000.00,"Corsa Sedan    ")

SELECT * FROM revenda_maiorcliente( '03' ) AS ( nome
CHAR(10), data DATE, lucro NUMERIC, modelo CHAR(15) );
    nome     |    data    |  lucro  |     modelo
-------------+------------+---------+----------------
 Jose        | 2010-01-21 | 3000.00 | Corsa Sedan
```

RETURN NEXT

O comando **RETURN NEXT** permite que uma função retorne diversas linhas.

Para que a função possa retornar mais de uma linha, é necessário que seja declarada como retornando um **SETOF** do tipo da linha a retornar.

```
RETURN NEXT expressão;
```

O comando **RETURN NEXT** não encerra a execução da função, apenas inclui a expressão no set de registros que será retornado. É necessário um **RETURN** final sem argumentos para encerrar a execução da função.

RETURN NEXT

Exemplo:

```
CREATE TYPE produtosaldo AS ( data DATE, diario INTEGER,
total INTEGER );

CREATE FUNCTION produto_saldo( cod produto.codigo%TYPE )
RETURNS SETOF produtosaldo AS $$
  DECLARE
    ret produtosaldo;
    valores RECORD;
  BEGIN
    ret.diario=0;
    ret.total=0;
    ret.data=NULL;
    FOR valores IN SELECT data, quantidade FROM entrada
WHERE produto=cod UNION SELECT data, quantidade*-1 FROM
saida WHERE produto=cod ORDER BY data LOOP
      IF ret.data IS NULL THEN
        ret.data = valores.data;
      END IF;
      IF valores.data != ret.data THEN
```

RETURN NEXT

```
        ret.total = ret.total + ret.diario;
        RETURN NEXT ret;
        ret.data = valores.data;
        ret.diario = 0;
      END IF;
      ret.diario = ret.diario + valores.quantidade;
    END LOOP;
    IF ret.data IS NOT NULL THEN
      ret.total = ret.total + ret.diario;
      RETURN NEXT ret;
    END IF;
    RETURN;
  END;
$$ LANGUAGE plpgsql;
```

RETURN NEXT

Teste:

```
SELECT * FROM produto_saldo( '04' );
    data    | diario | total
------------+--------+-------
 2010-03-01 |     20 |    20
 2010-03-02 |     -8 |    12
 2010-03-05 |      9 |    21
 2010-03-25 |     -7 |    14
 2010-03-29 |     -5 |     9
```

Retornando um Conjunto de Linhas

É possível retornar um conjunto de linhas genéricas, usando RECORD.

Retornando um Conjunto de Linhas

Exemplo:

```
CREATE FUNCTION produto_movimentacao( cod produto.codigo
%TYPE ) RETURNS SETOF RECORD AS $$
  DECLARE
    ret RECORD;
  BEGIN
    FOR ret IN SELECT CAST( 'entrada' AS TEXT ), data,
quantidade FROM entrada WHERE produto=cod LOOP
      RETURN NEXT ret;
    END LOOP;
    FOR ret IN SELECT CAST( 'saida' AS TEXT ), data,
quantidade FROM saida WHERE produto=cod LOOP
      RETURN NEXT ret;
    END LOOP;
    RETURN;
  END;
$$ LANGUAGE plpgsql;
```

Retornando um Conjunto de Linhas

Teste:

```
SELECT * FROM produto_movimentacao( '04' ) AS ( tipo TEXT,
data DATE, quantidade INTEGER );
```

```
  tipo   |    data    | quantidade
---------+------------+------------
 entrada | 2010-03-01 |         15
 entrada | 2010-03-01 |          5
 entrada | 2010-03-01 |          5
 entrada | 2010-03-05 |         10
 entrada | 2010-03-25 |          5
 saida   | 2010-03-02 |          8
 saida   | 2010-03-05 |          1
 saida   | 2010-03-25 |          8
 saida   | 2010-03-25 |          4
 saida   | 2010-03-29 |          5
```

Retornando um Conjunto de Linhas

Também é possível retornar um conjunto de linhas usando parâmetros de saída.

Retornando um Conjunto de Linhas

Exemplo:

```
CREATE FUNCTION produto_saldo( cod produto.codigo%TYPE, OUT
r_data DATE, OUT r_diario INTEGER, OUT r_total INTEGER )
RETURNS SETOF RECORD AS $$
  DECLARE
    valores RECORD;
  BEGIN
    r_data=NULL;
    r_diario=0;
    r_total=0;
    FOR valores IN SELECT data, quantidade FROM entrada
WHERE produto=cod UNION SELECT data, quantidade*-1 FROM
saida WHERE produto=cod ORDER BY data LOOP
      IF r_data IS NULL THEN
        r_data = valores.data;
      END IF;
      IF valores.data != r_data THEN
        r_total = r_total + r_diario;
        RETURN NEXT;
        r_data = valores.data;
        r_diario = 0;
```

Retornando um Conjunto de Linhas

```
      END IF;
      r_diario = r_diario + valores.quantidade;
    END LOOP;
    IF r_data IS NOT NULL THEN
      r_total = r_total + r_diario;
      RETURN NEXT;
    END IF;
    RETURN;
  END;
$$ LANGUAGE plpgsql;
```

Retornando um Conjunto de Linhas

Teste:

```
SELECT * FROM produto_saldo( '04' );
   r_data    | r_diario | r_total
-------------+----------+---------
 2010-03-01 |       20 |      20
 2010-03-02 |       -8 |      12
 2010-03-05 |        9 |      21
 2010-03-25 |       -7 |      14
 2010-03-29 |       -5 |       9
```

Retornando um Conjunto de Linhas

Outra forma de retornar um conjunto de linhas é declarando a função como retornando uma tabela.

Retornando um Conjunto de Linhas

Exemplo:

```
CREATE FUNCTION produto_saldo( cod produto.codigo%TYPE )
RETURNS TABLE ( r_data DATE, r_diario INTEGER, r_total
INTEGER ) AS $$
  DECLARE
    valores RECORD;
  BEGIN
    r_data=NULL;
    r_diario=0;
    r_total=0;
    FOR valores IN SELECT data, quantidade FROM entrada
WHERE produto=cod UNION SELECT data, quantidade*-1 FROM
saida WHERE produto=cod ORDER BY data LOOP
      IF r_data IS NULL THEN
        r_data = valores.data;
      END IF;
      IF valores.data != r_data THEN
        r_total = r_total + r_diario;
        RETURN NEXT;
        r_data = valores.data;
        r_diario = 0;
```

Retornando um Conjunto de Linhas

```
      END IF;
      r_diario = r_diario + valores.quantidade;
    END LOOP;
    IF r_data IS NOT NULL THEN
      r_total = r_total + r_diario;
      RETURN NEXT;
    END IF;
    RETURN;
  END;
$$ LANGUAGE plpgsql;
```

Retornando um Conjunto de Linhas

Teste:

```
SELECT * FROM produto_saldo( '04' );
   r_data    | r_diario | r_total
-------------+----------+---------
 2010-03-01 |       20 |      20
 2010-03-02 |       -8 |      12
 2010-03-05 |        9 |      21
 2010-03-25 |       -7 |      14
 2010-03-29 |       -5 |       9
```

RETURN QUERY

O comando RETURN QUERY inclui o resultado da consulta no set de registros que será retornado pela função. Esse comando também não encerra a execução da função.

RETURN QUERY

Exemplo:

```
CREATE TYPE movimento AS ( tipo TEXT, data DATE, quantidade
INTEGER );

CREATE FUNCTION produto_movimentacao( cod produto.codigo
%TYPE ) RETURNS SETOF movimento AS $$
  BEGIN
    RETURN QUERY SELECT CAST( 'entrada' AS TEXT ), data,
quantidade FROM entrada WHERE produto=cod;
    RETURN QUERY SELECT CAST( 'saida' AS TEXT ), data,
quantidade FROM saida WHERE produto=cod;
    RETURN;
  END;
$$ LANGUAGE plpgsql;
```

RETURN QUERY

Teste:

```
SELECT * FROM produto_movimentacao( '04' );
  tipo   |    data    | quantidade
---------+------------+------------
 entrada | 2010-03-01 |         15
 entrada | 2010-03-01 |          5
 entrada | 2010-03-01 |          5
 entrada | 2010-03-05 |         10
 entrada | 2010-03-25 |          5
 saida   | 2010-03-02 |          8
 saida   | 2010-03-05 |          1
 saida   | 2010-03-25 |          8
 saida   | 2010-03-25 |          4
 saida   | 2010-03-29 |          5
```

Exercícios

1) Escreva uma função que receba o código de uma diretoria como parâmetro e retorne o registro do funcionário de maior salário da diretoria.

2) Escreva uma função que receba o código de um fabricante e retorne uma tabela no formato venda_cliente com todos os clientes que compraram automóveis do fabricante.

3) Crie uma função que receba o código da conta e retorne a movimentação da conta com a data, valor, descrição do grupo de lançamento e saldo da conta até o lançamento para todos lançamentos da conta.

4) Crie uma função que receba o código da conta, data inicial e data final e retorne um extrato da conta, entre as datas indicadas, com a data, valor, descrição do grupo do lançamento e o saldo total da conta até o lançamento. A primeira linha ser um registro com a data inicial, a descrição 'Saldo inicial' e o saldo anterior a data inicial e a ultima linha deve ser um registro com a data final, a descrição 'Saldo final' e o saldo final do período.

Exercícios

5) Crie uma função que retorne uma tabela com as datas em que ocorreram vendas de automóveis, o total das vendas na data e o total acumulado de todas as vendas ate a data.

6) Crie uma função que receba como parâmetros o código do produto, mês e ano e retorne uma tabela com o saldo acumulado do produto para todos os dias do mês/ano indicado.