Variáveis Cursor

A linguagem PL/pgSQL permite a manipulação de cursores através de variáveis cursor, que sempre são do tipo de dado especial refcursor.

Em funções na linguagem PL/pgSQL, laços FOR utilizam cursores automaticamente.

Usando Cursores

Cursores podem ser criados na seção de declaração das variáveis da função.

Na linguagem PL/pgSQL, os cursores devem ser abertos com o comando **OPEN** antes que sejam utilizados.

Depois de abertos, podem ser usados os comandos **FETCH** e **MOVE** para recuperar dados do cursor ou mover. Pode ser usada qualquer cláusula de direção permitida pela linguagem SQL que não recupere mais de uma linha por vez. A variável **FOUND** indica se foi recuperada um registro do cursor ou se o cursor foi movido com sucesso.

O cursor pode ser fechado com o comando **CLOSE**. Após fechado um cursor pode ser aberto novamente.

Usando Cursores

Exemplo:

```
CREATE FUNCTION vendas_total() RETURNS NUMERIC(9,2) AS $$
  DECLARE
    ret NUMERIC(9,2) := 0;
    vendas CURSOR FOR SELECT * FROM venda;
    linha RECORD;
  BEGIN
    OPEN vendas;
    LOOP
      FETCH vendas INTO linha;
      IF NOT FOUND THEN
        EXIT;
      END IF;
      ret = ret + linha.valor;
    END LOOP;
    RETURN ret;
  END;
$$ LANGUAGE plpgsql;
```

Usando Cursores

Teste:

```
SELECT vendas_total();
 vendas_total
--------------
    328100.00
```

Cursores Desligados

Cursores que tem a consulta a ser executada determinada na declaração da variável cursor são chamados cursores ligados (bound).

É possível declarar uma variável cursor sem especificar a consulta a ser executada. Esse cursor é chamado cursor desligado (unbound) e podem ser usado com qualquer consulta SQL quando o cursor for aberto.

Cursores Desligados

Exemplo:

```
CREATE FUNCTION vendas_revenda( cod revenda.codigo%TYPE )
RETURNS NUMERIC(9,2) AS $$
  DECLARE
    ret NUMERIC(9,2) := 0;
    vendas refcursor;
    linha RECORD;
  BEGIN
    OPEN vendas FOR SELECT * FROM venda WHERE revenda=cod;
    LOOP
      FETCH vendas INTO linha;
      IF NOT FOUND THEN
        EXIT;
      END IF;
      ret = ret + linha.valor;
    END LOOP;
    RETURN ret;
  END;
$$ LANGUAGE plpgsql;
```

Cursores Desligados

Teste:

```
SELECT vendas_revenda( '01' );
 vendas_revenda
----------------
       81500.00
```

Cursores Desligados

Cursores desligados também podem ser abertos com comandos dinâmicos..

OPEN cursor_desligado [[NO] SCROLL] FOR EXECUTE query_string [USING expression [, ...]];

Cursores Ligados com Parâmetros

Um cursor ligado pode ser criado com uma consulta que receba parâmetros Esse cursor deve ser aberto passando os valores dos parâmetros para a execução da consulta.

OPEN cursor_ligado [([nome :=] valor [, ...])];

Cursores Ligados com Parâmetros

Exemplo:

```
CREATE TYPE revenda_total AS ( nome CHAR(15), total
NUMERIC(9,2) );

CREATE FUNCTION vendas_porrevenda() RETURNS SETOF
revenda_total AS $$
  DECLARE
    total NUMERIC(9,2);
    revendas RECORD;
    linha RECORD;
    ret revenda_total;
    vendas CURSOR ( key revenda.codigo%TYPE ) FOR SELECT *
FROM venda WHERE revenda=key;
  BEGIN
    FOR revendas IN SELECT * FROM revenda LOOP
      total = 0;
      OPEN vendas ( revendas.codigo );
      LOOP
        FETCH vendas INTO linha;
        IF NOT FOUND THEN
```

Cursores Ligados com Parâmetros

```
            EXIT;
          END IF;
          total = total + linha.valor;
        END LOOP;
        ret.nome = revendas.nome;
        ret.total = total;
        RETURN NEXT ret;
        CLOSE vendas;
      END LOOP;
    END;
$$ LANGUAGE plpgsql;
```

Cursores Ligados com Parâmetros

Teste:

```
SELECT * FROM vendas_porrevenda();
      nome        |  total
------------------+----------
 Paraiso          | 81500.00
 Alameda          | 50100.00
 Vale             | 39500.00
 Cabana           | 70000.00
 Portal           | 21500.00
 Santana          | 36000.00
 Triangulo        | 29500.00
```

FOR

Existe uma variação do **FOR** que executa um laço com as linhas de um cursor.

```
[<<rótulo>>]
FOR registro IN cursor_ligado [ ( [ nome := ] valor
[, ...] ) ] LOOP
    instruções
END LOOP;
```

O cursor deve ser um cursor ligado e que não esteja aberto. O laço irá abrir e fechar o cursor automaticamente no início e final da execução.

A variável registro é automaticamente definida com o tipo **RECORD** e só existe dentro do laço. Declarações anteriores da variável serão ignoradas dentro do laço. Cada linha do cursor é atribuída, sucessivamente, à variável registro, e o corpo do laço é executado uma vez para cada linha.

FOR

Exemplo:

```
CREATE FUNCTION vendas_porrevenda() RETURNS SETOF
revenda_total AS $$
  DECLARE
    total NUMERIC(9,2);
    revendas RECORD;
    ret revenda_total;
    vendas CURSOR ( key revenda.codigo%TYPE ) FOR SELECT *
FROM venda WHERE revenda=key;
  BEGIN
    FOR revendas IN SELECT * FROM revenda LOOP
      total = 0;
      FOR linha IN vendas ( key := revendas.codigo ) LOOP
        total = total + linha.valor;
      END LOOP;
      ret.nome = revendas.nome;
      ret.total = total;
      RETURN NEXT ret;
    END LOOP;
  END;
$$ LANGUAGE plpgsql;
```

FOR

Teste:

```
SELECT * FROM vendas_porrevenda();
      nome        |  total
------------------+----------
 Paraiso          | 81500.00
 Alameda          | 50100.00
 Vale             | 39500.00
 Cabana           | 70000.00
 Portal           | 21500.00
 Santana          | 36000.00
 Triangulo        | 29500.00
```

Retornando Cursores

As funções em PL/pgSQL podem retornar cursores.

Internamente, o valor da variável refcursor é uma string que determina o nome do portal através do qual é feito o acesso ao cursor.

O portal é válido apenas dentro da transação onde foi criado.

O nome do portal utilizado para o cursor pode ser especificado pelo programador ou gerado automaticamente. Para especificar o nome do portal deve-se, simplesmente, atribuir uma cadeia de caracteres à variável refcursor antes de abri-la. O valor cadeia de caracteres da variável refcursor será utilizado pelo OPEN como o nome do portal subjacente.

Retornando Cursores

Exemplo:

```
CREATE FUNCTION refaluno(refcursor) RETURNS refcursor AS $$
  BEGIN
    OPEN $1 FOR SELECT * FROM aluno;
    RETURN $1;
  END;
$$ LANGUAGE plpgsql;
```

Retornando Cursores

Teste:

```
BEGIN;
SELECT refaluno( 'alunocursor' );
FETCH ALL IN alunocursor;
COMMIT;
```

matricula	nome	rg	curso	serie	turma
1	Ana Lucia	20143531	0001	1	A
2	Luis Claudio	22336362	0001	1	A
3	Marcelo	25343256	0001	1	A
4	Debora	20356328	0001	1	B
5	Fernanda	26344325	0001	1	B
6	Alvaro	21764527	0001	1	B
7	Claudio	23336368	0002	1	A
8	Andrea	28456474	0002	1	A
9	Carla	23636731	0002	2	A
10	Fernanda	29563735			

Retornando Cursores

Quando a variável refcursor é nula, o OPEN gera automaticamente um nome que não conflita com nenhum portal existente, e atribui este nome à variável refcursor.

Retornando Cursores

Exemplo:

```
CREATE FUNCTION refcliente() RETURNS refcursor AS $$
  DECLARE
    ret refcursor;
  BEGIN
    OPEN ret FOR SELECT * FROM cliente;
    RETURN ret;
  END;
$$ LANGUAGE plpgsql;
```

Teste:

```
BEGIN;
SELECT refcliente();
     refcliente
--------------------
 <unnamed portal 1>
FETCH FORWARD 2 FROM "<unnamed portal 1>";
COMMIT;
 codigo |    nome     | sobrenome
--------+-------------+------------
 01     | Jose        | Santos
 02     | Paulo       | Cunha
```