SQL Procedural

Josino Rodrigues Neto
josinon@gmail.com

SQL Procedural

- Agregada em SQL-92
- As ferramentas têm nomes para suas linguagens SQL procedurais/embutidas
  - Oracle : PL/SQL
  - Postgres PL/Pgsql
  - SQL Server : Transact-SQL

SQL Procedural

- Vantagens
  - Melhor performance
  - Suporte da linguagem SQL

SQL Procedural

- Suporte a módulos de linguagem
- Cursores
- Estrutura de Seleção
- Estrutura de Loop
- Combinação com SQL declarativo
- Combinação com transações
- Tratamento de exceções
- Suporte a escopo de variáveis
- Suporte aos tipos primitivos, complexos e domínios (definidos pelo usuário)

Formato geral cabeçalho

CREATE FUNCTION *name* ([[*argname*] *argtype* [, ...]])

RETURNS *rettype*

Ex.
CREATE FUNCTION atualizaValor (varchar) RETURNS boolean

Formato geral cabeçalho

CREATE [OR REPLACE] FUNCTION

cria e especifica o tipo de objeto (função/procedure)

name ([[*argname*] *argtype* [, ...]])

RETURNS *rettype*

Ex.

CREATE FUNCTION atualizaValor (varchar) RETURNS boolean

Formato geral cabeçalho

CREATE [OR REPLACE] FUNCTION

name ([[*argname*] *argtype* [, ...]])

especifica o nome do objeto função

RETURNS *rettype*

Ex.

CREATE FUNCTION atualizaValor (varchar) RETURNS boolean

Formato geral cabeçalho

CREATE [OR REPLACE] FUNCTION

name ([[*argname*] *argtype* [, ...]])

especifica o(s) parâmetro(s) da função, se houver

RETURNS *rettype*

Ex.

CREATE FUNCTION atualizaValor (varchar) RETURNS boolean

Formato geral cabeçalho

CREATE [OR REPLACE] FUNCTION

name ([[*argname*] *argtype* [, ...]])

RETURNS *rettype*

especifica o tipo de retorno a ser esperado da invocação da função

Onde rettype pode ser :

- Tipo primitivo
  - Estrutura composta
  - Tipo do domínio
  - Coluna de tabela

Formato geral cabeçalho

CREATE [OR REPLACE] FUNCTION

name ([[*argname*] *argtype* [, ...]])

RETURNS *rettype*

especifica o tipo de retorno a ser esperado da invocação da função

Onde *rettype* pode ser :

- Tipo primitivo
- Estrutura composta
- Tipo do domínio
- Coluna de tabela

Aspecto de função

CREATE FUNCTION *name* ([[*argname*] *argtype* [, ...]])
RETURNS *rettype*
AS

‘

… declarações …

’
Language plpgsql ;

- Language pode ser C, sql, plpgsql

Aspecto do corpo da função

```
CREATE FUNCTION name ( [[ argname ] argtype [, ...] ] )
RETURNS rettype AS
    ‘Declare
    variavel tipo ;
Begin
     variavel   :=  20 ;
     return ??
End;
  ’
  Language plpgsql ;
```

Exemplo

- Na tabela Aluno
- Criar função armazenada (“stored procedure”) para atualizar o valor currículo do aluno.
- Parâmetros
  - valor novo para currículo
  - Num_matricula
- Tipo do retorno: boolean

Exemplo

```
CREATE FUNCTION atualizaCurriculo(varchar, varchar)
  RETURNS boolean AS
$$
BEGIN
    UPDATE aluno SET curriculo = $1    WHERE
   num_matricula = $2;

     RETURN FOUND;
END;
$$
  LANGUAGE 'plpgsql' ;
```

- $1, $2 são parâmetros passados pela chamada da função
- FOUND : palavra reservada do sistema; booleano que retorna true sse houve alteração

Binding da SP

- SP criada com sucesso (“compilada”)
- Binding significa invocação da SP !
- Postgres
  - select suaFuncao([parametros]);
- Mysql
  - call suaFuncao([parametros])
- SQL-Server
  - exec suaFuncao([parametros])

Binding da SP/SF

- select atualizaCurriculo('CX','90')

Estudo de caso: plpgsql

- Tipos utilizáveis
- Estrutura de loop
- Estrutura de seleção
- Cursores
- transações

Atributos

- Facilitar manuseio dos objetos de BD
- ROWTYPE
- TYPE

Pgplsql: atributo ROWTYPE

- Estrutura flexível
- Acomoda a estrutura da tabela
- Dinâmico
- Sintaxe

  Variavel nomeTabela%rowtype

Pgplsql: ROWTYPE

```
create function exibeLinhaAluno ( varchar ) returns text as
$$ declare
    linha aluno%rowtype ;
begin
   select * from aluno into linha where num_matricula like
   $1 ;
   return linha.nome || ',' || linha.num_matricula ;
end;
$$ LANGUAGE plpgsql;
```

Pgplsql: RECORD

- Estrutura mais flexível que rowtype
- Acomoda a estrutura durante FOR/LOOP
- Dinâmico
- NÃO é um tipo realmente
- Sintaxe

  Variavel RECORD ;

Plpgsql: estruturas de seleção

- IF ... THEN
- IF ... THEN ... ELSE
- IF ... THEN ... ELSE IF
- IF ... THEN ... ELSIF ... THEN ... ELSE
- IF ... THEN ... ELSEIF ... THEN ... ELSE
  - 5 formas
  - O bloco sempre deve fechar com END IF ;

Plpgsql: estruturas de seleção

IF *condição* THEN
    comandos;
ELSE
    comandos;
END IF;

Plpgsql: estruturas de repetição

- FOR..LOOP
- LOOP
- WHILE ... LOOP

Plpgsql: FOR...LOOP

- Variável para FOR pode ser RECORD ou ROW

```
DECLARE
   reg RECORD;
BEGIN
FOR reg IN consulta LOOP
    comandos
END LOOP;
```

Plpgsql: FOR...LOOP

- FOR i IN 1..10 LOOP
  RAISE NOTICE 'i is %', i;
END LOOP;

- FOR i IN REVERSE 10..1 LOOP
  -- comandos
END LOOP;

FOR..LOOP e RECORD

```
create or replace function exibeLinhaAluno () returns boolean as
  $$
declare
   linha  record ;
begin
  FOR linha IN  select * from aluno  order by num_matricula LOOP
      RAISE NOTICE '%', linha.nome ;
  END LOOP;
  return FOUND;
end; $$ LANGUAGE plpgsql;

Chamada : select exibeLinhaAluno();
```

WHILE LOOP

```
WHILE (condicao_v) LOOP
    declarações
END LOOP;
```

- Teste no início

LOOP

LOOP

declarações

END LOOP;

- Incondicional
- Uso de EXIT ou RETURN para saída

EXIT

EXIT;

EXIT WHEN *condição* ;