



Introdução à PL/pgSQL

TIAGO G MORAES

Roteiro

- Introdução
- Sintaxe básica de uma função
- Sintaxe básica
 - Tipos
 - Parâmetros
 - Retorno
 - Saída de dados
 - Controle de fluxo
 - Estrutura de repetição
 - SQL embutida
- Triggers

Introdução

- *Procedural Language for PostgreSQL*
 - extensão da linguagem SQL para o PostgreSQL
- É uma linguagem de programação estruturada
 - une lógica de programação com comandos SQL
- Uma forma de otimizar processamento no banco de dados e diminuir tráfego Cliente-Servidor
- Produz trechos de códigos que são armazenados nos bancos de dados
 - Procedimentos armazenados (*stored procedures*) → função que retorna void
 - Funções armazenadas (*stored functions*)
 - Gatilhos (*triggers*)

Introdução

- Por que utilizar?
 - O PL/pgSQL estende o poder da linguagem SQL
 - é uma linguagem de programação
- Considere um conjunto de regras (R) aplicadas a determinado banco de dados
 - Exemplo: regras para conceder um seguro de vida
- diversos softwares que utilizem essa base de dados

Introdução

- Por que utilizar?
 - O PL/pgSQL estende o poder da linguagem SQL
 - é uma linguagem de programação
- Considere um conjunto de regras (R) aplicadas a determinado banco de dados
 - Exemplo: regras para conceder um seguro de vida
- diversos softwares que utilizem essa base de dados

Opção 1: Regras R nas aplicações



Introdução

- Por que utilizar?
 - O PL/SQL estende o poder da linguagem SQL
 - é uma linguagem de programação
- Considere um conjunto de regras (R) aplicadas a determinado banco de dados
 - Exemplo: regras para conceder um seguro de vida
- diversos softwares que utilizem essa base de dados

Opção 2: Regras R no SGBD



Introdução



□ Uma função/procedimento pode conter:

- Nome da função
 - Parâmetros
 - retorno
- Declaração das variáveis;
- Código do programa;
- Linguagem utilizada

BANCO DE DADOS

7

Sintaxe básica



□ Sintaxe básica para criação de uma função:

```
CREATE FUNCTION <nome> (lista de parâmetros) RETURNS
<tipo_retorno> AS

$$
DECLARE
<var's>
BEGIN
<algoritmo>
END;
$$ LANGUAGE 'plpgsql'
```

BANCO DE DADOS

8

Sintaxe básica



□ Sintaxe básica para criação de uma função:

```
CREATE FUNCTION <nome> (lista de parâmetros) RETURNS
<tipo_retorno> AS

$$
DECLARE
<var's>
BEGIN
<algoritmo>
END;
$$ LANGUAGE 'plpgsql'
```

→ Cria uma função
CREATE OR REPLACE → salva
por cima se nome já salvo

BANCO DE DADOS

9

Sintaxe básica



□ Sintaxe básica para criação de uma função:

```
CREATE FUNCTION <nome> (lista de parâmetros) RETURNS
<tipo_retorno> AS

$$
DECLARE
<var's>
BEGIN
<algoritmo>
END;
$$ LANGUAGE 'plpgsql'
```

→ Nome da
função

BANCO DE DADOS

10

Sintaxe básica



□ Sintaxe básica para criação de uma função:

```
CREATE FUNCTION <nome> (lista de parâmetros) RETURNS
<tipo_retorno> AS

$$
DECLARE
<var's>
BEGIN
<algoritmo>
END;
$$ LANGUAGE 'plpgsql'
```

→ Parâmetros da função.
Exemplo: idade int

BANCO DE DADOS

11

Sintaxe básica



□ Sintaxe básica para criação de uma função:

```
CREATE FUNCTION <nome> (lista de parâmetros) RETURNS
<tipo_retorno> AS

$$
DECLARE
<var's>
BEGIN
<algoritmo>
END;
$$ LANGUAGE 'plpgsql'
```

→ Tipo de retorno.
Exemplo: int

BANCO DE DADOS

12

Sintaxe básica



□ Sintaxe básica para criação de uma função:

```
CREATE FUNCTION <nome> (lista de parâmetros) RETURNS
<tipo_retorno> AS

$$
DECLARE
<var's>
BEGIN
<algoritmo>
END;
$$ LANGUAGE 'plpgsql'
```

Início da função

BANCO DE DADOS

13

Sintaxe básica



□ Sintaxe básica para criação de uma função:

```
CREATE FUNCTION <nome> (lista de parâmetros) RETURNS
<tipo_retorno> AS

$$
DECLARE
<var's>
BEGIN
<algoritmo>
END;
$$ LANGUAGE 'plpgsql'
```

Declaração de
Variáveis.
Exemplo: aux int;

ATENÇÃO: Os parâmetros
não são redeclarados!!

BANCO DE DADOS

14

Sintaxe básica



□ Sintaxe básica para criação de uma função:

```
CREATE FUNCTION <nome> (lista de parâmetros) RETURNS
<tipo_retorno> AS

$$
DECLARE
<var's>
BEGIN
<algoritmo>
END;
$$ LANGUAGE 'plpgsql'
```

Começo do algoritmo

BANCO DE DADOS

15

Sintaxe básica



□ Sintaxe básica para criação de uma função:

```
CREATE FUNCTION <nome> (lista de parâmetros) RETURNS
<tipo_retorno> AS

$$
DECLARE
<var's>
BEGIN
<algoritmo>
END;
$$ LANGUAGE 'plpgsql'
```

Fim da função

BANCO DE DADOS

16

Sintaxe básica



□ Sintaxe básica para criação de uma função:

```
CREATE FUNCTION <nome> (lista de parâmetros) RETURNS
<tipo_retorno> AS

$$
DECLARE
<var's>
BEGIN
<algoritmo>
END;
$$ LANGUAGE 'plpgsql'
```

Linguagem utilizada:
sempre utilizaremos
'plpgsql'

BANCO DE DADOS

17

Sintaxe básica



□ Exemplo:

```
CREATE FUNCTION soma(a int, b int) RETURNS int AS
$$
DECLARE
    res int;
BEGIN
    res:= a + b;
    return res;
END;
$$ LANGUAGE 'plpgsql';
```

BANCO DE DADOS

18

Sintaxe básica

Exemplo:

```
CREATE FUNCTION soma(a int, b int) RETURNS int AS
$$
DECLARE
    res int;
BEGIN
    res:= a + b;
    return res;
END;
$$ LANGUAGE 'plpgsql';
```

```
--Uso:
SELECT soma(5,9);
```

BANCO DE DADOS

19

Sintaxe básica

Comentários:

```
--comentário de linha
/*comentário
De linhas */
```

Atribuição:

```
x := 10;
```

Escrita de dados:

```
RAISE NOTICE 'string';
```

Caractere '%' → define uma variável a ser impressa

Exemplo: RAISE NOTICE 'valor de x: %', x

BANCO DE DADOS

20

Sintaxe básica

Tipos de dados: mesmos do postgresSQL (tipos de colunas)

Dclaração de tipo baseado em tabela:

%TYPE → tipo baseado em outro tipo (coluna de uma tabela por exemplo)

```
v_cpf cliente.cpf%TYPE;
```

%ROWTYPE: o registro vai ter os mesmos campos (com seus tipos) de uma tabela

```
rec_usuario cliente%ROWTYPE;
--(cod number, nome varchar(100))
```

BANCO DE DADOS

21

Sintaxe básica

Exemplo:

```
CREATE OR REPLACE FUNCTION func1() RETURNS void AS
$$
DECLARE
    p pessoa%ROWTYPE;
    nom pessoa.nome%TYPE;
BEGIN
    p.nome:='tiago';
    nom := p.nome;
    p.idade:=30;
    RAISE notice 'nome % idade: %', nom, p.idade;
END;
$$ LANGUAGE 'plpgsql';
```

BANCO DE DADOS

22

Controle de fluxo - condicional

IF/Then/Else

Sintaxe:

IF condição THEN comandos ELSE comandos END IF

ou

IF condição THEN comandos ELSIF comandos ... ELSE comandos END IF

```
IF x>30 THEN
    x:=x-1;
ELSE
    x:=x+1;
END IF;
```

```
IF x>30 THEN
    x:=x-1;
ELSIF x=10
    x:=0;
ELSE
    x:=x+1;
END IF;
```

BANCO DE DADOS

23

Controle de fluxo - repetição

Loop

Sintaxe:

LOOP comandos EXIT WHEN condição; END LOOP

```
LOOP
    x:=x+1;
EXIT WHEN x=18;
END LOOP;
```

BANCO DE DADOS

24

Controle de fluxo - repetição



Loop

Sintaxe:

- LOOP comandos EXIT WHEN condição; END LOOP;

While

Sintaxe:

- WHILE condição LOOP comandos; END LOOP;

```
WHILE x<18 LOOP
  x:=x+1;
END LOOP;
```

BANCO DE DADOS

25

Controle de fluxo - repetição



Loop

Sintaxe:

- LOOP comandos EXIT WHEN condição; END LOOP;

While

Sintaxe:

- WHILE condição LOOP comandos; END LOOP;

For

Sintaxe:

- FOR contador IN início..fim LOOP comandos;END LOOP;

```
FOR i IN 1..10 LOOP
  x:=x+1;
END LOOP;
```

BANCO DE DADOS

26

SQL embutida



- Blocos PL/pgSQL podem conter comandos SQL (DML)

- insert, update e delete → com valores de variáveis definidas no bloco

- As variáveis, também podem receber valores a partir de um select

- Cláusula into → para retorno de uma linha

```
DECLARE
  x number;
BEGIN
  SELECT idade INTO x FROM pessoa WHERE codpessoa=3;
END
```

Ou:

```
SELECT idade FROM pessoa WHERE cod=3 INTO x;
```

Ou:

```
x:= idade FROM pessoa WHERE cod=3;
```

- FOR → para ler linha a linha um retorno de select

LAB08 - REVISÃO SQL E FUNÇÕES

27

SQL embutida



- Blocos PL/pgSQL podem conter comandos SQL (DML)

- insert, update e delete → com valores de variáveis definidas no bloco

- As variáveis, também podem receber valores a partir de um select

- Cláusula into → para retorno de uma linha

```
DECLARE
  x number;
BEGIN
  SELECT idade INTO x FROM pessoa WHERE codpessoa=3;
END
```

Opção
Mais
utilizada

- Strict: quando a select deve retornar apenas UMA linha

```
SELECT idade INTO STRICT x FROM pessoa WHERE  
cod=3;
```

LAB08 - REVISÃO SQL E FUNÇÕES

28

SQL embutida



- Blocos PL/pgSQL podem conter comandos SQL (DML)

- insert, update e delete → com valores de variáveis definidas no bloco

- As variáveis, também podem receber valores a partir de um select

- Cláusula into → para retorno de apenas uma linha

- FOR → para ler linha a linha um retorno de select

- Pode ser usado variável record genérica

```
declareI
  x record;
begin
  for x in select * from pessoa loop
    ...
  end loop;
end
```

LAB08 - REVISÃO SQL E FUNÇÕES

29

SQL embutida



- Blocos PL/pgSQL podem conter comandos SQL (DML)

- insert, update e delete → com valores de variáveis definidas no bloco

- As variáveis, também podem receber valores a partir de um select

- Cláusula into → para retorno de apenas uma linha

- FOR → para ler linha a linha um retorno de select

- Pode ser usado variável record genérica

```
declareI
  x record;
begin
  for x in select * from pessoa loop
    ...
  end loop;
end
```

Equivalente:

```
-SELECT * INTO x FROM  
pessoa WHERE cod =1  
...  
-SELECT * INTO x FROM  
pessoa WHERE cod =2  
...  
E assim sucessivamente
```

LAB08 - REVISÃO SQL E FUNÇÕES

30

SQL embutida



Exemplo:

```
CREATE OR REPLACE FUNCTION func() RETURNS varchar AS
$$
DECLARE
  d disciplina%ROWTYPE;
  aux varchar:= '';
BEGIN
  FOR d IN SELECT * FROM disciplina LOOP
    aux:= aux||d.cod||'|'||d.nome||chr(10);
  END LOOP;
  RETURN aux;
END;
$$ LANGUAGE 'plpgsql';
```

Fim de linha

LAB08 - REVISÃO SQL E FUNÇÕES

31

Controle de erros



Um erro gera uma exceção que pode ser tratada em um bloco EXCEPTION

```
EXCEPTION
WHEN <tipo exceção1> THEN <tratamento 1>
WHEN <tipo exceção2> THEN <tratamento 2> ...
WHEN OTHERS THEN <tratamento genérico >
```

```
declare
  x int;
begin
  x:= 2/0;
  EXCEPTION
    WHEN division_by_zero then
      RAISE NOTICE 'divisão por zero';
    WHEN others then
      RAISE NOTICE 'outra exceção';
end
```

LAB08 - REVISÃO SQL E FUNÇÕES

32

Controle de erros



Disparar um erro:

RAISE ERROR <mensagem>

- Equivalente ao RAISE MESSAGE mas gerando um erro com a mensagem especificada

```
CREATE OR REPLACE FUNCTION div (a int, b int) RETURNS
float AS
BEGIN
  IF (b=0) THEN
    RAISE ERROR 'divisao por zero!';
  ELSE
    RETURN a/b;
  end
```

LAB08 - REVISÃO SQL E FUNÇÕES

33