

Introdução à PL/pgSQL

TIAGO G MORAES

Roteiro



- oIntrodução
- oSintaxe 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 oextensão da linguagem SQL para o PostgreSQL
- □É uma linguagem de programação estruturada oune 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
- o Procedimentos armazenadas (*stored procedures*) → função que retorna void
- o 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
- o Exemplo: regras para conceder um seguro de vida

□diversos softwares que utilizem essa bæe 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
- o 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 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
- o Exemplo: regras para conceder um seguro de vida
- □diversos softwares que utilizem essa base de dados



Introdução Sintaxe básica □Uma função/procedimento pode conter: □Sintaxe básica para criação de uma função: oNome da função Parâmetros CREATE FUNCTION <nome> (lista de parâmetros) RETURNS • retorno <tipo_retorno> AS Declaração das variáveis; DECLARE <var's> BEGIN <algoritmo> •Código do programa; \$\$ LANGUAGE 'plpgsql' oLinguagem utilizada Sintaxe básica Sintaxe básica □Sintaxe básica para criação de uma função: □Sintaxe básica para criação de uma função: CREATE FUNCTION <nome> (lista de parâmetros) RETURNS CREATE FUNCTION <nome> (lista de parâmetros) RETURNS <tipo_retorno > AS <tipo_retorno > AS \$\$ \$\$ Cria uma função CREATE OR REPLACE → salva por cima se nome já salvo DECLARE DECLARE Nome da <var's> <var's> função BEGIN BEGIN <algoritmo> <algoritmo> \$\$ LANGUAGE 'plpgsql' \$\$ LANGUAGE 'plpgsql' Sintaxe básica Sintaxe básica □Sintaxe básica para criação de uma função: □Sintaxe básica para criação de uma função: CREATE FUNCTION <nome> (lista de parâmetros) RETURNS CREATE FUNCTION <nome> (lista de parâmetros) RETURNS <tipo_retorno > AS <tipo_retorno > AS Parâmetros da função. Tipo de retorno. DECLARE DECLARE Exemplo: idade int <var's> <var's> Exemplo: int <algoritmo> <algoritmo> \$\$ LANGUAGE 'plpgsql' \$\$ LANGUAGE 'plpgsql'

Sintaxe básica



□Sintaxe básica para criação de uma função:

BANCO DE DADOS 13

Sintaxe básica



□Sintaxe básica para criação de uma função:

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

$$

Declaração de Variáveis.
Exemplo: aux int;

$$ LANGUAGE 'plpgsql'

ATENÇÃO: Os parâmetros não são redeclarados!!
```

BANCO DE DADOS 1.

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 'plpgsq1'
```

NANCO 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'
```

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 <align="right">sempre utilizada: sempre utilizaremos 'plpgsql'

END;
$$ LANGUAGE 'plpgsql'
```

BANCO DE DADOS 17

Sintaxe básica

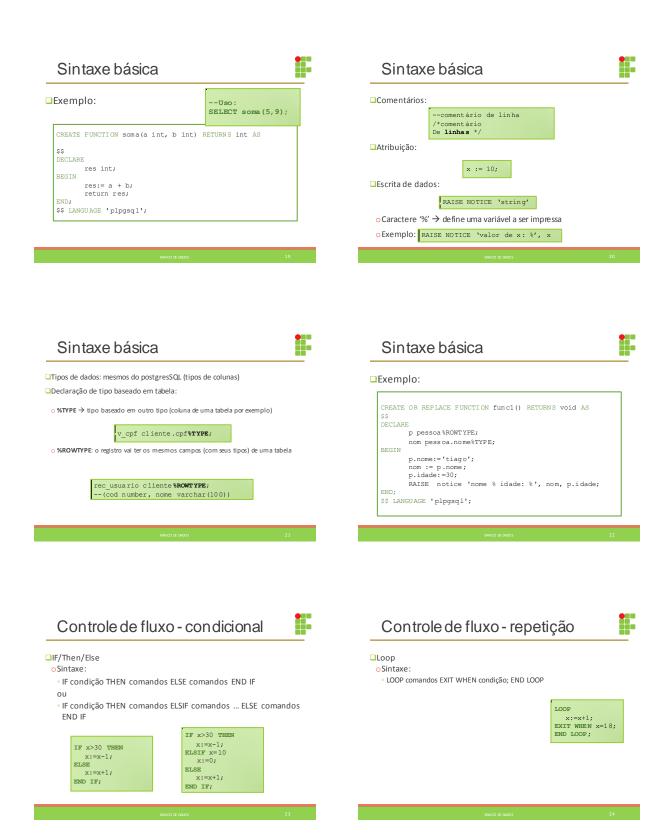


□Exemplo:

```
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

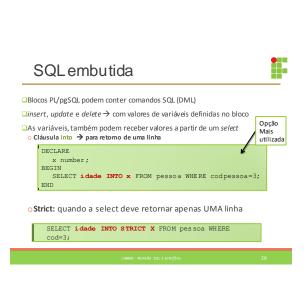


FOR i IN 1..10 LOOP
 x:=x+1;
END LOOP;









Controle de fluxo - repetição

· LOOP comandos EXIT WHEN condição; END LOOP;

WHILE condição LOOP comandos; END LOOP;

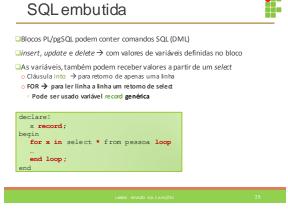
FOR contador IN início..fim LOOP comandos;END LOOP;

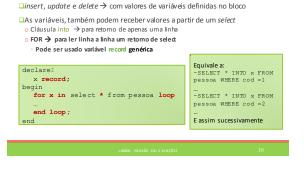
□Loop oSintaxe:

□While

□For ○Sintaxe:

Sintaxe:





SQL embutida

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

SQLembutida



■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;
$$ LANGUAGE 'plpgsq1';

Fim de linha
```

LABBON - REVISÃO SOL E JUNÇÕES 31

Controle de erros

end



□Um erro gera uma exceção que pode sertratada em um bloco EXCEPTION EXCEPTION WHEN <tipo exceção1> THEN <tratamento 1>

```
when ctipo exceção2 THEN ctratamento 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';
```

LARROW PRINCES FOLE HANGAGE

Controle de erros



□Disparar um erro:

RAISE ERROR <mensagem>

o 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
```

LABBDII - REVISÃO SQL E JUNÇÕE

33