PROCEDIMENTOS ARMAZENADOS

Explorando o poder do servidor de Banco de Dados

Prof. Rangel Nunes

INSTITUTO FEDERAL
Piauí

O QUE SÃO STORED PROCEDERES?

São procedimentos escritos em SQL ou em outras linguagens que são armazenados no banco de dados a fim de executar uma determinada função

No PostgreSQL, estes procedimentos são chamados de ***Functions ou Procedures***

O QUE PODEMOS FAZER COM ELAS?

- ✓ Consultar e retornar valores
- ✓ Realizar cálculos e retornar ou não valores
- ✓ E muito mais

LINGUAGENS PROCEDURAIS

- O PostgreSQL permite que as funções definidas pelo usuário sejam escritas em outras linguagens além de SQL
- Estas linguagens são chamadas genericamente de linguagens procedurais (PLs)
- No caso de uma função escrita em uma linguagem procedural, o servidor de banco de dados não possui nenhum conhecimento interno sobre como interpretar o código fonte da função
- Em vez disso, a tarefa é passada para um tratador especial que conhece os detalhes da linguagem

A LINGUAGEM PLPGSQL

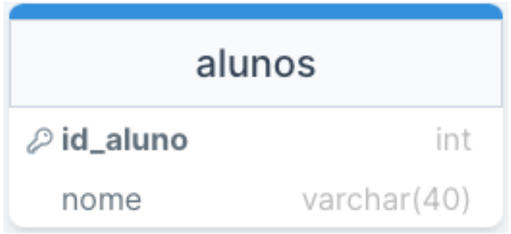
- Os objetivos de projeto da linguagem PL/pgSQL foram no sentido de criar uma linguagem procedural carregável que pudesse:

  - ✓ Adicionar estruturas de controle à linguagem SQL
  - ✓ Ser utilizada para criar procedimentos de funções e de gatilhos
  - ✓ Realizar processamentos complexos
  - ✓ Herdar todos os tipos de dado, funções e operadores definidos pelo usuário
  - ✓ Ser fácil de utilizar.

SINTAXE DE FUNCTION PLPGSQL

```
create [or replace] function nome_da_função(lista_de_parâmentros)
returns tipo_de_retorno as
$$
declare
    -- variáveis
begin
    -- lógica
end;
$$
language plpgsql;
```

UMA PEQUENA DEGUSTAÇÃO




```
create table if not exists alunos(
    id_aluno int primary key,
    nome varchar(40) not null
);

insert into alunos values(1, 'Maria Eduarda');
insert into alunos values(2, 'João Lucas');
```

NOSSA PRIMEIRA FUNÇÃO PLPGSQL

```
create or replace function get_nome_aluno(id int)
returns varchar as
$$
declare
    nome_aluno varchar;
begin
    select nome into nome_aluno from alunos where id_aluno = id;
    return nome_aluno;
end;
$$
language plpgsql;
```

ASSINATURA DE UMA FUNÇÃO PLPGSQL

```
create or replace function get_nome_aluno(id int)
returns varchar as
$$
declare
    nome_aluno varchar;
begin
    select nome into nome_aluno from alunos where id_aluno = id;
    return nome_aluno;
end;
$$
language plpgsql;
```

As funções escritas em PL/pgSQL aceitam como argumento qualquer tipo de dado suportado pelo servidor e podem retornar como resultado qualquer um destes tipos

INSTITUTO FEDERAL Piauí

PAPO DE PROGRAMADOR

O CIFRÃO DUPLO

```
create or replace function get_nome_aluno(id int)
returns varchar as
$$
declare
    nome_aluno varchar;
begin
    select nome into nome_aluno from alunos where id_aluno = id;
    return nome_aluno;
end;
$$
language plpgsql;
```

Os $$ são usados para limitar o corpo da função, e para o banco de dados entender que tudo o que está dentro dos limites do cifrão duplo é código de uma única função

A SEÇÃO DECLARE

```
create or replace function get_nome_aluno(id int)
returns varchar as
$$
declare
    nome_aluno varchar;
begin
    select nome into nome_aluno from alunos where id_aluno = id;
    return nome_aluno;
end;
$$
language plpgsql;
```

Todas as variáveis que você precisar utilizar deverão ser declaradas logo abaixo do declare

OUTROS EXEMPLOS DE DECLARAÇÃO DE VARIÁVEIS

```
-- variáveis

quantidade_estoque int DEFAULT 10;

url varchar := 'meusite.com';

codigo_turma CONSTANT integer := 2024;
```

A SEÇÃO BEGIN..END

```
create or replace function get_nome_aluno(id int)
returns varchar as
$$
declare
    nome_aluno varchar;
begin
    select nome into nome_aluno from alunos where id_aluno = id;
    return nome_aluno;
end;
$$
language plpgsql;
```

Seção onde incluímos a lógica da nossa função

CHAMANDO UMA FUNÇÃO PLPGSQL

```
select nome_da_função(parametros)

-- no nosso exemplo:
select get_nome_aluno(1);
```

O TIPO ESPECIAL RECORD

- As variáveis RECORD não possuem uma estrutura pré-definida
- Assumem a estrutura da linha para a qual são atribuídas pelo comando SELECT ou FOR
- A subestrutura da variável registro pode mudar toda vez que é usada em uma atribuição

EXEMPLO DE USO DE VARIÁVEL DO TIPO RECORD

```
create or replace function get_alunos(id int) returns varchar as
$$
    declare
        registro record;
    begin
        select * into registro from alunos where id_aluno = id;
        return registro.nome;
    end;

$$
language plpgsql;
```

OUTROS TIPOS ESPECIAIS DE VARIÁVEIS

```
-- tipos especiais

registro_da_tabela nome_da_tabela%ROWTYPE;

nome_da_variavel nome_da_tabela.nome_da_coluna%TYPE;
```

EXEMPLO DE USO

```
create or replace function get_alunos(id int) returns varchar as
$$
    declare
        registro alunos%rowtype;
    begin
        select * into registro from alunos where id_aluno = id;
        if found then
            return registro.nome;
        else
            raise notice 'Nãio existe aluno cadastrado com este id';
        end if;
    end;

$$
language plpgsql;
```

A VARIÁVEL FOUND

A variável especial FOUND pode ser verificada imediatamente após a instrução SELECT INTO para determinar se a atribuição foi bem-sucedida, ou seja, se foi retornada pelo menos uma linha pela consulta ou não

Ela também pode ser utilizada após um INSERT, DELETE ou UPTADE

EXEMPLO DE USO DA VARIÁVEL FOUND

```
create or replace function get_alunos(id int) returns varchar as
$$
    declare
        registro alunos%rowtype;
    begin
        select * into registro from alunos where id_aluno = id;
        if found then
            return registro.nome;
        else
            raise notice 'Nãio existe aluno cadastrado com este id';
        end if;
    end;

$$
language plpgsql;
```

ESTRUTURAS CONDICIONAIS

```
create or replace function get_alunos(id int) returns varchar as
$$
    declare
        registro alunos%rowtype;
    begin
        select * into registro from alunos where id_aluno = id;
        if found then
            return registro.nome;
        else
            raise notice 'Nãio existe aluno cadastrado com este id';
        end if;
    end;

$$
language plpgsql;
```

INSTITUTO FEDERAL Piauí

PAPO DE PROGRAMADOR

ESTRUTURAS CONDICIONAIS

A linguagem PL/pgSQL possui cinco formas diferentes

```
IF ... THEN

IF ... THEN ... ELSE

IF ... THEN ... ELSE IF

IF ... THEN ... ELSIF ... THEN ... ELSE

IF ... THEN ... ELSEIF ... THEN ... ELSE
```

E SE VOCÊ QUISER EXECUTAR UMA CONSULTA QUE NÃO RETORNE NENHUM RESULTADO?

INSTITUTO FEDERAL
Piauí

PAPO DE PROGRAMADOR

PERFORM

```
create or replace function exclui_aluno(id int) returns void as
$$
    begin
        perform * from alunos where id_aluno = id;
        if not found then
            raise exception 'aluno não encontrado!';

        else
            delete from alunos where id_aluno = id;
            raise notice 'aluno excluído com sucesso!';
        end if;
    end;

$$
language plpgsql;
```