

PostgreSQL - Funções

Banco de Dados I

Fontes:

<https://www.postgresql.org/docs/9.3/static/xfunc.html>

<https://www.postgresql.org/docs/9.3/static/xfunc-sql.html>

<https://www.postgresql.org/docs/9.3/static/plpgsql.html>

PostgreSQL

- Quatro tipos de funções
 - Linguagem de consulta (SQL)
 - Linguagem procedural (PL/pgSQL)
 - Funções Internas
 - Linguagem C
-

PostgreSQL

□ Funções SQL

- Executam comandos SQL
 - Retorno
 - Nada (void)
 - Um único valor
 - Resultado de uma expressão algébrica ou função agregada (exemplo: count)
 - Conjunto de valores
 - Pode ser usado o operador Setof
 - Pode retornar uma tabela com Returns table (columns)
 - Chamada da função
 - Select "nome da funcao (parâmetros – se for o caso)"
-

PostgreSQL

- ❑ Corpo da função deve ser escrito como uma cadeia de caractere delimitado por ` `

CREATE FUNCTION nomefuncao

(tipop nomep tipodado)

RETURNS tipodado AS ' codigo SQL' LANGUAGE SQL;

tipop: tipo de parâmetro (IN, OUT, INOUT)

nome: nome do parâmetro

Tipodado: tipo do dado atribuído ao parâmetro ou tipo de retorno da função (função pode ser void)

departamento(codigo,nome)
funcionário(codigo,nome,codept,salario)
codept referencia departamento(código)
projeto(codigo,nome,descrição,datai,dataf)
alocação(codf,cofp,horas,datai)
codp referencia projeto (código) , codf referencia funcionário (código)

□ Exemplo

Retorne o departamento de um funcionário que possui um determinado código

funcionário(codigo,nome,codept,salario)

```
CREATE FUNCTION primeira () RETURNS int AS  
'select codept from funcionario where codigo = 1'  
LANGUAGE SQL
```

Para testar a função: Select primeira()

departamento(codigo,nome)
funcionário(codigo,nome,codept,salario)
codept referencia departamento(código)
projeto(codigo,nome,descrição,datai,dataf)
alocação(codf,cofp,horas,datai)
codp referencia projeto (código) , codf referencia funcionário (código)

□ Exemplo

Retorne o departamento de um funcionário que possui um determinado código que deve ser passado como parâmetro.

funcionário(codigo,nome,codept,salario)

CREATE OR REPLACE FUNCTION primeira (cod int)

RETURNS int as 'select codept from funcionario f

where f.codigo = cod' **LANGUAGE SQL**

Para testar a função: Select primeira(1)

departamento(codigo,nome)
funcionario(codigo,nome,codept,salario)
codept referencia departamento(código)
projeto(codigo,nome,descrição,datai,dataf)
alocação(codf,cofp,horas,datai)
codp referencia projeto (código) , codf referencia funcionario (código)

□ Delimitador \$

- Usado para facilitar a elaboração do código quando o delimitador ` ` precisa ser usado na consulta como no exemplo abaixo:

funcionario(codigo,nome,salario,codept)

CREATE FUNCTION primeira () RETURNS int AS

`select codept from funcionario where nome = 'Joao'`


LANGUAGE SQL;

Para testar a função: Select primeira()


PostgreSQL

□ Delimitador \$

- O delimitador ` pode ser substituído pelo delimitador \$ no corpo da função. O \$ pode ser combinado com outros caracteres como nos exemplos abaixo:

CREATE FUNCTION nomefuncao \\ ` substituído por \$\$  **delimitador \$\$**
(tipop nomep tipodado)

RETURNS tipodado AS \$\$ codigo SQL \$\$ LANGUAGE SQL;

CREATE FUNCTION nomefuncao \\ ` substituído por \$funcao\$  **delimitador \$funcao\$**
(tipop nomep tipo)

RETURNS tipo AS \$funcao\$ codigo SQL \$funcao\$ LANGUAGE SQL;

PostgreSQL

❑ Delimitador \$

- No exemplo abaixo, o ` foi substituído por \$ que foi combinado com outro \$, criando o delimitador \$\$

funcionario(codigo,codept,nome,salario)

CREATE FUNCTION primeira () RETURNS int AS

\$\$ select codept from funcionario where nome =
'Joao' \$\$

LANGUAGE SQL;

PostgreSQL

❑ Delimitador \$

- A mesma função do exemplo anterior com parâmetro

funcionario(codigo,codept,nome,salario)

```
CREATE OR REPLACE FUNCTION primeira(nm  
character varying) RETURNS int $$ select codept  
from funcionario where nome = nm $$ LANGUAGE  
SQL
```

Para testar a função: select primeira('Bia')

PostgreSQL - Exemplos

funcionario(codigo,codept,nome,salario)
departamento(codigo,nome)

- ❑ Como o retorno é um valor int, só retorna o código do primeiro funcionário

create function consulta () returns int as

\$\$ select codigo from funcionario \$\$

LANGUAGE SQL;

**// RETORNA SOMENTE O CÓDIGO DO PRIMEIRO FUNCIONARIO
CADASTRADO**

PostgreSQL

funcionario(codigo,codept,nome,salario)
departamento(codigo,nome)

❑ Retorna o código do primeiro departamento

create function consulta () returns int as

\$\$ select codigo from funcionario; select codigo from departamento \$\$

LANGUAGE SQL;

// RETORNA O DADO DA PRIMEIRA TUPLA DA ÚLTIMA CONSULTA

PostgreSQL

- ❑ Argumentos podem ser referenciados através da posição \$n ou pelo nome

create function removefuncionario (cod int)
returns void as

\$\$ delete from funcionario where codigo = \$1 \$\$

LANGUAGE SQL;

create function removefuncionario (cod int)
returns void as

\$\$ delete from funcionario where codigo = cod \$\$

LANGUAGE SQL;

PostgreSQL

- ❑ Caso tenha conflito entre o nome do argumento e o nome de uma coluna, o nome do argumento pode ser qualificado pelo nome da função

create function removefuncionario (codigo
int) returns void as \$\$

delete from funcionario where codigo =
removefuncionario.codigo

\$\$ LANGUAGE SQL;

PostgreSQL

- ❑ Os parâmetros não precisam ser nomeados

create or replace function
removefuncionario (int) returns void
as

\$\$ delete from funcionario where codigo = \$1 \$\$
LANGUAGE SQL;

PostgreSQL

- Tipo de retorno deve ser idêntico ao campo na tabela (verificar o create table)

**create or replace function alterafuncionario
(cod int, taxa real) returns real as**

```
$$ update funcionario set salario = salario + (salario*taxa)/100
```

```
where codigo = cod;
```

```
select salario from funcionario where codigo = cod $$
```

language SQL

PostgreSQL

- ❑ O select pode ser usado para retornar alguma mensagem ou um valor para confirmar uma operação

create or replace function atualizafuncionario (in cod int,
in sal double precision) returns text as

```
$$ update funcionario set salario = sal where codigo =  
cod;
```

```
select 'Novo salario:' || salario from funcionario where  
codigo = cod $$
```

language sql

PostgreSQL

- ❑ O *returning* também pode ser usado para retornar valores

create or replace function removefuncionario (in cod int)
returns varchar as

\$\$ delete from funcionario where codigo = cod
returning nome \$\$

language sql

returning com insert retorna os dados inseridos
returning com delete retorna os dados removidos
returning com update retorna os dados atualizados

PostgreSQL

funcionario(codigo,codept,nome,salario)

departamento(codigo,nome)

- ❑ Função pode retornar um registro de uma tabela (tipo composto)

create or replace function retornacodept (cod int)
returns departamento as

\$\$ select * from departamento where codigo = cod \$\$

language sql

//Pode ser usado no from: select nome from retornacodept(1)

PostgreSQL

funcionario(codigo,codept,nome,salario)

departamento(codigo,nome)

- ❑ Funções podem ser usadas no from e no where
 - Select nome from retornacodept(1)
 - ❑ A função retornacodept(1) retorna uma tuple da tabela departamento cujo código = 1, o select retorna somente o nome dessa tupla
 - Select nome from departamento where codigo = retornadept (3)
 - ❑ A função retornadept(3) retorna o código do departamento do funcionário cujo código = 3, o select retorna o nome desse departamento
-

PostgreSQL

funcionario(codigo,codept,nome,salario)

departamento(codigo,nome)

- ❑ Função pode retornar um registro qualquer
 - ❑ record: tipo de retorno que corresponde a um registro (várias colunas) não necessariamente idêntico a um registro em uma tabela
-

PostgreSQL

RETORNA O NOME DO FUNCIONÁRIO E DO DEPARTAMENTO DO FUNCIONÁRIO COM CÓDIGO = COD

create or replace function retornacodeptf
(cod int) returns record as

```
$$ select f.nome, d.nome from funcionario f ,  
departamento d where f.codept = d.codigo and  
f.codigo = cod $$
```

language sql

PostgreSQL

funcionario(codigo,codept,nome,salario)
departamento(codigo,nome)

- ❑ Função pode retornar um conjunto de valores (setof)

create function consulta () returns setof int as

\$\$ select codigo from funcionario \$\$

LANGUAGE SQL;

// RETORNA O CÓDIGO DE TODOS FUNCIONARIOS

PostgreSQL

❑ Retornar vários registros

RETORNA O (s) NOME (s) DO (s) FUNCIONÁRIO (s) E DO DEPARTAMENTO DO (s) FUNCIONÁRIO (s) COM SALÁRIO > SAL

create function consultadadossal (sal double precision) returns setof record as

\$\$ select f.nome, d.nome from funcionario f ,
departamento d where f.codept = d.codigo and
salario > sal \$\$

LANGUAGE SQL;

PostgreSQL

- ❑ O retorno pode ser definido no parâmetro de saída

RETORNA O CÓDIGO E O NOME DO PRIMEIRO REGISTRO DE FUNCIONARIO (usando parâmetro de saída)

create function consultados (out codigo int, out nome text) returns record as

\$\$ select codigo,nome from funcionario \$\$

LANGUAGE SQL;

PostgreSQL

❑ Retornar várias colunas

RETORNA O CÓDIGO E O NOME DE TODOS OS REGISTROS DE FUNCIONARIOS

create function consultados (out codigo int, out nome text) returns setof record as

\$\$ select codigo,nome from funcionario \$\$

LANGUAGE SQL;

//Como foram definidos mais de um parâmetro de saída, o tipo de retorno deve ser record

PostgreSQL

Retorne o nome dos funcionários de um departamento

create function consultados (codept text)
returns setof text as \$\$

Select f.nome from funcionario f join departamento d on
f.codept = d.codigo and d.nome = \$1

\$\$ LANGUAGE SQL;

Retorne o nome do departamento e dos funcionários de cada departamento

Select nome,consultados(nome) from
departamento

Esquema Empresa

funcionário(codigo,nome,codept, ...)

projeto(codigo,nome, ...)

alocação(codf,cofp,horas, ...)

1. Elabore uma função que receba o código do funcionário, o código do projeto e um novo valor de horas e altere quantidade de horas que o funcionário está alocado no projeto. A função deve retornar o novo valor.
 2. Elabore uma função que receba um código de projeto retorne a quantidade de funcionários que trabalham no projeto e a quantidade total de horas desses funcionários no projeto
 3. Elabore uma função que retorne o nome do funcionário e a quantidade de projetos em que cada funcionário trabalha
-

Esquema Empresa

funcionário(codigo,nome,codept, ...)

projeto(codigo,nome, ...)

alocação(codf,cofp,horas, ...)

1. Elabore uma função que receba o código do funcionário, o código do projeto e um novo valor de horas e altere quantidade de horas que o funcionário está alocado no projeto. A função deve retornar o novo valor.

create or replace function alterahoras(pcodf int, pcodp
int, phoras double precision)

returns double precision as \$\$

update alocao set horas = phoras where codf =
pcodf and codp = pcodp

returning horas

\$\$ language SQL

Esquema Empresa

funcionário(codigo,nome,codept, ...)

projeto(codigo,nome, ...)

alocação(codf,cofp,horas, ...)

1. (MODIFICADA) Elabore uma função que receba o código do funcionário, o código do projeto e um novo valor de horas e altere quantidade de horas que o funcionário está alocado no projeto. **A função deve retornar os dados do registro atualizado**

create or replace function alterahoras2(pcodf int,
pcodp int, phoras double precision)

returns record as \$\$

update alocacao set horas = phoras where codf =
pcodf and codp = pcodp

returning *

\$\$ language SQL

Esquema Empresa

funcionário(codigo,nome,codept, ...)

projeto(codigo,nome, ...)

alocação(codf,cofp,horas, ...)

2. Elabore uma função que receba um código de projeto retorne a quantidade de funcionários que trabalham no projeto e a quantidade total de horas desses funcionários no projeto.

create or replace function exer2 (pcodp int)

returns record as

```
$$ select count(codf), sum(horas)from alocacao
```

```
where codp = pcodp $$
```

```
language SQL
```

Esquema Empresa

funcionário(codigo,nome,codept, ...)

projeto(codigo,nome, ...)

alocação(codf,cofp,horas, ...)

3. Elabore uma função que retorne o nome do funcionário e a quantidade de projetos em que cada funcionário trabalha

create or replace function exer3 () returns setof record as

```
$$ select f.nome, count(a.codp) from funcionario f
join alocao a on f.codigo = a.codf group by
(f.nome)$$
```

language SQL

PostgreSQL - Funções com linguagens procedurais

- PL/pgSQL
 - Declarações (declare)
 - Comandos (begin-end)
 - Finalizados com ;

CREATE OR REPLACE FUNCTION nome(parametros) RETURNS
tipoRetorno AS \$\$

DECLARE declarations;

BEGIN

 statements;

END;

\$\$ LANGUAGE plpgsql

PostgreSQL - PL/pgSQL

- ❑ CREATE OR REPLACE FUNCTION acrescimo(subtotal real, taxa real) returns real AS \$\$

```
DECLARE total real;
```

```
BEGIN
```

```
    total := subtotal + subtotal * taxa/100;
```

```
    return total;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
select acrescimo(100,10) retorna 110
```

PostgreSQL - Funções com linguagens procedurais

```
create or replace function retornavalores (x int, y int)
returns int as $$
begin
    return x + y;
end; $$ language plpgsql
```

Sem parâmetro de saída
Cláusula return(só retorna um valor) é obrigatório

```
create or replace function retornavaloresout (x int, y int, out
soma int, out prod int ) as
$$ begin
    soma := x+y;
    prod:= x*y;
end; $$ language plpgsql
```

Com parâmetro de saída
A função pode retornar mais de u valor sem especificar o tipo

PostgreSQL - Funções com linguagens procedurais

- ❑ Funções podem retornar uma tabela. Neste caso, deve ser definido um "return query"

```
create or replace function retornatable ( ) returns table  
(cod int, nome varchar) as  
$$  
begin  
    return query select f.codigo, f.nome from funcionario f ;  
end;  
$$ language plpgsql
```

PostgreSQL - Funções com linguagens procedurais

plpgsql

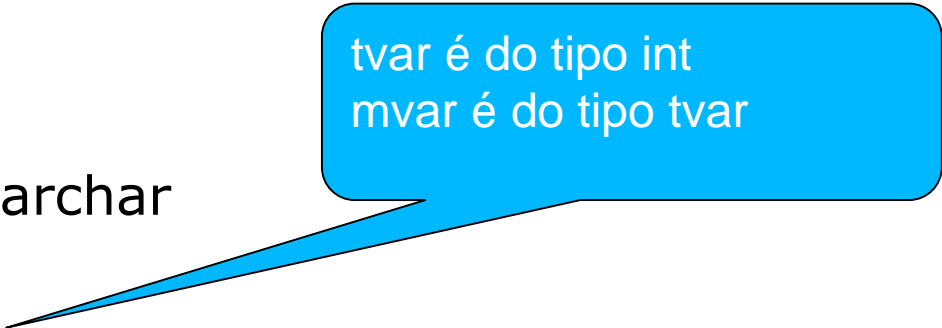
```
create or replace function retornatable ( ) returns table  
(cod int, nome varchar) as $$  
begin  
    return query select f.codigo, f.nome from funcionario f ;  
end;  
$$ language plpgsql
```

```
create or replace function retornatablesql ( ) returns table  
(cod int, nome varchar) as $$  
    select f.codigo, f.nome from funcionario f ;  
$$ language sql
```

sql

PostgreSQL - PL/pgSQL

- ❑ Atribuição :=
- ❑ Tipos de variáveis
 - integer, numeric, varchar
 - Cópia de tipo
 - ❑ tvar int, mvar tvar%type
 - ❑ mlinha tablename%ROWTYPE
 - p professor%ROWTYPE (p é uma variável que possui a mesma definição das linhas da tabela professor)
 - ❑ mcampo tablename.columnname%TYPE
 - nome professor.nome%TYPE (nome é uma variável que possui a mesma definição da coluna nome da tabela professor)
 - arrow RECORD



tvar é do tipo int
mvar é do tipo tvar

PostgreSQL - PL/pgSQL

- Tipo tablename.columnname%TYPE

create or replace function funcnome (codigo
funcionario.codigo%TYPE) returns
funcionario.nome%TYPE as \$\$

declare nm funcionario.nome%TYPE;

begin

 select nome into nm from funcionario f where f.codigo = \$1;
 return nm;

end;

\$\$ Language plpgsql

PostgreSQL - PL/pgSQL

❑ Conversão de tipos

```
CREATE OR REPLACE FUNCTION testetipo(x integer, y integer) RETURNS real AS
$$ declare z real;
begin
    z := x/y;
    return z;
end;$$
LANGUAGE plpgsql
```

select testetipo(5,2) retorna 2
Como x e y são inteiros, x/y é realizado como uma divisão inteira

```
CREATE OR REPLACE FUNCTION testetipo(x real, y real) RETURNS integer AS $$
declare z real;
begin
    z := x/y;
    return z;
end;$$
LANGUAGE plpgsql
```

select testetipo(5,2) retorna 2
O valor de retorno foi convertido para o tipo definido no returns (returns integer)

PostgreSQL - PL/pgSQL

□ Conversão de tipos

```
CREATE OR REPLACE FUNCTION testetipo(x integer, y integer) RETURNS real AS
$$ declare z real;
begin
    z := cast (x as real) /y;
    return z;
end;$$
LANGUAGE plpgsql
```

select testetipo(5,2) retorna 2,5
x é usado como valor real, logo x/y
não é realizada como divisão inteira



cast

PostgreSQL - PL/pgSQL

- Podem ser usados comandos procedurais
 - Estrutura condicional
 - IF ... THEN ... END IF
 - IF ... THEN ... ELSE ... END IF
 - IF ... THEN ... ELSIF ... THEN ... ELSE ... END IF
-

PostgreSQL - PL/pgSQL

❑ Tratamento de exceção

```
CREATE OR REPLACE FUNCTION funcnome(codigo int) RETURNS
varchar AS $$
declare nm funcionario.nome%TYPE;
begin
    select f.nome into nm from funcionario f where f.codigo = $1;
    IF NOT FOUND THEN
        RAISE EXCEPTION 'funcionario% não encontrado', codigo;
    END IF;
return nm;
end; $$ LANGUAGE plpgsql
```

PostgreSQL - PL/pgSQL

□ Estruturas de repetição

WHILE boolean-expression LOOP
statements

END;

FOR i IN 1..10 LOOP
statements

END LOOP;

FOR i IN REVERSE 10..1 LOOP
statements

END LOOP;

PostgreSQL - PL/pgSQL

- Estruturas de repetição

FOR variavel IN consulta LOOP
statements

END LOOP;

- variavel deve ser declarada e deve ser do mesmo tipo do valor de retorno da consulta
 - record, rowtype ou uma lista de tipos esclares separados por vírgula
 - Cada iteração do loop, a variavel vai apontar/assumir os valores de uma linha da consulta
-

Elabore uma função que receba um valor como parâmetro que representa quantidade máxima de funcionários permitida e verifique a quantidade de funcionário de cada departamento. Caso o departamento tenha mais funcionários que o permitido, a função deve exibir uma mensagem. Ao final, a função deve retornar a quantidade de departamentos que possuem mais que o valor permitido.

```
create or replace function testefuncao1 (valor int) returns int as $$
declare
    d departamento%rowtype; qtde int; cont int = 0;
begin
    for d in select * from departamento loop
        select count(f.codigo) into qtde from funcionario f where codept = d.codigo;
        if (qtde > valor) then
            raise notice 'O departamento % possui mais funcionarios que o
permitido', d.codigo;
            cont := cont +1;
        end if;
    end loop;
    return cont;
end;
$$
language plpgsql
```

Elabore uma função para atualizar a quantidade de funcionários dos departamentos cuja quantidade = 0

```
create or replace function alteraqtde ( ) returns void as $$
declare codept departamento%rowtype;
BEGIN
  for codept in select * from departamento
  loop
    if codept.qtdef = 0 then
      update departamento set qtdef = (select count(f.codigo) from funcionario f where
f.codept = codept.codigo)
      where departamento.codigo = codept.codigo;
    end if;
  end loop;
return;
END;
$$
LANGUAGE 'plpgsql';
```

Elabore uma função que receba um valor e faça a seguinte verificação: para cada professor, deve ser calculada a quantidade de turmas alocadas, caso a quantidade seja maior que o valor, deve ser exibida na tela uma mensagem.

```
CREATE OR REPLACE FUNCTION verificaprofessor(max integer)
  RETURNS text AS
$BODY$ declare p professor%ROWTYPE; qtde  int := 0;
begin
  for p in select * from professor
  loop
    select count(matriculap) into qtde from turma where matriculap = p.matricula;
    if qtde > max then
      raise notice 'O professor % possui mais turmas do que o permitido', p.matricula;
    end if;
  end loop;
end;
$BODY$
LANGUAGE plpgsql
```


PostgreSQL - Funções com linguagens procedurais

- ❑ Código pode ser estruturado em blocos

```
create or replace function testebloco () returns text as $$
```

```
<<blocoexterno>>
```

```
declare num integer := 10;
```

```
begin
```

```
    RAISE NOTICE 'Numero aqui e % ', num;
```

```
    declare num integer := 50;
```

```
    begin
```

```
        RAISE NOTICE 'Numero aqui e % ', num;
```

```
        RAISE NOTICE 'Numero no bloco externo e % ', blocoexterno.num;
```

```
    end;
```

```
end; $$
```

```
language plpgsql;
```

Saída

Numero aqui e 10

Numero aqui e 50

Numero no bloco externo e 10

Suponha que alguém tenha criado a tabela alocação sem a restrição de chave primária

Exercícios (PLPGSQL)

1. Verifique o que é feito no código abaixo:

```
create or replace function verificaalocacao ( ) returns int as $$
```

```
declare a record; cont int := 0;
```

```
begin
```

```
  for a in select count(codp) as qtde from alocação group by (codp,codf) loop
```

```
    if (a.qtde > 1) then
```

```
      cont := cont+1;
```

```
    end if;
```

```
  end loop;
```

```
return cont;
```

```
end; $$
```

```
language plpgsql
```

funcionário(codigo,nome,codept, ...)

projeto(codigo,nome, ...)

alocação(codf,cofp,horas, ...)

Suponha que ao criar a tabela turma, não tenha sido criada a restrição de chave estrangeira para a coluna codigod

professor(matricula,nome,titulacao,codept)
disciplina(codigo,nome,descricao)
turma(matriculap, codigod,semestre)

Exercícios (PLPGSQL)

Elabore as seguintes funções:

- ❑ O código de uma disciplina foi alterado, atualize a tabela turma de maneira que todas as turmas da disciplina alteradas também sejam modificadas.
- ❑ Podemos pensar em uma função SQL

```
CREATE OR REPLACE FUNCTION alteradic_t (codv integer,codn integer)
RETURNS void AS $$
    update turma
    set codigod = codn where codigod = codv
$$
LANGUAGE sq
```

professor(matricula,nome,titulacao,codept)
disciplina(codigo,nome,descricao)
turma(matriculap, codigod,semestre)

Exercícios (PLPGSQL)

Elabore as seguintes funções:

- ❑ O código de uma disciplina foi alterado, atualize a tabela turma de maneira que todas as turmas da disciplina alteradas também sejam modificadas. Emitir uma mensagem com o status da alteração (disciplina alterada ou disciplina não possui turma)
 - ❑ É necessário implementar uma função PLPGSQL
-

professor(matricula,nome,titulacao,codept)
disciplina(codigo,nome,descricao)
turma(matriculap, codigod,semestre)

Exercícios (PLPGSQL)

Elabore as seguintes funções:

- ❑ O código de uma disciplina foi alterado, atualize a tabela turma de maneira que todas as turmas da disciplina alteradas também sejam modificadas. Emitir uma mensagem com o status da alteração (disciplina alterada ou disciplina não possui turma)

```
CREATE OR REPLACE FUNCTION alteradic_t (codv integer,codn integer)
RETURNS text AS $$
BEGIN
    update turma
    set codigod = codn where codigod = codv;
    IF NOT FOUND THEN
        RAISE EXCEPTION 'Disciplina não tem turma';
    END IF;
    RETURN 'Disciplina alterada';
END;
$$ LANGUAGE plpgsql
```

professor(matricula,nome,titulacao,codept)
disciplina(codigo,nome,descricao)
turma(matriculap, codigod,semestre)

Exercícios (PLPGSQL)

Elabore as seguintes funções:

- ❑ Elabore uma função que receba o código de um professor e retorne o nome das disciplinas que ele leciona. Caso não seja encontrada turma do professor, retorne uma mensagem "Professor %codigo não possui turma cadastrada"

```
CREATE OR REPLACE FUNCTION public.verificaprof(mat_p integer) RETURNS
SETOF character varying AS $$
begin
    return query select d.nome from disciplina d join turma t on t.codigod = d.codigo
where t.matriculap = mat_p;
    IF NOT FOUND THEN
        RAISE EXCEPTION 'professor % nao tem turma', mat_p;
    END IF;
end; $$
LANGUAGE plpgsql
```

Departamento(codigo,nome)

funcionário(codigo,nome,codept,salario)

projeto(codigo,nome,descrição,datai,dataf)

alocação(codf,cofp,horas,datai)

1. Elabore uma função que receba o código de um funcionário e retorne o nome dos projetos que esse funcionários está alocado. Caso o funcionário não tenha alocação, emitir uma mensagem com essa informação.
 2. Elabore uma função que receba um valor que representa a quantidade máxima de funcionários que um projeto pode ter. Analise a quantidade de funcionários de cada projeto, caso seja maior que a quantidade permitida, emitir uma mensagem informando que o projeto com o código identificado possui mais funcionários que o permitido
-