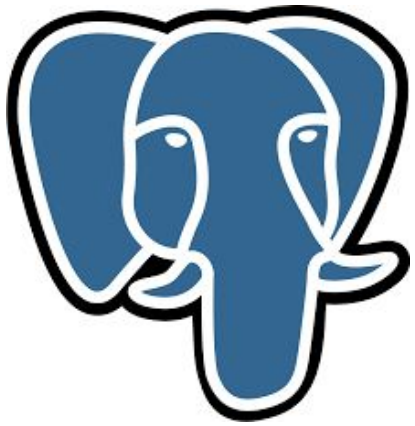


SQL Procedural

https://www.postgresql.org/docs/current/plpgsql.html

SQL Procedural

- Vantagens
  - Agrupamento de instruções
  - Redução de rodadas entre cliente e servidor
  - Redução da geração de dados intermediários
  - Melhora no performance

Aspecto de função

```
CREATE FUNCTION name (parametros)
    RETURNS tipo
    AS
    $$
    declare
    -- variáveis
    begin
    -- corpo
    end;
     $$
Language plpgsql ;
```

*Linguagem pode ser C, sql, plpgsql

Hi

Aspecto de função

```
CREATE FUNCTION name (parametros)
    RETURNS tipo
    AS
    $$
    declare
    -- variáveis
    begin
    -- corpo
    end;
     $$
Language plpgsql ;
```

cont int=0;

*Linguagem pode ser C, sql, plpgsql

Aspecto de função

```
CREATE FUNCTION name (parametros)
    RETURNS tipo
    AS
    $$
    declare
    -- variáveis
    begin
    -- corpo
    end;
     $$
Language plpgsql ;
```

Corpo da função

*Linguagem pode ser C, sql, plpgsql

Exemplo

```
CREATE or REPLACE FUNCTION nome()
RETURNS void
AS
$$
DECLARE
    quantidade int = 30;
BEGIN
    RAISE NOTICE 'Quantidade é  %', quantidade;
END;
$$
LANGUAGE plpgsql;
```

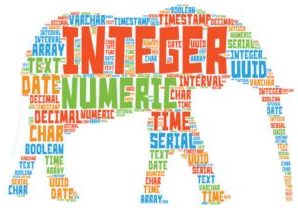
Vamos testar!

Variáveis- Tipos

- int: -/+2147483648
- numeric: 12.1245
- varchar: string
- record: comportamento similar a uma struct

*https://www.postgresql.org/docs/current/datatype.html

Variable RECORD



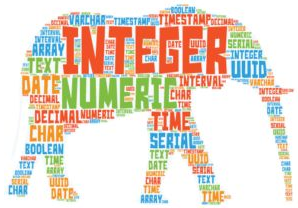
- Dinâmico
- NÃO é um tipo realmente
- Sintaxe:

  Variável RECORD;

Variável Record

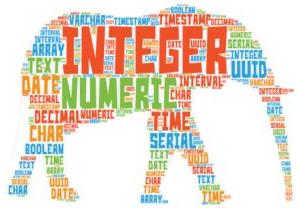

```
CREATE or REPLACE FUNCTION merge_fields()
RETURNS text
AS $$
DECLARE
    r record;
BEGIN
    SELECT * INTO r FROM dados where id = 1;
    RETURN r.nome;
END;
$$ LANGUAGE plpgsql;
```

Parâmetros




```
CREATE FUNCTION sales_tax(subtotal real)
RETURNS real
AS $$
BEGIN
    RETURN subtotal * 0.06;
END;
$$
LANGUAGE plpgsql;
```

Parâmetros-Exemplo




```
CREATE FUNCTION atualizaCurriculo2(varchar, int)
RETURNS boolean
AS$$
BEGIN
    UPDATE aluno SET curriculo = $1 WHERE num_matricula = $2;
    RETURN FOUND;
END;
$$
  LANGUAGE 'plpgsql' ;
```

Atividade

```
create table dados (name varchar(50),  salary float, id int);
```

- Insira 3 tuplas na tabela acima
- Faça uma função capaz de retornar o nome do maior id;
- O que acontece se tiver dois ids iguais (não é chave) ?

Retorno

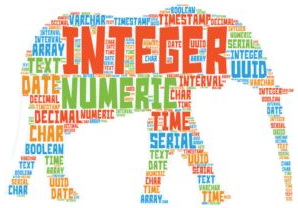


Table (args…)

- Retornar uma nova tabela;
    - Exemplo:

```
CREATE FUNCTION name()
RETURNS TABLE(var1 int,  var2 float) ;
```

Setof tablenome

- Retornar linha da tabela;
    - Exemplo:

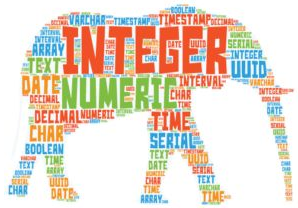
```
CREATE FUNCTION name()
RETURNS setof tableName AS $$
```

Retorno tipo Table


```
create table dados (name varchar(50),  salary float, id int);
```

```
CREATE FUNCTION selecionar(p_itemno int) RETURNS
    TABLE(name varchar(50),  salary float)
AS $$
BEGIN
   RETURN QUERY SELECT s.name, s.salary FROM dados AS s
WHERE s.id = p_itemno;
END;
$$ LANGUAGE plpgsql;
```

Retorno tipo SetOf




```
create table dados (name varchar(50),  salary float, id
int);
```

```
CREATE FUNCTION selecionar(p_itemno int)
   RETURNS setof dados AS
$$
BEGIN
   RETURN QUERY SELECT * FROM dados AS s
            WHERE s.id = p_itemno;
END;
$$
LANGUAGE plpgsql;
```

Atividade

```
create table dados (name varchar(50),  salary float, id int);
```

- Insira 3 tuplas com nomes ‘joao’
- Faça uma função capaz de retornar os nomes e ids das pessoas que tenham ‘joao’ no nome;

Comando condicional

```
IF boolean-expression THEN
    statements
ELSE
    statements
END IF;
```

Operadores
>
<
>=
<=
!=
=

Exemplo

```
create table users (id int, nome varchar(50), salario float)
```

```
CREATE OR REPLACE FUNCTION test(p int, n int)
   RETURNS setof users
AS $$
BEGIN
  IF p = 1 THEN
    RETURN query SELECT * FROM users u where u.id=n;
  ELSE
    UPDATE users  set salario=salario*1.1 where id=n;
    RETURN query SELECT * FROM users u where u.id=n;
  END IF;
  RETURN;
END;
$$ LANGUAGE plpgsql;
```

LOOPs

```
CREATE FUNCTION teste () returns void as $$
declare
        i int;
begin
        for  i in 1..100000
        loop
insert into test2  values (random(), i * random());
        end loop;
end;
$$ LANGUAGE plpgsql;
```

For exemplo

Crie uma função capaz de listar os números ma
a média do campo id1;

```
CREATE OR REPLACE FUNCTION teste () RETURNS void as $$
DECLARE
        r record;
        media float;
BEGIN



END;
$$ LANGUAGE plpgsql;

select teste();
```

For exemplo

Crie uma função capaz de listar os números ma: a média do campo id1;

```
create or replace function teste () returns void as $$
declare
        r record;   media numeric;
begin
        select avg(id1) into media from cliente;
        for  r in  select * from cliente loop
                if r.id1 > media then
                        raise notice 'é maior %',r.id1;
                end if;
        end loop;
end;
$$ LANGUAGE plpgsql;

select teste();
```

For exemplo

Criar uma função capaz de listar os números maiores que a média do campo id1;

```
create or replace function teste () returns int as $$
    declare
        temp numeric;
    begin
    select count(*) into temp from cliente where id1> (select avg(id1) from
    cliente);
        return temp;
    end;
    $$ LANGUAGE plpgsql;
select teste();
```

Atividade

1-Crie uma função capaz de incrementar um dado número;

Ex: select funcao_increment(1)

Retorno: 2

2-Crie uma função capaz de retornar um texto passado por argumento;

3- Crie uma tabela com a assinatura “usuario (id int, nome varchar(50))”. Após inserir 5 tuplas, faça uma função capaz de retornar os nomes com id maiores que a média;

Conectar: **sudo -u postgres psql postgres**

Vamos criar uma função condicional

1- Crie uma função capaz de executar uma operação de incremento de 10% de um valor, se o parâmetro inicial for 1. Caso o parâmetro inicial tenha o valor 2, a função deve decrementar 10% do valor.

Assinatura: calcula_valor(operacao,valor)

2. A partir da tabela a seguir, crie uma função capaz de atualizar o salário em 5% se o mesmo for menor que 10k e em 1% se o salário for maior que 10k.

```
create table users (id int, nome
varchar(50), salario float)
```

Questão 1 - Resposta

```
CREATE FUNCTION incremento10 (op int, valor float)
RETURNS float AS $$
begin
  if op = 1 then
    return valor*1.1;
  else
    return valor * 0.9;
  end if;
 end;
$$
Language plpgsql ;
```

Exercícios A

Crie uma tabela com a assinatura “employee (id int, name varchar(50), BirthYear int, salary float)”.

Insira 5 tuplas

Após:

A - Faça uma função capaz de aplicar um **aumento** de 10% em todos os funcionários;

B- Faça uma função capaz de aplicar um aumento de X% nos funcionários com **id maior que N**. Importante: **X e N** serão passados por argumento. O valor de x pode ser um float entre 0 e 1.

C- Faça uma função capaz de remover os funcionários com salário acima da média.

Exercícios A

D- Crie uma função que armazene o usuário corrente e a data atual ao adicionar uma nova tupla na tabela. Ex: insereDados(10,'joao',2000', 1000.00)

**current_user*- retorna o usuário atual

**current_date* - retorna a data atual

*ALTER TABLE table_name ADD column_name datatype;

Exercícios B

Utilize a base de dados DOJO

1- Faça uma função que liste os empregados que recebem mais que o seu chefe

2- Faça uma função que remova os empregados que ganham mais que o seu chefe;

3- Faça uma função que liste a média de salário de cada departamento e a média geral (mesma linha) sem incluir os supervisor_id=0;

4-Faça uma função que retorne uma nova tabela com o nome de cada funcionário, Nome Departamento e o Nome do seu superior;

Resposta letra D

```
drop table employee;
create table employee (id int, name varchar(50), BirthYear int, salary float);

alter table employee add usercurrent varchar(50);

alter table employee add datecurrent date;

--insereDados( 10,’joao’,2000’, 1000.00)

CREATE or replace FUNCTION insereDados(id int, name varchar(50), BirthYear int, salary float )
RETURNS void AS $$
```

Resposta letra D

```
BEGIN

        insert into employee values (id,name,BirthYear,salary, current_user, current_date);
        END;
        $$
LANGUAGE plpgsql;



insereDados( 10,'joao',2000, 1000.00);
```

SQL Procedural: Function vs Trigger

Triggers representam gatilhos acionados por um evento (um insert ou update)

Os gatilhos são implementados através de funções.

Exemplo 1

```
CREATE OR REPLACE FUNCTION incremento (valor int) returns int AS
$$
declare
  NovoValor int;
begin
    NovoValor=valor+1;
    return NovoValor;
end;
$$
LANGUAGE 'plpgsql';
```

Exemplo 1

```
CREATE OR REPLACE FUNCTION calcula_valor(valor int) RETURNS setof users AS $$
BEGIN
  if valor = 1 then
     UPDATE users SET salario = salario*1.10;
  else
    if valor =2 then
     UPDATE users SET salario = salario*0.90;
    end if;
  end if;
  return query select * from users where salario>1000;
END;
$$
LANGUAGE 'plpgsql';

select * from calcula_valor(3);
```