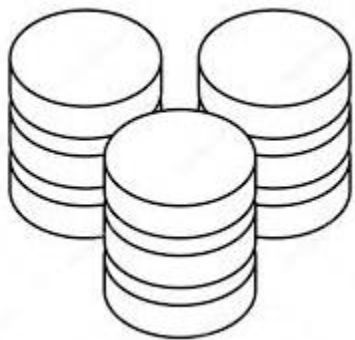


## Banco de Dados II

PL/pgSQL

Funções/Procedures Armazenadas





Profa. Damires Souza  
damires@ifpb.edu.br



# Linguagens no PostgreSQL

```
select * from pg_language;
```

oid [PK] oid 	lanname name 
12	internal
13	c
14	sql
13744	plpgsql

# Bloco anônimo

```
select * from artista;  
alter table artista add indicaoscar integer;  
update artista  
set indicaoscar = 0;
```

```
Do $$  
BEGIN  
    UPDATE artista  
        SET indicaoscar = 10  
        WHERE codart = 100;  
    IF NOT FOUND THEN  
        INSERT INTO artista (codart, nomeart, indicaoscar)  
        VALUES (100, 'XXX', 10);  
    END IF;  
END$$;
```

**Seria interessante  
salvar essa rotina  
no BD??**

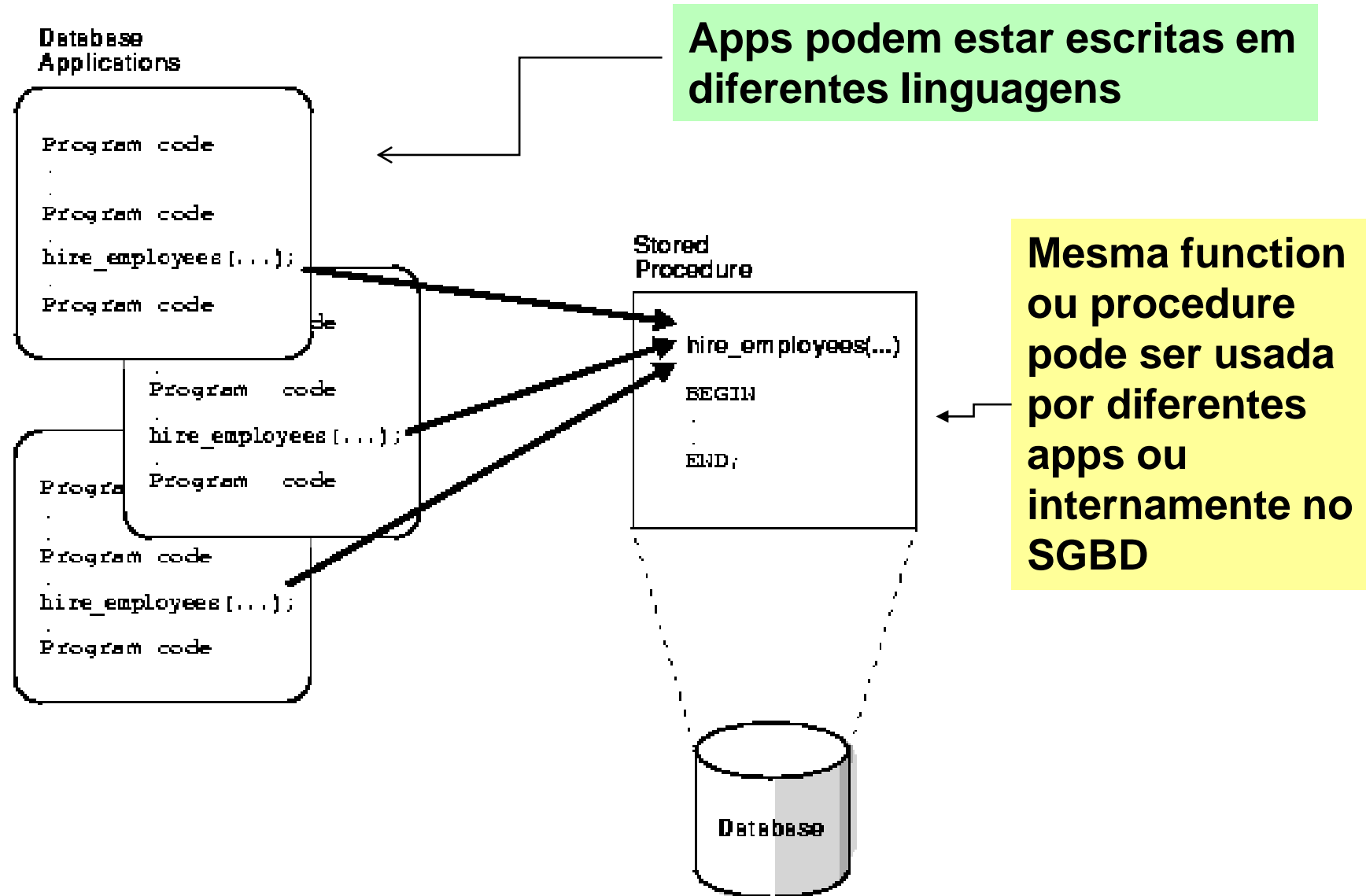
**FOUND: True ou False  
Verificado após um select/update**

# Funções pré-definidas no SGBD

## 1.1 Principais funções para números

Função	Descrição	Exemplo
ABS(n)	Valor absoluto de n	Select ABS(-15) “Absoluto”;
CEIL(n)	Retorna o menor inteiro maior ou igual a n	Select CEIL(15.7);
FLOOR(n)	Retorna o maior inteiro menor ou igual a n	Select FLOOR(15.7);
MOD(a1,a2)	Retorna o resto da divisão do argumento 1 dividido pelo argumento 2	Select MOD(10,4);
POWER(a1,a2)	Calcula o argumento1 elevado ao argumento2	Select power(3,2);
ROUND(n,a)	N arredondado	Select ROUND(15.193,1) “Round”; Select round(42.4382, 2);
TRUNC(n,a)	N truncado	Select trunc(42.8); Select trunc(42.4382, 2);

# Stored Procedure e Stored Function



# Stored Function em SQL

Create or replace function **soma**(integer, integer)  
returns integer as '

    Select \$1+\$2;

' language SQL;

**select soma(1,1);**

Objeto do  
Banco de Dados

Isso pode ser ampliado???



PLPGSQL

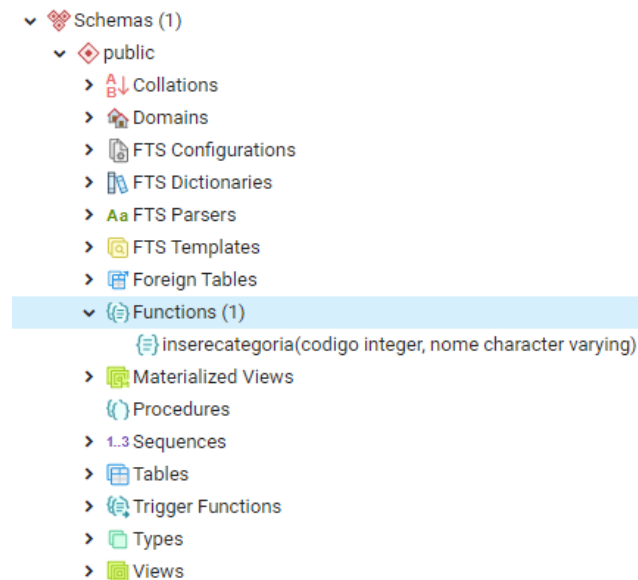
# Stored Procedure/Function

- Uma stored procedure/function é um grupo de comandos **SQL e PL/pgSQL** que executam determinada tarefa e podem ser invocados por um **nome**.

=> Um **bloco PL/pgSQL** nomeado

- A grande vantagem sobre um bloco anônimo é que ela pode ser **compilada e armazenada** no banco de dados como um **objeto de esquema**.
- **A function sempre retorna um valor.**

**Objeto do  
Banco de Dados**



# Stored Procedure/Function

---

- Pode ser acionada por uma **aplicação**, por um **trigger**, por outra **function** ou via **ferramenta** do SGBD
- Traz a vantagem de poder receber valores por meio de **parâmetros ou argumentos**

```
int somar(int param1, int param2)  
{  
    return param1 + param2;  
}  
variavelResultado = somar(var1, var2);
```



**Ideia**



# Exemplo

```
CREATE OR REPLACE FUNCTION InsererCategoria
```

```
(codigo INTEGER, nome VARCHAR(25))
```

```
RETURNS void
```

```
AS $$
```

```
BEGIN
```

```
    INSERT INTO categoria VALUES (codigo, nome);
```

```
END;
```

```
$$ LANGUAGE 'plpgsql';
```

```
Select * from Categoria;
```

```
Select InsererCategoria (7, 'Sci-Fi');
```

codcateg [PK] integer		desccateg character varying (25)
	1	Aventura
	2	Romance
	3	Comédia
	4	Ação
	5	Suspense
	6	Drama
	7	Sci-Fi

# Criando uma Stored Function

---

```
CREATE [ OR REPLACE ] FUNCTION name ( [ [ argmode ] [  
  argname ] argtype [ { DEFAULT | = } default_expr ] [, ...] ] ) [  
  RETURNS rettype | RETURNS TABLE ( column_name column_type  
  [, ...] ) ] { LANGUAGE lang_name } | AS 'definition'
```

Onde:

- **Argmode**: define o modo do argumento, que pode ser IN, OUT, INOUT. Por padrão, o parâmetro utilizado é o **IN**;
- **ArgName**: nome de um argumento;
- **Argtype**: tipo de dado dos argumentos da função
  - Pode ser simples, composto ou referência ao tipo de uma coluna da tabela;

# Criando uma Stored Function

---

- **default\_expr**: valor padrão caso o parâmetro não tenha sido especificado;
- **Rettype**: tipo de dados de retorno
  - Quando existem parâmetros OUT ou INOUT, a cláusula RETURNS pode ser omitida
  - O modificador SETOF indica que a função irá retornar um conjunto de itens, ao invés de um único item;
- **column\_name**: nome de uma coluna de saída na sintaxe do RETURNS TABLE;
- **column\_type**: tipo de dados de uma coluna de saída na sintaxe RETURNS TABLE;
- **lang\_name**: nome da linguagem na qual a função é implementada, podendo esta ser SQL, C, plpgsql ou outra.

# Mais comandos

---

➤ Remover a function:

```
drop function insereCategoria(integer,varchar);
```

➤ Recompilar:

```
alter function insereCategoria;
```

# Exemplo

```
CREATE OR REPLACE function mostra_filmes (v_categoria IN
categoria.desccateg%type)
returns void
as $$
declare
    vfilme cursor (v_categoria categoria.desccateg%type) for
        select titulo, ano
        from filme f join categoria c on f.codcateg = c.codcateg
        where desccateg = v_categoria;
BEGIN
    FOR v_f IN vfilme(v_categoria) LOOP
        raise notice 'Título = %',v_f.titulo;
        raise notice 'Ano = %',v_f.ano;
    END LOOP;
END;
$$ LANGUAGE 'plpgsql';
```

```
Select mostra_filmes('Ação');
```

```
NOTICE:  Título = Era uma vez em Hollywood
NOTICE:  Ano = 2020
NOTICE:  Título = Encontro Explosivo
NOTICE:  Ano = 2010
```

# Exemplo

```
Create or replace function contafilme (  
  cat IN categoria.desccateg%type)  
  RETURNS integer as $$  
  declare totalfilme integer;  
  Begin  
    Select count(*) into totalfilme  
    from filme f join categoria c on f.codcateg = c.codcateg  
    Where desccateg = cat;  
    Return totalfilme;  
  End;  
$$ LANGUAGE 'plpgsql';
```

```
Select contafilme('Ação');
```

contafilme	🔒
integer	
	2

# Exemplo

**Create or replace function DolarToReal**

(dolar in numeric, cotacao numeric)

Returns numeric

As \$\$

Begin

Return dolar \*cotacao;

End;

\$\$ LANGUAGE 'plpgsql';

---

NOTICE: Valor em Real: 563.00  
DO

Do \$\$ declare v numeric;

begin

v:= DolarToReal(100, 4.71);

raise notice 'Valor em Real: %',v;

End\$\$;

# Stored Procedure - Exemplo

```
CREATE OR REPLACE PROCEDURE
mostraNumFilmes(nome varchar(25))
LANGUAGE plpgsql
AS $$
DECLARE
    contador integer := 0;
BEGIN
    Select count(*) INTO contador
    From artista a join personagem p on a.codart = p.codart
    Where a.nomeart = nome;
    RAISE NOTICE '% Fez % filmes', nome, contador;
END $$;
```

```
Call mostraNumFilmes('Joaquin Phoenix');
Call mostraNumFilmes('Cameron Diaz');
```

Data Output Explain Messages Notifications

NOTICE: Cameron Diaz Fez 2 filmes  
CALL

Query returned successfully in 131 msec.



---

---

# Vamos exercitar.

