



Instrução de função de criação do PostgreSQL

RESUMO: Neste tutorial, você aprenderá como usar a `CREATE FUNCTION` instrução PostgreSQL para desenvolver funções definidas pelo usuário.

1) Introdução à instrução Create Function

A instrução `create function` permite que você defina uma nova função definida pelo usuário.

A imagem seguinte ilustra a sintaxe da instrução `create function`:

```
create [or replace] function function_name(param_list)
    returns return_type
    language plpgsql
as
$$
declare
-- variable declaration
begin
-- logic
end;
$$
```

Nesta sintaxe:

- Primeiro, especifique o nome da função após as palavras-chave `create function`. Se você deseja substituir a função existente, pode usar as `or replace`.
- Em seguida, especifique a lista de parâmetros da função entre parênteses após o nome da função. Uma função pode ter zero ou muitos parâmetros.
- Em seguida, especifique o tipo de dados do valor retornado após a palavra-chave `returns`.
- Depois disso, use o `language plpgsql` para especificar a linguagem procedural da função. Observe que o PostgreSQL suporta muitas linguagens procedurais, não apenas `plpgsql`.
- Por fim, coloque um `bloco` entre os marcadores `dollar-quoted string constant`. `$$`

Exemplos de instruções de criação de função do PostgreSQL

Usaremos a tabela film do banco de dados de exemplo Sakila (<https://github.com/jOOQ/sakila>)

film
* film_id title description release_year language_id rental_duration rental_rate length replacement_cost rating last_update special_features fulltext

A instrução a seguir cria uma **função** que conta os filmes cuja duração está entre os parâmetros **len_from** e: **len_to**

```
create function get_film_count(len_from int, len_to int)
returns int
language plpgsql
as
$$
declare
    film_count integer;
begin
    select count(*)
    into film_count
    from film
    where length between len_from and len_to;

    return film_count;
end;
$$;
```

A função `get_film_count` tem duas seções principais: cabeçalho e corpo.

Na seção de cabeçalho:

- Primeiro, o nome da função é `get_film_count` quem vem logo após a chamada de `create function`
- Em segundo lugar, a `get_film_count()` função aceita dois parâmetros `len_from` e `len_to` com o tipo de dados **inteiro**.
- Em terceiro lugar, a função `get_film_count` retorna um inteiro especificado pela cláusula `returns int`.
- Finalmente, a linguagem da função é `plpgsql` indicado pelo `language plpgsql`.

No corpo da função:

- Use a sintaxe constante de cadeia de caracteres entre aspas que começa com **\$\$** e termina com **\$\$**. Entre eles **\$\$**, você pode colocar um bloco que contém a declaração e a lógica da função.
- Na seção de declaração, **declare** uma variável chamada **film_count** que armazena o número de filmes selecionados na tabela film.
- No corpo do bloco, use a **select into** instrução para selecionar o número de filmes cujo comprimento está entre **len_from** e **len_to** e atribua o resultado à variável **film_count**. No final do bloco, use a **return** instrução para retornar o arquivo **film_count**.

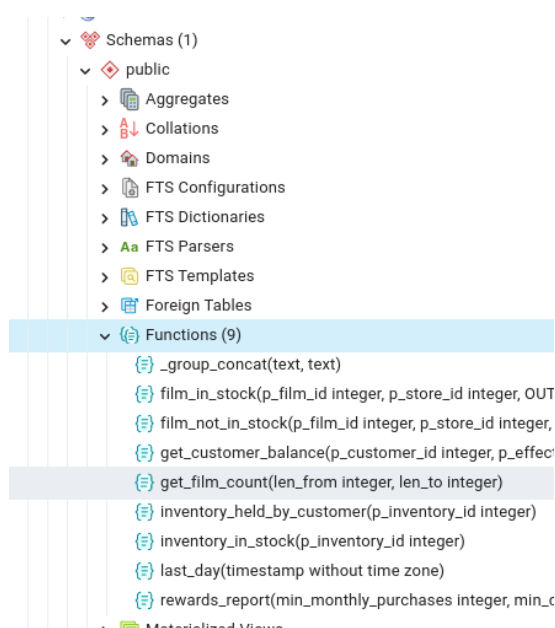
The screenshot shows a PostgreSQL IDE interface with the following components:

- Top Bar:** Dashboard, Properties, SQL, Statistics, Dependencies, Dependents, Processes. The active connection is **sakila/postgres@localhost***.
- Toolbar:** Includes icons for file operations, filters, and execution. A dropdown menu is set to "No limit".
- Query Editor:** Contains the following SQL code:


```

1 create function get_film_count(len_from int, len_to int)
2 returns int
3 language plpgsql
4 as
5 $$
6 declare
7     film_count integer;
8 begin
9     select count(*)
10    into film_count
11   from film
12  where length between len_from and len_to;
13
14     return film_count;
15 end;
16 $$;
      
```
- Output Panel:** Shows the message "CREATE FUNCTION" and "Query returned successfully in 41 msec."

Você depois de fazer o refresh pode verificar que a função foi criado em :

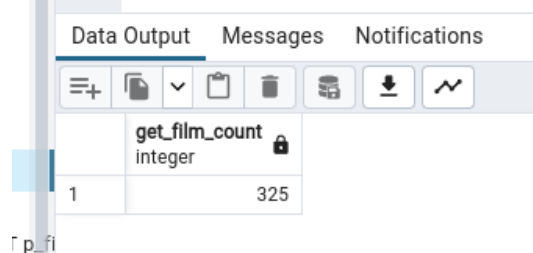


2) Chamando uma função definida pelo usuário

O PostgreSQL fornece três maneiras de chamar uma função definida pelo usuário:

- **Usando a notação posicional**

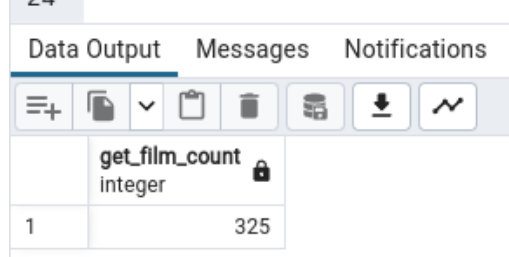
```
17
18 select get_film_count(40,90);
19
```



	get_film_count
1	325

- **Usando notação nomeada**

```
20 select get_film_count(
21     len_from => 40,
22     len_to => 90
23 );
24
```



	get_film_count
1	325

- **Usando a notação mista.**

É a junção das duas formas sendo proibido começar pela nomeada.

3) Introdução aos modos de parâmetro PL/pgSQL

Os modos de parâmetro determinam os comportamentos dos parâmetros. PL/pgSQL suporta três modos de parâmetro: in, out e inout. Um parâmetro assume o inmodo por padrão se você não o especificar explicitamente.

IN	OUT	INOUT
O padrão	Especificado explicitamente	Especificado explicitamente
Passe um valor para a função	Retornar um valor de uma função	Passe um valor para uma função e retorne um valor atualizado.
inparâmetros agem como constantes	outparâmetros agem como variáveis não inicializadas	inoutparâmetros agem como variáveis inicializadas
Não pode ser atribuído um valor	Deve atribuir um valor	Deve ser atribuído um valor

O modo IN

A função a seguir encontra um filme por seu id e retorna o título do filme:

```
create or replace function find_film_by_id(p_film_id int)
returns varchar
language plpgsql
as $$
declare
    film_title film.title%type;
begin
    -- find film title by id
    select title
    into film_title
    from film
    where film_id = p_film_id;

    if not found then
        raise 'Film with id % not found', p_film_id;
    end if;

    return title;

end;$$
```

Como não especificamos o modo para `p_film_id` o parâmetro, ele assume o `in` modo por padrão.

O modo OUT

Os `out` parâmetros são definidos como parte da lista de argumentos e retornados como parte do resultado.

Os `out` parâmetros são muito úteis em funções que precisam retornar vários valores.

Observe que o PostgreSQL oferece suporte aos `out` parâmetros desde a versão 8.1.

Para definir `out` parâmetros, preceda explicitamente o nome do parâmetro com a `out` palavra-chave da seguinte forma:

```
out parameter_name type
```

O exemplo a seguir define a `get_film_stat` função que possui três `out` parâmetros:

```
create or replace function get_film_stat(  
    out min_len int,  
    out max_len int,  
    out avg_len numeric)  
language plpgsql  
as $$  
begin  
  
    select min(length),  
           max(length),  
           avg(length)::numeric(5,1)  
    into min_len, max_len, avg_len  
    from film;  
  
end;$$
```

O modo INOUT

O `inout` modo é a combinação `in` e `out` os modos.

Isso significa que o chamador pode passar um argumento para uma função. A função altera o argumento e retorna o valor atualizado.

A `swap` função a seguir aceita dois inteiros e seus valores:

```
create or replace function swap(  
    inout x int,  
    inout y int  
)  
language plpgsql  
as $$  
begin  
    select x,y into y,x;  
end; $$;
```

A seguinte declaração chama a `swap()` função:

```
select * from swap(10,20);
```

	x integer	y integer
1	20	10

EXERCÍCIO:

Implemente o tutorial que está descrito neste link:

<https://www.postgresqltutorial.com/postgresql-plpgsql/plpgsql-function-returns-a-table/>