

Banco de Dados 1

Roteiro Básico sobre VIEWS, PROCEDURES e TRIGGERS.

Neste roteiro, iremos praticar conceitos básicos de VIEWS, PROCEDURES e TRIGGERS, usando o banco de dados COMPANY, já usado em outros roteiros.

Instruções para submissão

Siga as instruções abaixo para produzir seu material:

* Crie um arquivo de texto chamado **roteiroXX-matricula.sql**
* Adicione os comandos utilizados ao arquivo, na **mesma ordem** que os executar.
* Use comentários no seu código para sinalizar onde cada questão do roteiro está começando e para anotar qualquer outra coisa que desejar. As linhas que denotam comentários iniciam com ‘--’.

Materiais Úteis

Os seguintes links poderão ser úteis durante a elaboração do roteiro:

**VIEWS:**

<https://www.postgresql.org/docs/9.5/static/sql-createview.html>

**FUNCTIONS / PROCEDURES** (são conceitos equivalentes no PostgreSQL):

<https://www.postgresql.org/docs/9.5/sql-createfunction.html>

**TRIGGERS:**

<https://www.postgresql.org/docs/9.5/sql-createtrigger.html>

Antes de começar

Execute os arquivos de criação das tabelas e povoamento, **company.sql** e **company-data.sql** respectivamente.

Questões Propostas

1. Crie as seguintes VIEWS:

1. **vw\_dptmgr**: contém apenas o número do departamento e o nome do gerente;
2. **vw\_empl\_houston:** contém o ssn e o primeiro nome dos empregados com endereço em Houston;
3. **vw\_deptstats**: contém o número do departamento, o nome do departamento e o número de funcionários que trabalham no departamento;
4. **vw\_projstats:** contém o id do projeto e a quantidade de funcionários que trabalham no projeto;

2. Faça consultas nas VIEWS para ver se ficaram corretas;

3. Remova todas as VIEWS criadas;

4. Crie uma FUNCTION chamada *check\_age*, que recebe como parâmetro o ssn de um empregado e retorna:

* 'SENIOR', se o funcionário tem 50 anos ou mais;
* 'YOUNG', se o funcionário tiver menos de 50 anos;
* 'UNKNOWN', se a data de nascimento for NULL;
* 'INVALID', se a data de nascimento for uma data futura;

Para facilitar, abaixo segue um template para criação da sua função:

CREATE OR REPLACE FUNCTION xxxxx(param\_name PARAM\_TYPE)

RETURNS RETURN\_TYPE AS

$$

DECLARE

var1\_name VAR1\_TYPE;

var2\_name VAR2\_TYPE;

var3\_name VAR3\_TYPE;

BEGIN

IF (CONDITION) THEN

         ...

ELSIF (CONDITION) THEN

         ...

END IF;

END;

$$  LANGUAGE plpgsql;

Em seguida, execute os comandos abaixo e confira os resultados:

your\_db=> SELECT check\_age('666666609');

 check\_age

-----------

 SENIOR

(1 row)

your\_db=> SELECT check\_age('333333301');

 check\_age

-----------

 YOUNG

(1 row)

your\_db=> SELECT check\_age('987987987');

 check\_age

-----------

 INVALID

(1 row)

your\_db=> SELECT check\_age('x');

 check\_age

-----------

 UNKNOWN

(1 row)

your\_db=> SELECT check\_age(null);

 check\_age

-----------

 UNKNOWN

(1 row)

Observe que sua função pode ser usada em suas consultas:

your\_db=> SELECT ssn FROM employee WHERE check\_age(ssn) = 'SENIOR';

ssn

-----------

 111111100

 444444400

 ...

 666666608

 666666609

(24 rows)

5. É muito comum combinar FUNCTIONs e TRIGGERs para se definir CONSTRAINTs mais sofisticadas. O PostgreSQL inclui o conceito de TRIGGER PROCEDURES, que permite fazer esta combinação. Veja a documentação [aqui](https://www.postgresql.org/docs/9.5/plpgsql-trigger.html) e responda o que se pede.

1. Crie uma TRIGGER PROCEDURE chamada **check\_mgr** que não permite que um gerente inadequado seja alocado a um departamento. Um gerente é considerado inadequado se:

* não for um funcionário atualmente alocado no departamento; ou
* não possuir subordinados; ou
* não for 'SENIOR';

1. Insira um novo employee com ssn '999999999' que atenda aos requisitos para ser gerente.
2. Agora execute os comandos abaixo e verifique se obteve os mesmos resultados (não precisa adicionar os comandos ao material a ser enviado - apenas informe se deu tudo certo).

INSERT INTO department VALUES ('Test', 2, '999999999', now());

INSERT 0 1

your\_db=> UPDATE department SET mgrssn = null WHERE dnumber=2;

ERROR:  mgrssn cannot be null

your\_db=> UPDATE department SET mgrssn = '888665555' WHERE dnumber=2;

ERROR:  manager must be a SENIOR employee

your\_db=> UPDATE department SET mgrssn = '111111102' WHERE dnumber=2;

ERROR:  manager must have supevisees

your\_db=> UPDATE department SET mgrssn = '111111100' WHERE dnumber=2;

ERROR:  manager must be a department's employe