

Universidade Federal Fluminense

Anna Beatriz Chaboudet Chazan

Avaliação Continuada #3

Niterói– RJ

2025

Anna Beatriz Chaboudet Chazan

Avaliação Continuada #3

Atividade avaliativa, solicitado na aula de Projeto de Banco de Dados para Sistemas de Informação, como requisito para obtenção de nota.

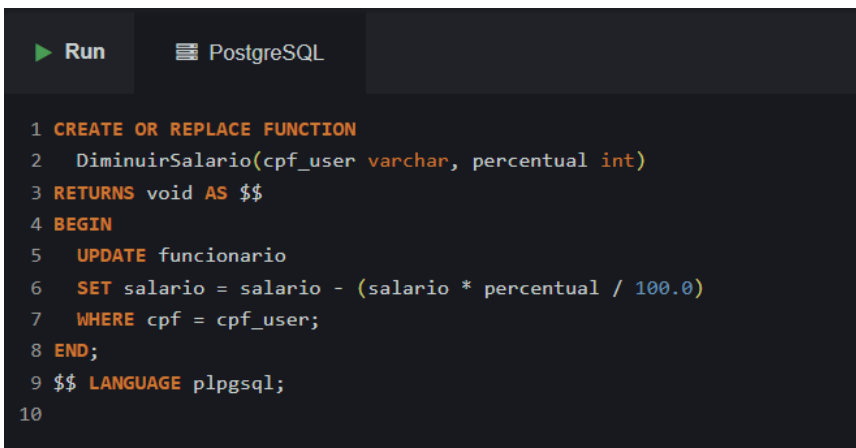
NITERÓI-RJ

2025

Como todos sabem, o estado do Rio de Janeiro bem passando uma por uma severa crise financeira. Dessa forma, o governo estadual cogita diminuir horas de trabalho e salários de muitos funcionários. (fonte: <http://oglobo.globo.com/rio/rj-estuda-reduzir-jornada-salarios-de-servidores-do-estado-20358106>)

Você foi contratado para atualizar o banco de dados dos funcionários estaduais de acordo com as novas diretrizes. Considere a tabela **FUNCIONARIO** apresentada como base para os itens a seguir. As respostas podem ser dadas para os SGBDs MySQL, SQL Server, Oracle e PostgreSQL. Cada discente deve enviar um zip ou rar com os *scripts* de criação dos objetos.

1. Implemente uma função para diminuir o salário de um funcionário em um determinado percentual. A sua função deve se chamar DiminuirSalario e deve receber como parâmetros de entrada o CPF do funcionário e um valor inteiro que representa o percentual de redução.



```
1 CREATE OR REPLACE FUNCTION
2   DiminuirSalario(cpf_user varchar, percentual int)
3 RETURNS void AS $$
4 BEGIN
5   UPDATE funcionario
6   SET salario = salario - (salario * percentual / 100.0)
7   WHERE cpf = cpf_user;
8 END;
9 $$ LANGUAGE plpgsql;
10
```

2. Além da redução de salários, o governo prevê demissões para funcionários que faltem sem justificativa apresentada. Esse tipo de controle não existe hoje no banco de dados. Sua tarefa é desenvolver um mecanismo que controle as faltas de cada um dos funcionários. A partir da 5ª (quinta) falta sem justificativa o campo ATIVO da tabela funcionário deve ser setado para 'N' significando que ele foi demitido.

Sugestão: criem uma tabela que controle as faltas e justificativas e uma trigger associada a essa tabela para verificar a quantidade de faltas.

Tabela de faltas:

```
Run PostgreSQL
1 CREATE TABLE faltas (
2     ID SERIAL PRIMARY KEY,
3     justificativa BOOLEAN DEFAULT FALSE
4     cpf_funcionario VARCHAR(11) REFERENCES funcionario(cpf),
5     data_falta DATE NOT NULL,
6 );
7
```

Trigger:

```
Run PostgreSQL
1 CREATE FUNCTION VerificarFaltas() RETURNS TRIGGER AS $$
2 DECLARE
3     total_faltas_injustificadas INT;
4 BEGIN
5     SELECT COUNT(*) INTO total_faltas_injustificadas
6     FROM faltas
7     WHERE cpf_funcionario = NEW.cpf_funcionario AND justificativa = FALSE;
8
9     IF total_faltas_injustificadas >= 5 THEN
10         UPDATE funcionario
11         SET ativo = 'N'
12         WHERE cpf = NEW.cpf_funcionario;
13     END IF;
14
15     RETURN NEW;
16 END;
17 $$ LANGUAGE plpgsql;
18
19 CREATE TRIGGER VerificarFaltasTrigger
20 AFTER INSERT ON faltas
21 FOR EACH ROW EXECUTE PROCEDURE VerificarFaltas();
22
```

3. O governo do estado também deseja controlar todas as promoções dos funcionários ao longo do anos. Assim como no caso das faltas, esse mecanismo não se encontra implementado no banco de dados. É sua responsabilidade implementar esse controle. Cada funcionário possui um cargo (que por simplificação pode variar entre CARGO1, CARGO2 e CARGO3) e seu nível pode variar entre 1 e 7. Ou seja, o funcionário pode ter o CARGO1 e Nível 5 no momento, e, na próxima promoção ele terá o CARGO1 (que não muda) e Nível 6, e assim por diante. Lembrando que cada funcionário só pode aumentar seu nível de 3 em 3 anos e não pode haver interseção de períodos entre dois níveis. Além disso, um funcionário só pode

ser promovido para o nível imediatamente superior ao atual, logo uma promoção do Nível 1 para o Nível 3 é proibida. Desenvolva uma função que implemente a promoção de um determinado funcionário. Sua função deve receber o CPF do funcionário e o nível para promoção como parâmetros de entrada.

Tabela de promoções:

```
Run PostgreSQL
1 CREATE TABLE promocoas (
2   ID SERIAL PRIMARY KEY,
3   nivel INT NOT NULL CHECK (nivel BETWEEN 1 AND 7),
4   cpf_funcionario VARCHAR(11) REFERENCES funcionario(cpf),
5   cargo VARCHAR(10) NOT NULL,
6   data_promocao DATE NOT NULL
7 );
8
```

Função para promover funcionário:

```
Run PostgreSQL
1 CREATE OR REPLACE FUNCTION
2   PromoverFuncionario(cpf_input VARCHAR, novo_nivel INT)
3 RETURNS VOID AS $$
4 DECLARE
5   nivel_atual INT := 0;
6   data_ultima_promocao DATE := NULL;
7   cargo_funcionario VARCHAR(10);
8   anos_desde_ultima INT;
9 BEGIN
10  SELECT cargo INTO cargo_funcionario
11  FROM funcionario
12  WHERE cpf = cpf_input;
13
14  IF NOT FOUND THEN
15    RAISE NOTICE 'Nenhum funcionário encontrado';
16    RETURN;
17  END IF;
18
19  SELECT nivel, data_promocao INTO nivel_atual, data_ultima_promocao
20  FROM promocoas
21  WHERE cpf_funcionario = cpf_input
22  ORDER BY data_promocao DESC
23  LIMIT 1;
24
25  IF NOT FOUND THEN
```

```
Run PostgreSQL
24
25 IF NOT FOUND THEN
26     RAISE NOTICE 'Ainda não teve promoção';
27     nivel_atual := 0;
28     data_ultima_promocao := CURRENT_DATE - INTERVAL '3 years';
29 END IF;
30
31 IF novo_nivel <> nivel_atual + 1 THEN
32     RAISE NOTICE 'Não é permitido pular níveis';
33     RETURN;
34 END IF;
35 anos_desde_ultima := EXTRACT(YEAR FROM age(CURRENT_DATE, data_ultima_promocao));
36
37 IF anos_desde_ultima < 3 THEN
38     RAISE NOTICE 'Não é possível realizar promoções em um intervalo menor que 3 anos';
39     RETURN;
40 END IF;
41
42 INSERT INTO promocoes (cpf_funcionario, cargo, nivel, data_promocao)
43 VALUES (cpf_input, cargo_funcionario, novo_nivel, CURRENT_DATE);
44
45 RAISE NOTICE 'Promoção realizada com sucesso';
46 END;
47 $$ LANGUAGE plpgsql;
48
```