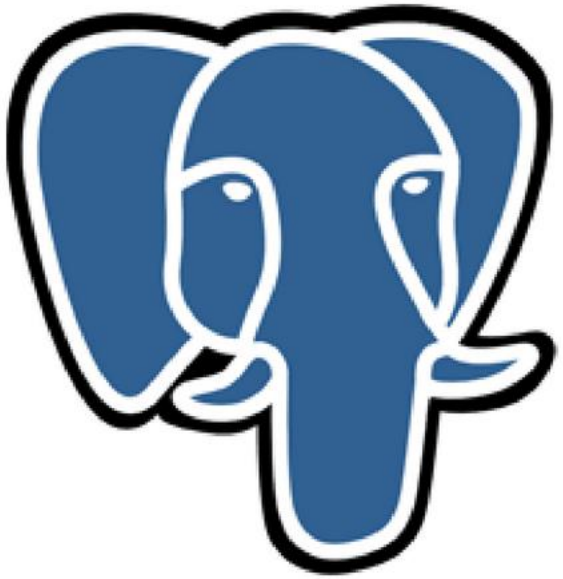


PostgreSQL

Inserindo valores padrão utilizando o PGAdmin

Inserindo valores padrão utilizando o PGAdmin

Sintaxe:

Nome_campo tipo_campo default valor_padrão

Inserindo valores padrão utilizando o PGAdmin

Exemplo:

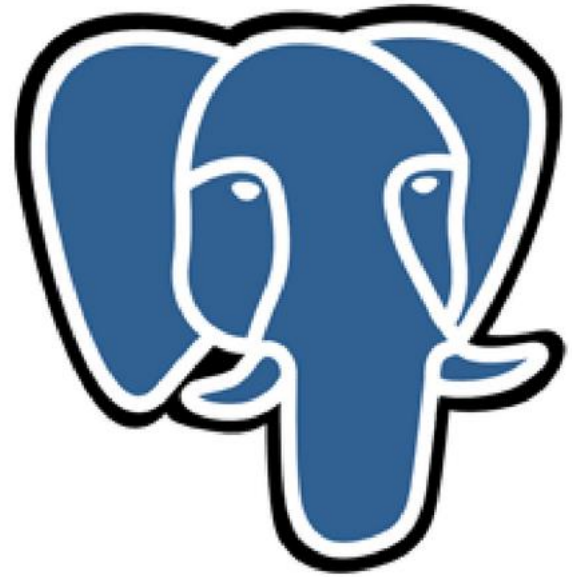
```
DATA_VENDA DATE NOT NULL DEFAULT CURRENT_DATE
```

Funções SQL

utilizando o PGAdmin

SINTAXE

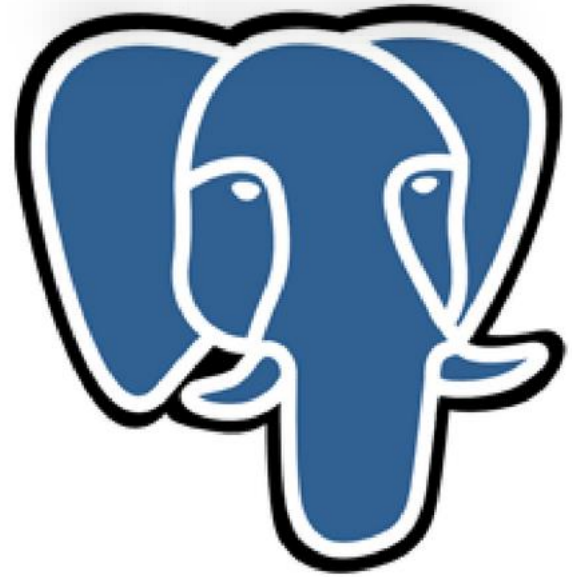
```
CREATE FUNCTION nomeFunção()
RETURNS retorno
LANGUAGE PLPGSQL AS
$$
BEGIN
    RETURN saída
END;
$$;
```



Funções SQL

```
SELECT campo,
nomeDaFunção(campo)
FROM tabela
```

Chamada da função



Funções SQL

Stored Procedures

utilizando o PGAdmin

SINTAXE

```
CREATE PROCEDURE
nomeProcedure(campo tipoDeDados)
LANGUAGE SQL AS
$$
    INSERT INTO tabela(campo)
    VALUES(param_procedure);
$$;
```

STORED PROCEDURE

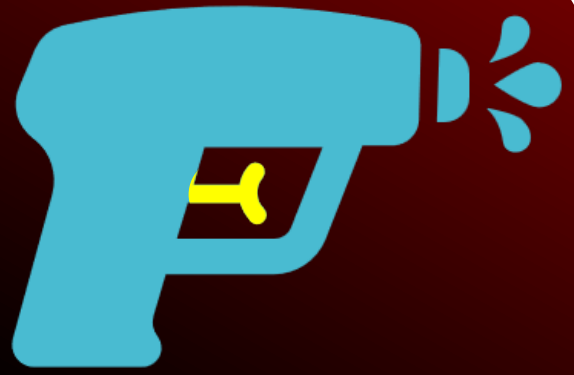
Exercícios procedures

1. Crie uma stored procedure que receba como parâmetro o ID do produto e o percentual de aumento, e reajuste o preço somente deste produto de acordo com o valor passado como parâmetro

2. Crie uma stored procedure que receba como parâmetro o ID do produto e exclua da base de dados somente o produto com o ID correspondente

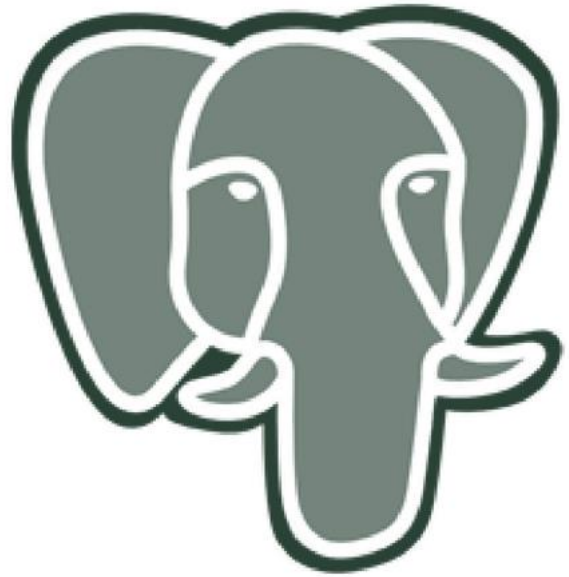
Exercícios

STORED PROCEDURE



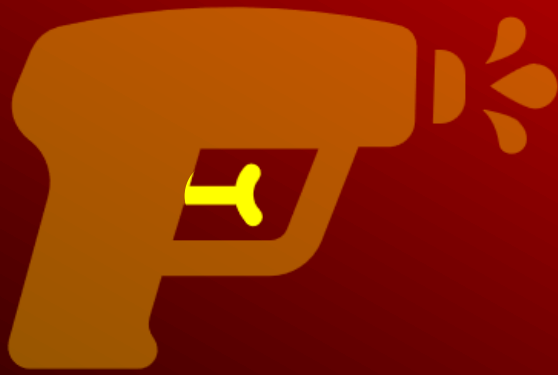
TRIGGERS

utilizando o PGAdmin



Triggers, ou gatilhos,
são rotinas armazenadas em um banco de dados, que são executadas automaticamente quando um evento específico ocorre em uma tabela.

Esses eventos podem ser:



- ✓ Alteração de dados,
- ✓ Inclusão de dados,
- ✓ Exclusão de dados.

Os triggers são opcionais e podem ser definidos usando a instrução CREATE TRIGGER. Para removê-los, deve-se usar DROP TRIGGER.

Os triggers são muito utilizados para:

Manter a consistência dos dados

Propagar alterações em um determinado dado de uma tabela para outras

Controlar quem alterou a tabela

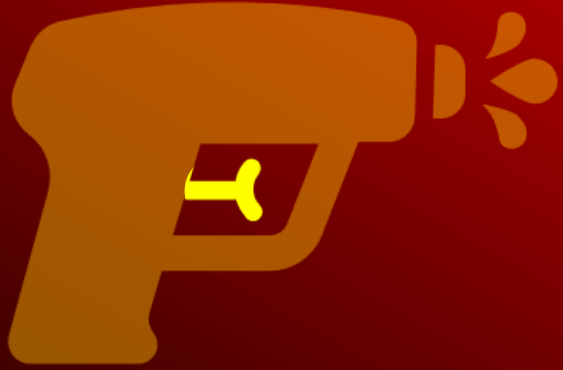
Impor regras de integridade de dados

Causar atualizações em outras tabelas

Gerar ou transformar automaticamente valores para linhas inseridas ou atualizadas

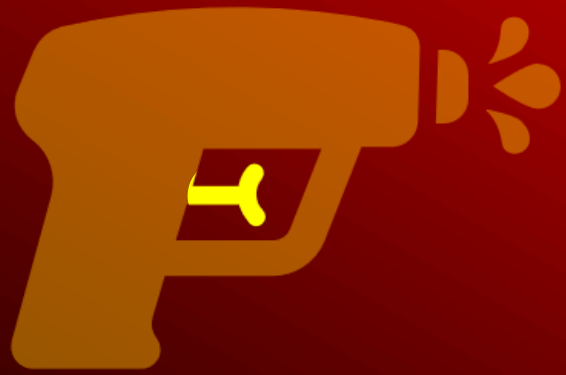
Chamar funções para executar tarefas como emissão de alertas

Os triggers são similares a stored procedures, mas são executadas implicitamente quando ocorre algum evento no banco de dados.

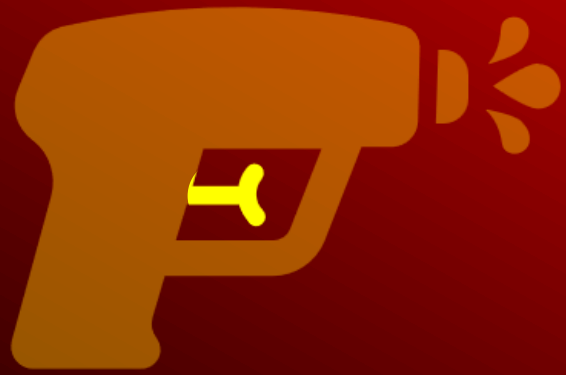


Os triggers são muito utilizados para:

- Manter a consistência dos dados;
- Propagar alterações em algum dado de uma tabela para outras;
- Controlar quem alterou a tabela;
- Impor regras de integridade de dados;

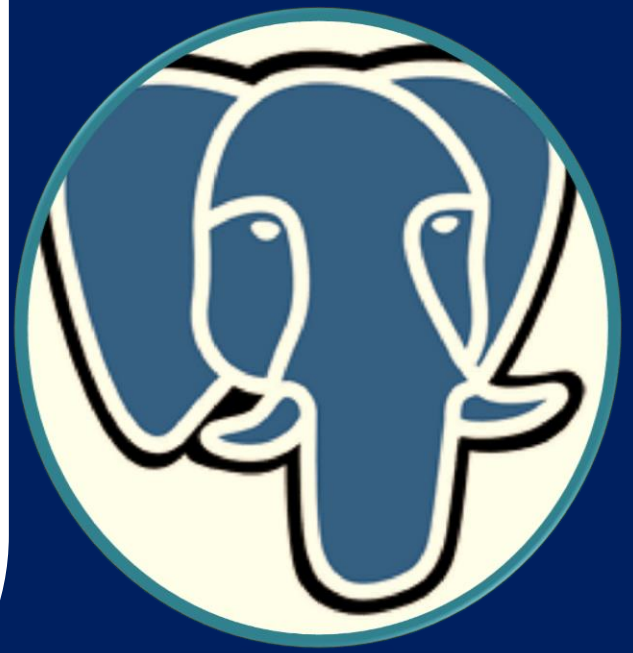


- Causar atualizações em outras tabelas;
- Gerar ou transformar automaticamente valores para linhas inseridas ou atualizadas;
- Chamar funções para executar tarefas como emissão de alertas;
- Os triggers são similares a stored procedures, mas são executadas implicitamente quando ocorre algum evento no banco de dados.



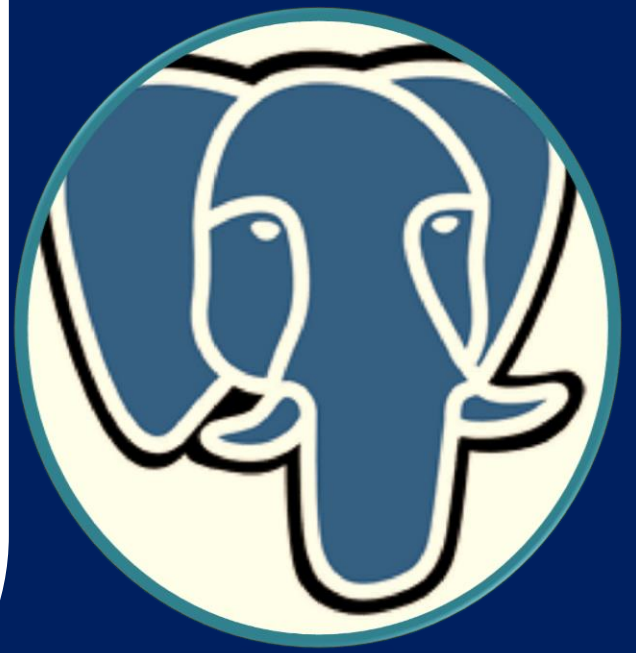
Verificaremos, de forma abstrata, uma trigger function.

As triggers functions podem ser definidas em linguagens compatíveis ao PostgreSQL, como PL/pgSQL, PL/Python, PL/Java dentre outros.



```sql
CREATE OR REPLACE FUNCTION trigger_function_name
RETURNS trigger AS $ExemploFuncao$
BEGIN
/* Aqui definimos nossos códigos.*/
RETURN NEW;
END;
$ExemploFuncao
```

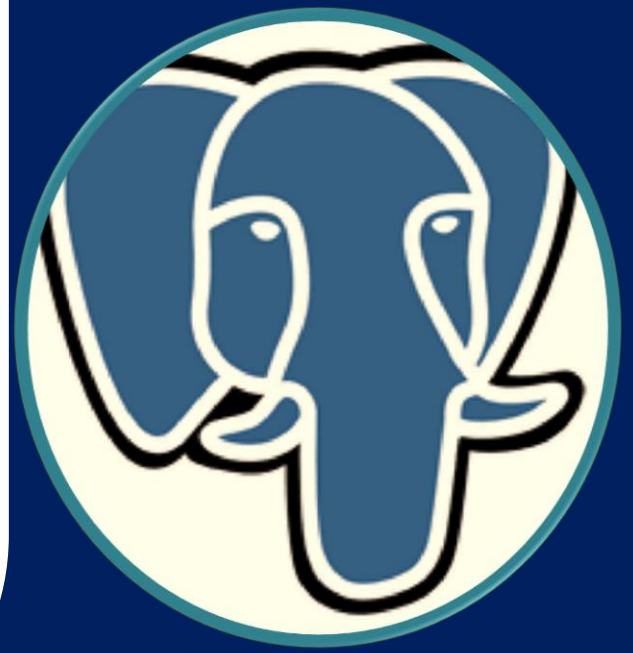
Repare que uma trigger function é, na realidade, uma função no PostgreSQL, mas com a diferença de que ela não recebe argumentos, e sim uma estrutura de dados especial chamada de TriggerData. Repare também que o seu tipo de retorno é a trigger, onde ela é chamada automaticamente no momento da ocorrência dos eventos (que podem ser INSERT, UPDATE, DELETE ou TRUNCATE). Com o PostgreSQL temos dois tipos de trigger disponíveis: trigger de nível de linha (row-level Trigger) e a trigger a nível de instrução (statement level trigger). Ambos são especificados com a utilização das cláusulas FOR EACH ROW (nível gatilho de linha) e FOR EACH STATEMENT, respectivamente. A utilização delas pode ser definida de acordo com a quantidade de vezes que a trigger deverá ser executada. Por exemplo, se uma instrução UPDATE for executada, e esta afetar seis linhas, temos que a trigger de nível de linha será executada seis vezes, enquanto que a trigger a nível de instrução será chamada apenas uma vez por instrução SQL.



Quando utilizamos triggers podemos conectá-las tanto a tabelas quanto a Views, de forma que as triggers são executadas para as tabelas em duas situações:

BEFORE e AFTER,

para qualquer uma das instruções DML (INSERT, UPDATE, DELETE)

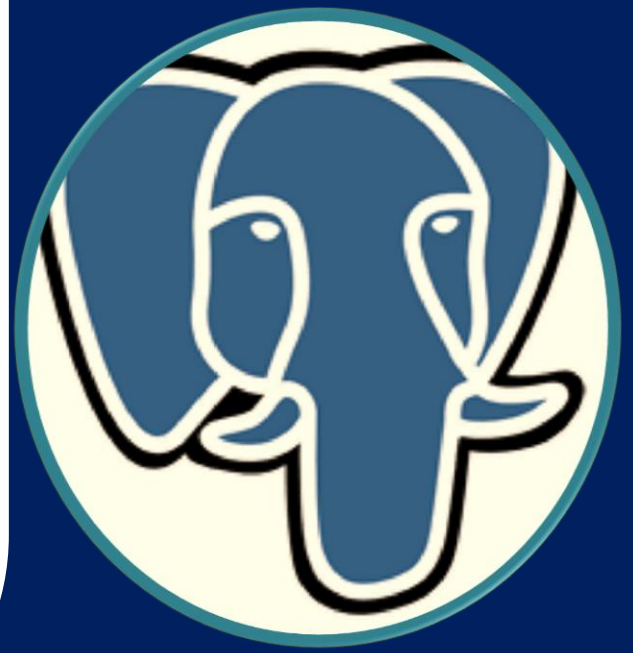


CREATE TRIGGER

PostgreSQL

Sintaxe Básica

```
CREATE TRIGGER <nome>
    BEFORE|AFTER
        INSERT|DELETE|UPDATE
ON <tabela>
     FOR EACH ROW
EXECUTE
    PROCEDURE|FUNCTION
        <nomeProcFunc>;
```

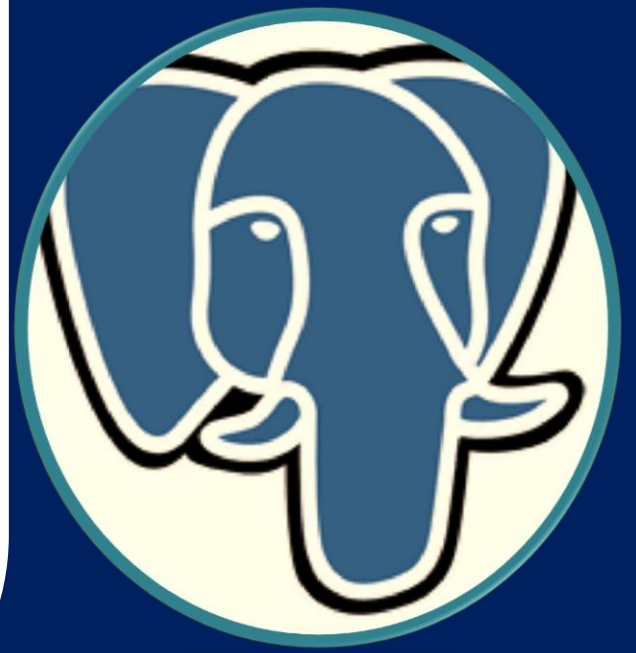


CREATE TRIGGER

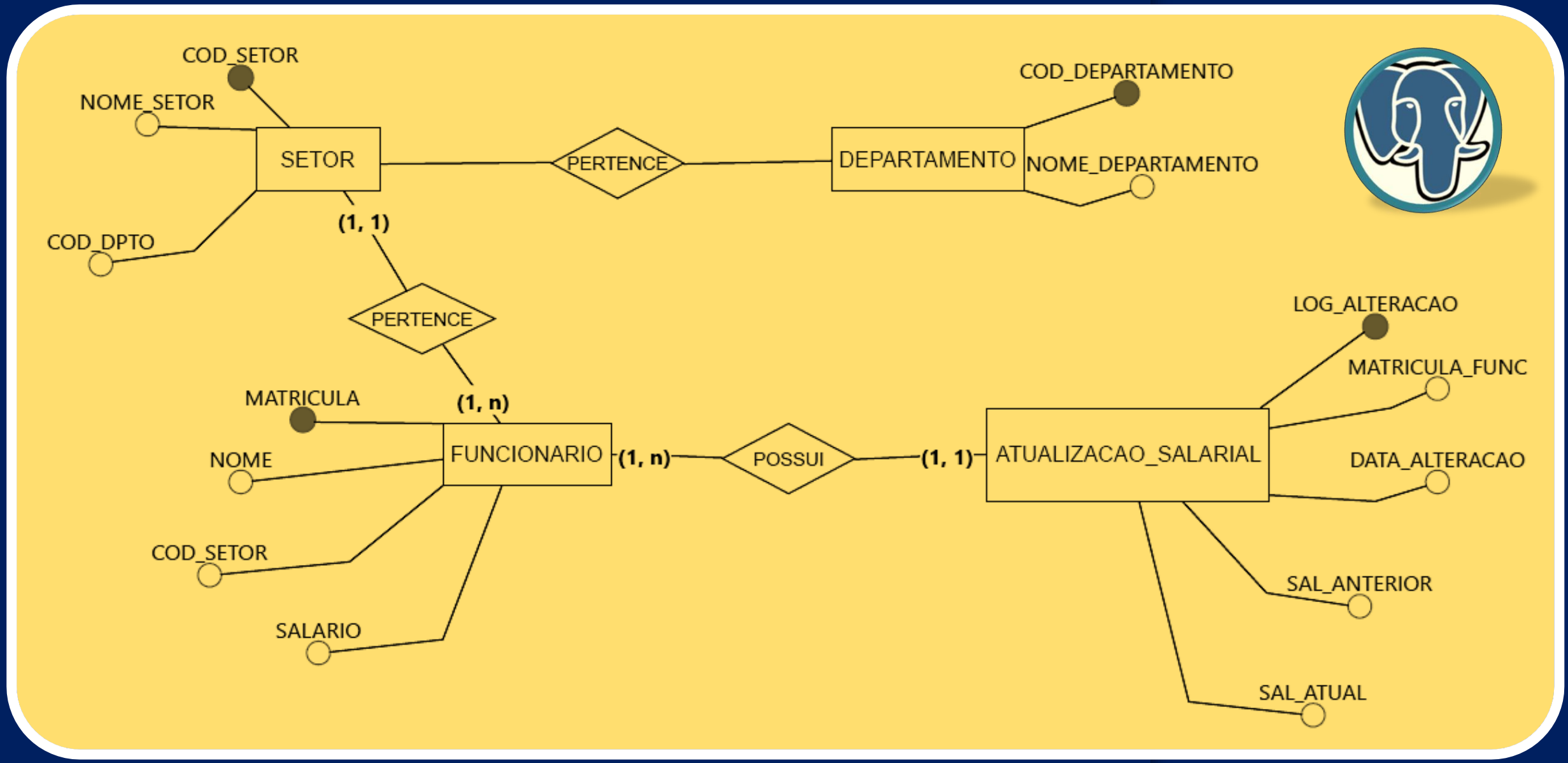
PostgreSQL

SINTAXE Função TIPO TRIGGER

```
CREATE OR REPLACE FUNCTION <nome> RETURNS
trigger AS $$
BEGIN
    //comandos
RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```



Implementar, no PgAdmin o seguinte BD, conforme o modelo abaixo:



Popular as tabelas, conforme o exemplo abaixo:

cod_departamento [PK] integer	nome_departamento character varying (30)
1	ADMINISTRATIVO
2	TECNOLOGIA
3	FINANCEIRO
4	DIRETORIA
5	OPERACONAL

cod_setor [PK] integer	nome_setor character varying (30)	cod_dpto integer
1	FROTA	5
2	SECRETARIA	1
3	RECEPÇÃO	1
4	CONTROLE	1
5	SUPORTE TECNICO	2
6	BANCO DE DADOS	2
7	DESENVOLVIMENTO	2
8	CONTAS A PAGAR	3
9	CONTAS A RECEBER	3
10	ADMINISTRATIVA	4
11	OPERACIONAL	4
12	LIMPEZA	5
13	ENGENHARIA	2

matricula [PK] integer	nome character varying (100)	cod_setor integer	salario double precision
1500	JOAQUIM	1	2653.96
1501	MARTA	1	1960
1502	TANIA	2	3452.67
1503	HELENA	3	2740
1504	GILMAR	4	3243
1505	FELIPE	5	3750
1506	KARLA	6	6200
1507	ANTONIO	7	5500
1509	MARIA	8	3500
1510	LISIANE	9	3500
1511	PAULO	10	7500
1512	VIVIANE	11	7500
1513	ALBERTO	12	4500
1514	FRANCISCO	13	8500

EXEMPLO: Caso haja alguma modificação nos dados da tabela de salários, uma Trigger será disparada.

PostgreSQL

Criando a função que vai atualizar a tabela

```
CREATE OR REPLACE FUNCTION
atualiza_salario() RETURNS trigger AS $$
BEGIN
 INSERT INTO atualizacao_salarial
    (matricula, data_alteracao, salario)
 VALUES
    (NEW.matricula, NOW(), NEW.salario);
RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

sal_antigo,

OLD.salario,

EXEMPLO: Caso haja alguma modificação nos dados da tabela de salários, uma Trigger será disparada.

PostgreSQL

Criando a trigger em funcionarios

```
CREATE TRIGGER tg_atualiza_salario
   AFTER INSERT OR UPDATE
   ON funcionarios
FOR EACH ROW
   EXECUTE PROCEDURE atualiza_salario();
```