

**Banco de Dados**

**Aluno: Jonas Sales Rebouças**

**Matrícula: 20191135000296**

**SEMESTRE. 2020.2**

**BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO**

**TRIGGER**

**1 - Crie um TRIGGER para baixar o estoque de um PRODUTO quando ele for vendido;**

Criando a Função

CREATE FUNCTION q01() RETURNS TRIGGER AS

$$

BEGIN

UPDATE EX2\_PRODUTO SET quantidade = quantidade - NEW.quantidade

WHERE codproduto = NEW.codproduto;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- Criando o Trigger

CREATE TRIGGER ex2\_baixar\_estoque\_trigger AFTER INSERT ON

EX2\_ITEMPEDIDO FOR EACH ROW

EXECUTE PROCEDURE q01();

--Verificando o Estoque

SELECT \* FROM EX2\_PRODUTO;

-- Criando uma Inserção de Pedido e seus Itens

INSERT INTO EX2\_PEDIDO VALUES (7, 1, TO\_DATE('01/04/2012',

'DD/MM/YYYY'), '00001', 400.00);

INSERT INTO EX2\_ITEMPEDIDO VALUES (7, 1, 10.90, 1, 1);

INSERT INTO EX2\_ITEMPEDIDO VALUES (7, 2, 389.10, 1, 3);

--Verificando o Estoque

SELECT \* FROM EX2\_PRODUTO;

**2 - Crie um TRIGGER para criar um log dos CLIENTES modificados;**

-- Criando a Função

CREATE FUNCTION q02() RETURNS TRIGGER AS

$$

BEGIN

INSERT INTO EX2\_LOG VALUES (nextval('EX2\_LOG\_SEQ'), 'now',

'INSERINDO NOVO CLIENTE - ' || NEW.CPF);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- CRIANDO TRIGGER

CREATE TRIGGER ex2\_log\_clientes AFTER INSERT ON EX2\_CLIENTE FOR EACH

ROW

EXECUTE PROCEDURE q02();

--CRIANDO UM NOVO CLIENTE

INSERT INTO EX2\_CLIENTE VALUES (8, 'José Adolfo',

TO\_DATE('05/11/1998', 'DD/MM/YYYY'), '55555555591');

--VERIFICANDO TABELA DE LOG

SELECT \* FROM EX2\_LOG;

**3 - Crie um TRIGGER para criar um log dos PRODUTOS atualizados;**

-- Criando a Função

CREATE FUNCTION q03() RETURNS TRIGGER AS

$$

BEGIN

INSERT INTO EX2\_LOG VALUES (nextval('EX2\_LOG\_SEQ'), 'now',

'INSERINDO NOVO PRODUTO - CODIGO - ' || NEW.codproduto);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- CRIANDO TRIGGER

CREATE TRIGGER ex2\_log\_produtos AFTER INSERT ON EX2\_PRODUTO FOR EACH

ROW

EXECUTE PROCEDURE q03();

--CRIANDO UM NOVO PRODUTO

SELECT \* FROM EX2\_PRODUTO;

INSERT INTO EX2\_PRODUTO VALUES (6, 'PC GAMER', 5);

--VERIFICANDO TABELA DE LOG

SELECT \* FROM EX2\_PRODUTO;

SELECT \* FROM EX2\_LOG;

**4 - Crie um TRIGGER para criar um log quando não existir a quantidade do ITEMPEDIDO em estoque;**

-- Criando a Função

CREATE FUNCTION q04() RETURNS TRIGGER AS

$$

BEGIN

IF NEW.quantidade > (SELECT EX2\_PRODUTO.quantidade from

EX2\_PRODUTO where NEW.codproduto=EX2\_PRODUTO.codproduto) THEN

INSERT INTO EX2\_LOG VALUES (nextval('EX2\_LOG\_SEQ'), 'now',

'PRODUTO COM ESTOQUE LIMITADO - CODIGO - ' || NEW.codproduto);

DELETE FROM EX2\_ITEMPEDIDO WHERE quantidade=NEW.quantidade

and codproduto=NEW.codproduto;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- CRIANDO TRIGGER

CREATE TRIGGER ex2\_log\_estoque AFTER INSERT ON EX2\_ITEMPEDIDO FOR EACH

ROW

EXECUTE PROCEDURE q04();

--CRIANDO UM NOVO CLIENTE

SELECT \* FROM EX2\_PRODUTO;

INSERT INTO EX2\_ITEMPEDIDO VALUES (6, 2, 10.90, 20, 1);

INSERT INTO EX2\_ITEMPEDIDO VALUES (6, 3, 10.90, 30, 1);

INSERT INTO EX2\_ITEMPEDIDO VALUES (6, 4, 10.90, 5, 1);

--VERIFICANDO TABELA DE LOG

SELECT \* FROM EX2\_ITEMPEDIDO;

SELECT \* FROM EX2\_LOG;

**5 - Crie um TRIGGER para criar uma requisição de REQUISICAO\_COMPRA quando o estoque atingir 50% da venda mensal;**

--criando função

CREATE FUNCTION Q05() RETURNS TRIGGER AS

$$ BEGIN

IF NEW.quantidade > OLD.quantidade\*1.5 THEN

INSERT INTO EX2\_REQUISICAO\_COMPRA VALUES (nextval('EX2\_LOG\_SEQ'),

old.codproduto, now(), new.quantidade);

END if;

RETURN NEW;

end;

$$ LANGUAGE PLPGSQL;

--criando trigger

CREATE TRIGGER REQUISIÇÃO\_DE\_REQUISIÇÃO\_DE\_COMPRA AFTER UPDATE ON

EX2\_PRODUTO FOR EACH ROW EXECUTE PROCEDURE Q05();

--fazendo inserts

select \* from EX2\_PRODUTO;

UPDATE EX2\_PRODUTO SET quantidade=20 WHERE codproduto=1;

select \* from EX2\_REQUISICAO\_COMPRA

**6 - Crie um TRIGGER para criar um log quando um ITEMPEDIDO for removido;**

-- CRIANDO A FUNÇÃO

CREATE FUNCTION q06() RETURNS TRIGGER AS

$$

BEGIN

INSERT INTO EX2\_LOG VALUES (nextval('EX2\_LOG\_SEQ'), 'now','PRODUTO REMOVIDO - ITEM PEDIDO - CODIGO - ' || OLD.codproduto);

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

-- CRIANDO TRIGGER

CREATE TRIGGER ex2\_log\_estoque after delete ON EX2\_ITEMPEDIDO FOR EACH ROW EXECUTE PROCEDURE q06();

-- EXECUTANDO

SELECT \* FROM EX2\_ITEMPEDIDO;

SELECT \* FROM EX2\_LOG;

DELETE from EX2\_ITEMPEDIDO WHERE codpedido= 4;

SELECT \* FROM EX2\_ITEMPEDIDO;

SELECT \* FROM EX2\_LOG;

**7 - Crie um TRIGGER para criar um LOG quando o valor total do pedido for maior que R$1000.**

CREATE OR REPLACE TRIGGER ex2\_log\_pedidos\_caros

AFTER INSERT ON EX2\_PEDIDO

FOR EACH ROW

WHEN (NEW.valortotal > 1000)

BEGIN

INSERT INTO EX2\_LOG

VALUES (EX2\_LOG\_SEQ.nextval, SYSDATE, 'PEDIDO CARO - '||:NEW.codpedido);

--mensagem na tela

dbms\_output.put\_line('Pedido Caro registrado - '||:NEW.codpedido);

END;

**8 - Crie um TRIGGER para NÃO deixar valores negativos serem INSERIDOS em ITEMPEDIDO, o valor mínimo é "0";**

CREATE FUNCTION q08() RETURNS TRIGGER AS

$$

BEGIN

IF NEW.valorunitario < 0 THEN

RAISE EXCEPTION 'O VALOR NÃO PODE SER NEGATIVO';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

**9 - Crie um TRIGGER que NÃO permita que uma PESSOA com data de nascimento anterior a data de hoje seja inserida ou atualizada.**

CREATE FUNCTION q09() RETURNS TRIGGER AS $$

BEGIN

IF (EX2\_CLIENTE.DATANASCIMENTO > 'now') THEN

RAISE EXCEPTION 'Data inválida, maior do que data atual';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER EX2\_CLIENTE\_DATA BEFORE INSERT OR UDPATE ON EX2\_CLIENTE

FOR EACH ROW EXECUTE PROCEDURE q09();

**10 - Crie um TRIGGER para automaticamente numerar o ITEMPEDIDO.numeroitem de acordo com valores anteriores do mesmo PEDIDO.**

CREATE FUNCTION q10() RETURNS TRIGGER AS

$$

DECLARE

contador int;

BEGIN

SELECT COUNT(\*) INTO contador

FROM EX2\_ITEMPEDIDO

WHERE EX2\_ITEMPEDIDO.CODPEDIDO = NEW.CODPEDIDO;

UPDATE EX2\_ITEMPEDIDO SET numeroitem= contador WHERE CODPEDIDO =

NEW.CODPEDIDO and numeroitem=NEW.numeroitem;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER EX2\_numerar AFTER INSERT ON EX2\_ITEMPEDIDO

FOR EACH ROW EXECUTE PROCEDURE q10();

SELECT \* FROM EX2\_ITEMPEDIDO;

INSERT INTO EX2\_ITEMPEDIDO VALUES (4, 6, 30, 1, 5);

SELECT \* FROM EX2\_ITEMPEDIDO;

INSERT INTO EX2\_ITEMPEDIDO VALUES (4, 20, 50, 1, 5);

SELECT \* FROM EX2\_ITEMPEDIDO;

**11 - Crie um TRIGGER para não permitir quantidade negativa no ITEMPEDIDO.**

CREATE FUNCTION q11() RETURN TRIGGER AS

$$

BEGIN

IF NEW.quabtidade < 0 THEN

RAISE EXCEPTION 'A QUANTIDADDE NÃO PODE SER NEGATIVA';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER ex2\_quant\_negativos BEFORE INSERT ON EX2\_ITEMPEDIDO FOR EACH ROW

EXECUTE PROCEDURE q11();

INSERT INFO EX2\_ITEMPEDIDO VALUES (6,5,1,-20,1);

select \*from EX2\_ITEMPEDIDO;

**12 - Crie um TRIGGER para acrescentar a palavra "Sr(a)" ao nome das PESSOAS que tem nasceram há mais de 30 anos.**

CREATE FUNCTION q12() RETURNS TRIGGER AS

$$

BEGIN

IF (NEW.DATANASCIMENTO < '01/01/1990') THEN

UPDATE EX2\_CLIENTE SET nome='Sr(a) ' || NEW.nome WHERE

NEW.codcliente=codcliente;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER EX2\_idade AFTER INSERT ON EX2\_CLIENTE FOR EACH ROW

EXECUTE PROCEDURE q12();

select\*from EX2\_CLIENTE;

INSERT INTO EX2\_CLIENTE VALUES (19, 'José Adolfo',

TO\_DATE('05/11/1980', 'DD/MM/YYYY'), '55555555531');

select\*from EX2\_CLIENTE;

**14 - Crie um TRIGGER para remover as REQUISICOESCOMPRA de um produto que é removido.**

CREATE FUNCTION q14() RETURNS TRIGGER AS

$$

BEGIN

DELETE from EX2\_REQUISICAO\_COMPRA where

EX2\_REQUISICAO\_COMPRA.codproduto = EX2\_PRODUTO.codproduto;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER EX2\_cod AFTER DELETE ON EX2\_PRODUTO FOR EACH ROW

EXECUTE PROCEDURE q14();

SELECT \* FROM EX2\_PRODUTO;

INSERT INTO EX2\_REQUISICAO\_COMPRA VALUES (1, 1, 'now', 5);

INSERT INTO EX2\_REQUISICAO\_COMPRA VALUES (2, 2, 'now', 5);

SELECT \* FROM EX2\_REQUISICAO\_COMPRA;

DELETE FROM EX2\_REQUISICAO\_COMPRA WHERE codproduto=1;

DELETE FROM EX2\_ITEMPEDIDO WHERE codproduto=1;

SELECT \* FROM EX2\_REQUISICAO\_COMPRA;

--15) Crie um TRIGGER que ajuste os pedidos de compra para que não

--existam itens repetidos, ou seja,

--quando o mesmo ITEMPEDIDO for inserido deve-se disparar uma

mensagem.

CREATE FUNCTION q15() RETURNS TRIGGER AS

$$

DECLARE

var\_contador\_item int;

BEGIN

SELECT COUNT(\*) INTO var\_contador\_item

FROM EX2\_ITEMPEDIDO

WHERE EX2\_ITEMPEDIDO.CODPRODUTO = NEW.CODPRODUTO AND

EX2\_ITEMPEDIDO.CODPEDIDO = NEW.CODPEDIDO;

IF (var\_contador\_item >= 1) THEN

RAISE EXCEPTION 'ITEM REPETIDO, AJUSTAR QUANTIDADE DE ITEM

EXISTENTE!';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER EX2\_ITEMPEDIDO\_VALIDAR BEFORE INSERT ON EX2\_ITEMPEDIDO

FOR EACH ROW EXECUTE PROCEDURE q15();

SELECT \* FROM EX2\_ITEMPEDIDO;

INSERT INTO EX2\_ITEMPEDIDO VALUES (2, 3, 10.90, 1, 2);

INSERT INTO EX2\_ITEMPEDIDO VALUES (2, 4, 10.90, 1, 1);

**15 - Crie um TRIGGER que ajuste os pedidos de compra para que não existam itens repetidos, ou seja,**

**quando o mesmo ITEMPEDIDO for inserido deve-se disparar uma mensagem.**

Crie um TRIGGER que ajuste os pedidos de compra para que não existam itens repetidos, ou seja,

--quando o mesmo ITEMPEDIDO for inserido deve-se disparar uma mensagem.

CREATE FUNCTION q15() RETURNS TRIGGER AS

$$

DECLARE

var\_contador\_item int;

BEGIN

SELECT COUNT(\*) INTO var\_contador\_item

FROM EX2\_ITEMPEDIDO

WHERE EX2\_ITEMPEDIDO.CODPRODUTO = NEW.CODPRODUTO AND EX2\_ITEMPEDIDO.CODPEDIDO = NEW.CODPEDIDO;

IF (var\_contador\_item >= 1) THEN

RAISE EXCEPTION 'ITEM REPETIDO, AJUSTAR QUANTIDADE DE ITEM EXISTENTE!';

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER EX2\_ITEMPEDIDO\_VALIDAR BEFORE INSERT ON EX2\_ITEMPEDIDO

FOR EACH ROW EXECUTE PROCEDURE q15();

SELECT \* FROM EX2\_ITEMPEDIDO;

INSERT INTO EX2\_ITEMPEDIDO VALUES (2, 3, 10.90, 1, 2);

INSERT INTO EX2\_ITEMPEDIDO VALUES (2, 4, 10.90, 1, 1);