```sql
/****** Aula 6 ******/
-- Trigger composta
ALTER TABLE itens_pedido ADD total_item NUMBER(10,2) ;
UPDATE itens_pedido SET preco_item = get_preco(cod_prod)*0.97 ;
UPDATE itens_pedido SET total_item = qtde_pedida*preco_item*(100 - descto_itens_pedido)/100 ;
SELECT * FROM itens_pedido ;

-- desabilitando os gatilhos
ALTER TABLE itens_pedido DISABLE ALL TRIGGERS ;
ALTER TRIGGER atualiza_total_item DISABLE ; -- trigger simples desabilitada, valerá a composta
ALTER TRIGGER valida_preco_item ENABLE ;
ALTER TRIGGER atualiza_pedido ENABLE ;

-- gatilho para calcular o total do item automaticamente quando insere ou atualiza um item de pedido
CREATE OR REPLACE TRIGGER atualiza_total_item
AFTER INSERT OR UPDATE ON itens_pedido
FOR EACH ROW
DECLARE
totalnovo itens_pedido.total_item%TYPE;
BEGIN
totalnovo := :NEW.qtde_pedida*:NEW.preco_item*(100 -(:NEW.descto_itens_pedido))/100 ;
-- atualizando o total
UPDATE itens_pedido
SET total_item = totalnovo
WHERE cod_prod = :NEW.cod_prod AND num_ped = :NEW.num_ped ;
END ;

SELECT * FROM itens_pedido WHERE num_ped = 1910 ;
SELECT get_preco ( 5003) FROM dual ;

-- erro tabela mutante
-- Ocorre quando uma trigger no nivel de linha tenta ler ou alterar uma tabela que
-- está sendo modificada (via INSERT, UPDATE, ou DELETE).
-- Especificamente ocoore quando está tentando ler ou alterar a mesma tabela que disparou o gatilho,
-- por exemplo com um SELECT na tabela que dispara o gatilho com UPDATE
-- ou um gatilho que dispara para um UPDATE e dentro do próprio gatilho tenta atualizar a mesma tabela
INSERT INTO itens_pedido (num_ped, cod_prod, qtde_pedida, descto_itens_pedido, situacao_item, preco_item)
VALUES (1910, 5007 , 1, 25, 'SEPARACAO', 1);
SELECT * FROM itens_pedido ;
-- erro tabela mutante
UPDATE itens_pedido SET qtde_pedida = 3, descto_itens_pedido = 25
WHERE num_ped = 1910 AND cod_prod = 5008 ;

-- Utilizando trigger composta
-- gatilho composto que é tratado como uma única transação
DROP TRIGGER atualiza_total_item_compound ;
CREATE OR REPLACE TRIGGER atualiza_total_item_compound
FOR INSERT OR UPDATE OF qtde_pedida, descto_itens_pedido ON itens_pedido
-- limitando as colunas do update na sequência porque senão chama recursivamente
-- por causa do update no total do item
COMPOUND TRIGGER
    linha_alterada    rowid;
    -- gatilho que recupera o ID da linha que foi alterada ou inserida
    AFTER EACH ROW IS
    BEGIN
    linha_alterada := :NEW.rowid ;
    DBMS_OUTPUT.put_line ( linha_alterada ) ;
    END AFTER EACH ROW;
    -- gatilho que dispara a execução da procedure para a linha que acabou de ser inserida ou atualizada
    AFTER STATEMENT IS
    BEGIN
    DBMS_OUTPUT.put_line ( 'Estou aqui..'||linha_alterada) ;
    -- chama a procedure acima que atualiza o saldo da movimentação
       atualiza_total_item_proc ( linha_alterada) ;
    END AFTER STATEMENT;
END ;

--Procedure que atualiza o total do item
DROP PROCEDURE atualiza_total_item_proc ;
CREATE OR REPLACE PROCEDURE atualiza_total_item_proc ( vlinha IN VARCHAR2)
IS
vtotal_atual  itens_pedido.total_item%TYPE;
BEGIN
-- recuperando os dados
DBMS_OUTPUT.PUT_LINE ('Begin...') ;
SELECT qtde_pedida*preco_item*(100 - descto_itens_pedido)/100
INTO vtotal_atual
FROM itens_pedido
WHERE rowid = vlinha ;
DBMS_OUTPUT.PUT_LINE ('Total atual :'|| TO_CHAR(vtotal_atual)) ;
-- atualizando
DBMS_OUTPUT.PUT_LINE ('Procedure :'||vlinha) ;
UPDATE itens_pedido
SET total_item = vtotal_atual
```

```
85  WHERE rowid = vlinha ;
86  END ;
87
88  BEGIN
89  atualiza_total_item_proc ( 'AAAE+tAABAAALHhAA9') ;
90  END ;
91
92  SELECT * FROM itens_pedido WHERE rowid = 'AAAE+tAABAAALHhAA9';
93  UPDATE itens_pedido SET total_item = 10 WHERE rowid = 'AAAE+tAABAAALHhAA9';
94
95  -- Procedure com parâmetro de saída
96  -- Procedure para reajustar o preço dos produtos indicando se deu certo ou não
97  CREATE OR REPLACE PROCEDURE reajusta_preco ( vindice IN NUMBER, vresultado OUT NUMBER)
98  IS
99  BEGIN
100 UPDATE produto
101 SET preco_venda = preco_venda * ( 100 + vindice)/100 ;
102 COMMIT ;
103 vresultado := 1;
104  EXCEPTION
105       WHEN OTHERS THEN vResultado := 0;
106 END ;
107
108 -- Código de teste
109 SET SERVEROUTPUT ON;
110 DECLARE
111 -- Parâmetro de saída deve ser declarado no bloco de chamada
112 resultado NUMBER;
113 BEGIN
114 -- Esse teste deve se completado com sucesso.
115 -- O procedimento atribui valor ao parâmetro de saída (Resultado).
116 reajusta_preco ( 10 , resultado );
117 IF resultado = 1 THEN
118 dbms_output.put_line('Reajuste de preços realizado com sucesso !!!');
119 ELSE
120 dbms_output.put_line('Reajuste não realizado ! Ocorreu algum erro !!!');
121 END IF;
122 END;
123
124 SELECT * FROM produto ;
```