**Cenário da Prática: Gestão de Loja com Estoque**

Vamos supor que temos uma loja de produtos e queremos gerenciar os produtos, pedidos, estoque, e registrar alterações importantes no banco de dados. Para isso, utilizaremos Triggers, Stored Procedures e Funções.

**Estrutura do Banco de Dados e Tabelas**

Primeiro, criamos o banco de dados e as tabelas necessárias:

SQL

CREATE DATABASE IF NOT EXISTS MinhaBaseDeDados;

USE MinhaBaseDeDados;

-- Tabela de Produtos

CREATE TABLE IF NOT EXISTS produtos (

ProdutoID INT AUTO\_INCREMENT PRIMARY KEY,

NomeProduto VARCHAR(100),

Preco DECIMAL(10,2),

Estoque INT

);

**Inserção de Produtos na Tabela produtos**

Estes registros nos darão produtos para trabalhar e testar a prática:

SQL

INSERT INTO produtos (NomeProduto, Preco, Estoque) VALUES

('Notebook', 3000.00, 10),

('Mouse Sem Fio', 50.00, 100),

('Teclado Mecânico', 200.00, 50),

('Monitor 24"', 800.00, 20),

('Cadeira Gamer', 1200.00, 15);

-- Tabela de Pedidos

CREATE TABLE IF NOT EXISTS pedidos (

PedidoID INT AUTO\_INCREMENT PRIMARY KEY,

ProdutoID INT,

Quantidade INT,

DataPedido DATETIME,

FOREIGN KEY (ProdutoID) REFERENCES produtos(ProdutoID)

);

**Inserção Inicial de Pedidos na Tabela pedidos**

Vamos inserir pedidos para fazer uma validação completa:

SQL

INSERT INTO pedidos (ProdutoID, Quantidade, DataPedido) VALUES

(1, 2, '2024-11-04 10:00:00'),

(2, 5, '2024-11-04 11:00:00'),

(3, 3, '2024-11-04 12:00:00');

-- Tabela de Log de Estoque (para rastrear alterações no estoque)

CREATE TABLE IF NOT EXISTS log\_estoque (

LogID INT AUTO\_INCREMENT PRIMARY KEY,

ProdutoID INT,

Alteracao INT,

DataAlteracao DATETIME,

Descricao VARCHAR(255)

);

**1. Criação de um Trigger: Atualizar Log de Estoque**

Este trigger será acionado automaticamente **sempre que o estoque de um produto for atualizado**. Ele vai registrar o evento na tabela de log\_estoque.

SQL

DELIMITER //

CREATE TRIGGER after\_estoque\_update

AFTER UPDATE ON produtos

FOR EACH ROW

BEGIN

-- Insere um registro no log de estoque quando houver alteração no campo de Estoque

IF OLD.Estoque <> NEW.Estoque THEN

INSERT INTO log\_estoque (ProdutoID, Alteracao, DataAlteracao, Descricao)

VALUES (OLD.ProdutoID, NEW.Estoque - OLD.Estoque, NOW(), CONCAT('Estoque atualizado de ', OLD.Estoque, ' para ', NEW.Estoque));

END IF;

END //

DELIMITER ;

**Exemplo**

UPDATE produtos

SET Estoque = 15

WHERE ProdutoID = 1;

**Explicação**:

* **AFTER UPDATE ON produtos**: O trigger será disparado após a atualização da tabela produtos.
* **IF OLD.Estoque <> NEW.Estoque**: Só registra no log se houver de fato uma alteração no estoque.
* **INSERT INTO log\_estoque**: Registra a alteração, incluindo a diferença do estoque (NEW.Estoque - OLD.Estoque) e uma descrição da mudança.

**2. Criação de uma Stored Procedure: Realizar um Pedido**

Esta Stored Procedure será usada para **realizar um pedido**, diminuindo automaticamente o estoque e inserindo o registro na tabela pedidos.

SQL

DELIMITER //

CREATE PROCEDURE RealizarPedido(IN pProdutoID INT, IN pQuantidade INT)

BEGIN

DECLARE vEstoqueAtual INT;

-- Verifica o estoque atual do produto

SELECT Estoque INTO vEstoqueAtual

FROM produtos

WHERE ProdutoID = pProdutoID;

-- Verifica se há estoque suficiente

IF vEstoqueAtual >= pQuantidade THEN

-- Insere o pedido na tabela de pedidos

INSERT INTO pedidos (ProdutoID, Quantidade, DataPedido)

VALUES (pProdutoID, pQuantidade, NOW());

-- Atualiza o estoque do produto

UPDATE produtos

SET Estoque = Estoque - pQuantidade

WHERE ProdutoID = pProdutoID;

ELSE

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Estoque insuficiente para realizar o pedido';

END IF;

END //

DELIMITER ;

**Exemplo Prático de Chamada**

Vamos supor que você deseja realizar um pedido de 5 unidades do produto com ProdutoID = 2 (por exemplo, um "Mouse Sem Fio"). A chamada seria:

SQL

CALL RealizarPedido(2, 5);

**Explicação**:

* **2**: É o ProdutoID do produto que você quer comprar. Neste exemplo, significa que queremos comprar o produto com ProdutoID = 2.
* **5**: É a quantidade do produto que queremos comprar.

*Obs*. Atualiza na tabela pedidos.

**Explicação**:

* **Verifica Estoque**: Antes de realizar o pedido, verifica se o estoque atual é suficiente.
* **Inserção no Pedido**: Caso haja estoque, insere o pedido na tabela pedidos e reduz o estoque do produto correspondente.
* **Erro de Estoque Insuficiente**: Caso não haja estoque suficiente, gera um erro com a mensagem "Estoque insuficiente para realizar o pedido".

**3. Criação de uma Função: Calcular Valor Total dos Pedidos de um Produto**

Esta função será usada para **calcular o valor total dos pedidos realizados para um determinado produto**.

SQL

DELIMITER //

CREATE FUNCTION CalcularValorTotalPedidos(pProdutoID INT) RETURNS DECIMAL(10,2)

BEGIN

DECLARE vValorTotal DECIMAL(10,2);

-- Calcula o valor total dos pedidos para o produto especificado

SELECT SUM(p.Quantidade \* pr.Preco) INTO vValorTotal

FROM pedidos p

JOIN produtos pr ON p.ProdutoID = pr.ProdutoID

WHERE p.ProdutoID = pProdutoID;

-- Se não houver pedidos, define como 0

IF vValorTotal IS NULL THEN

SET vValorTotal = 0.00;

END IF;

RETURN vValorTotal;

END //

DELIMITER ;

**Explicação**:

* **Calcular Valor Total**: A função usa um JOIN para multiplicar a quantidade pedida pelo preço de cada produto e sumarizar o valor total.
* **Tratamento de Nulos**: Se não houver pedidos (vValorTotal IS NULL), define o valor total como 0.00.

**4. Testando a Prática Completa**

Agora vamos testar toda a prática de uma forma integrada:

**4.1. Inserção de Produtos no Estoque**

SQL

-- Inserindo produtos no estoque

INSERT INTO produtos (NomeProduto, Preco, Estoque) VALUES

('Notebook', 3000.00, 10),

('Mouse Sem Fio', 50.00, 100),

('Teclado Mecânico', 200.00, 50);

**4.2. Realizando um Pedido**

Vamos chamar a Stored Procedure RealizarPedido para realizar um pedido:

SQL

CALL RealizarPedido(1, 3); -- Realiza um pedido de 3 unidades do produto com ProdutoID = 1 (Notebook)

Após a execução deste comando:

* **O estoque do produto Notebook** será atualizado de 10 para 7.
* **Um registro** será adicionado na tabela log\_estoque indicando que o estoque foi alterado.
* **Um registro** será adicionado na tabela pedidos.

**4.3. Verificando o Valor Total dos Pedidos**

Podemos utilizar a função CalcularValorTotalPedidos para saber quanto já foi pedido de determinado produto:

SQL

SELECT CalcularValorTotalPedidos(1) AS ValorTotalNotebook;

Este comando retornará o valor total dos pedidos realizados para o produto com ProdutoID = 1.

**Conclusão**

Esta prática demonstra o uso de **Triggers**, **Stored Procedures** e **Funções** em um único cenário prático:

* **Trigger**: Mantém um registro automático de alterações no estoque, útil para auditoria.
* **Stored Procedure**: Automatiza a lógica de realizar um pedido, verificando se há estoque disponível e atualizando as tabelas envolvidas.
* **Função**: Permite calcular o valor total dos pedidos de um produto, facilitando consultas específicas.