**Gatilhos (Triggers)**

É muito comum, em aplicações que utilizam [**bancos de dados**](http://www.devmedia.com.br/cursos/banco-de-dados), que ações sejam disparadas em resposta ou como consequência de outras, realizando operações de cálculo, validações e, em geral, surtindo alterações na base de dados.

Em muitos casos, os programadores optam por executarem tais ações a partir da própria aplicação, executando várias instruções SQL em sequência para obter o resultado esperado. De fato essa é uma solução que pode até ser tida como mais segura, por certos pontos de vista, mas tende a tornar ainda mais “pesada” a execução de certas tarefas, requisitando mais recursos da máquina cliente.

A “solução” (ou pelo menos uma forma alternativa) a essa está na utilização de **TRIGGERS** no banco de dados, automatizando certas ações com base em eventos ocorridos.

**Triggers**(“gatilhos” em português) são objetos do banco de dados que, relacionados a certa tabela, permitem a realização de processamentos em consequência de uma determinada ação como, por exemplo, a inserção de um registro.

Os **triggers**podem ser executados ANTES ou DEPOIS das operações de **INSERT**, **UPDATE**e **DELETE**de registros.

Observação: [**o suporte a triggers foi incluído na versão 5.0.2 do MySQL**](http://www.devmedia.com.br/mysql-triggers/8088).

**Prós e Contras das Triggers**

Os principais pontos positivos sobre os **triggers** são:

* Parte do processamento que seria executado na aplicação passa para o banco, poupando recursos da máquina cliente.
* Facilita a manutenção, sem que seja necessário alterar o código fonte da aplicação.

Já contra sua utilização existem as seguintes considerações:

* Alguém que tenha acesso não autorizado ao banco de dados poderá visualizar e alterar o processamento realizado pelos gatilhos.
* Requer maior conhecimento de manipulação do banco de dados (SQL) para realizar as operações internamente.

A seguir é explicado o processo de criação de **triggers**, a sintaxe utilizada e o significado de cada instrução.

**Sintaxe da Trigger**

A sintaxe dos comandos para criar um novo trigger no [**MySQL**](http://www.devmedia.com.br/guias/mysql/19) é a seguinte:

CREATE TRIGGER nome momento evento

ON tabela

FOR EACH ROW

BEGIN

/\*corpo do código\*/

END

**Listagem 1.** Sintaxe para criação de trigger

Onde se tem os seguintes parâmetros:

* nome: nome do gatilho, segue as mesmas regras de nomeação dos demais objetos do banco.
* momento: quando o gatilho será executado. Os valores válidos são BEFORE (antes) e AFTER (depois).
* evento: evento que vai disparar o gatilho. Os valores possíveis são **INSERT**, **UPDATE** e **DELETE**. Vale salientar que os comandos **LOAD DATA** e **REPLACE** também disparam os eventos de inserção e exclusão de registros, com isso, os gatilhos também são executados.
* tabela: nome da tabela a qual o gatilho está associado.

Não é possível criar mais de um trigger para o mesmo evento e momento de execução na mesma tabela. Por exemplo, não se pode criar dois gatilhos **AFTER INSERT** na mesma tabela.

**Os registros NEW e OLD**

Como os *triggers*, são executados em conjunto com operações de inclusão e exclusão, é necessário poder acessar os registros que estão sendo incluídos ou removidos. Isso pode ser feito através das palavras NEW e OLD.

Em gatilhos executados após a inserção de registros, a palavra reservada NEW dá acesso ao novo registro. Pode-se acessar as colunas da tabela como atributo do registro NEW, como veremos nos exemplos.

O operador OLD funciona de forma semelhante, porém em gatilhos que são executados com a exclusão de dados, o OLD dá acesso ao registro que está sendo removido.

**Utilização do trigger**

Para exemplificar e tornar mais clara a utilização de gatilhos, simularemos a seguinte situação: um mercado que, ao realizar vendas, precisa que o estoque dos produtos seja automaticamente reduzido. A devolução do estoque deve também ser automática no caso de remoção de produtos da venda.

Como se trata de um ambiente hipotético, teremos apenas duas tabelas de estrutura simples, cujo script de criação é mostrado na listagem a seguir.

CREATE TABLE Produtos

(

Referencia VARCHAR(3) PRIMARY KEY,

Descricao VARCHAR(50) UNIQUE,

Estoque INT NOT NULL DEFAULT 0

);

INSERT INTO Produtos VALUES ('001', 'Feijão', 10);

INSERT INTO Produtos VALUES ('002', 'Arroz', 5);

INSERT INTO Produtos VALUES ('003', 'Farinha', 15);

CREATE TABLE ItensVenda

(

Venda INT,

Produto VARCHAR(3),

Quantidade INT

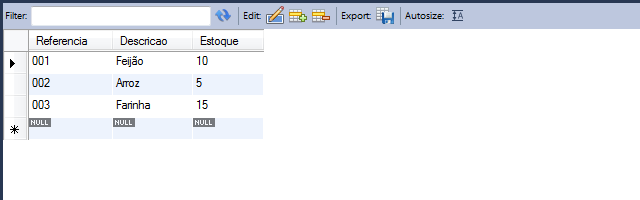
);

**Listagem 2.** Criação das tabelas utilizadas

Ao inserir e remover registro da tabela ItensVenda, o estoque do produto referenciado deve ser alterado na tabela Produtos. Para isso, serão criados dois triggers: um *AFTER INSERT* para dar baixa no estoque e um *AFTER DELETE*para fazer a devolução da quantidade do produto.

*Observação: como usaremos instruções que requerem ponto e vírgula no final, alteraremos o delimitador de instruções para $$ e depois de criar os triggers, voltaremos para o padrão. Essa alteração não está diretamente ligada aos triggers.*

Apenas para registrar e conferir, a imagem a seguir mostra um select feito sobre a tabela Produtos após a inserção dos registros de exemplo.

  
**Figura 1.** Registros iniciais da tabela Produtos

Na **Listagem 3** são criados os gatilhos para executar as ações já discutidas.

DELIMITER $

CREATE TRIGGER Tgr\_ItensVenda\_Insert AFTER INSERT

ON ItensVenda

FOR EACH ROW

BEGIN

UPDATE Produtos SET Estoque = Estoque - NEW.Quantidade

WHERE Referencia = NEW.Produto;

END$

CREATE TRIGGER Tgr\_ItensVenda\_Delete AFTER DELETE

ON ItensVenda

FOR EACH ROW

BEGIN

UPDATE Produtos SET Estoque = Estoque + OLD.Quantidade

WHERE Referencia = OLD.Produto;

END$

DELIMITER ;

**Listagem 3.** Criação dos triggers

No primeiro gatilho, foi utilizado o registro NEW para obter as informações da linha que está sendo inserida na tabela. O mesmo é feito no segundo gatilho, onde se obtém os dados que estão sendo apagados da tabela através do registro OLD.

Tendo criado os **triggers**, podemos testá-los inserindo dados na tabela ItensVenda. Nesse caso, vamos simular uma venda de número 1 que ontem três unidades do produto 001, uma unidade do produto 002 e cinco unidades do produto 003.

INSERT INTO ItensVenda VALUES (1, '001',3);

INSERT INTO ItensVenda VALUES (1, '002',1);

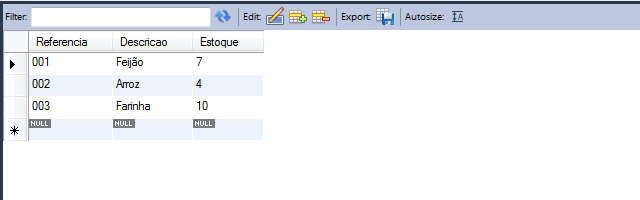
INSERT INTO ItensVenda VALUES (1, '003',5);

**Listagem 4.** Inserindo dados na tabela

DELETE FROM ItensVenda WHERE Venda = 1 AND Produto = '001';

**Listagem 5.** Excluindo dados da tabela ItensVenda

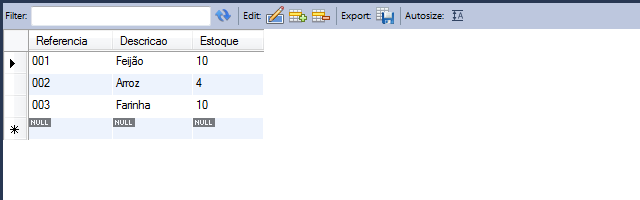
Agora, fazendo uma consulta à tabela Produtos, obtemos o resultado exibido na **Figura 2**.

  
**Figura 2.** Baixa no estoque após a inserção na tabela ItensVenda

Nota-se que o estoque dos produtos foi corretamente reduzido, de acordo com as quantidades “vendidas”.

Agora para testar o **trigger** da exclusão, removeremos o produto 001 dos itens vendidos. Com isso, o seu estoque deve ser alterado para o valor inicial, ou seja, 10.

Executando novamente um select na tabela Produtos, veremos que apenas o produto 001 teve o estoque atualizado, voltando a 10, como vemos na **Figura 3**.

  
**Figura 3.**Devolução do estoque após exclusão de registro na tabela ItensVenda

Com isso confirmamos que os gatilhos estão funcionando da forma esperada.

Abaixo temos um comando que exibe as triggers que foram criadas:

SHOW TRIGGERS

**Listagem 6.** Exibição das triggers criadas

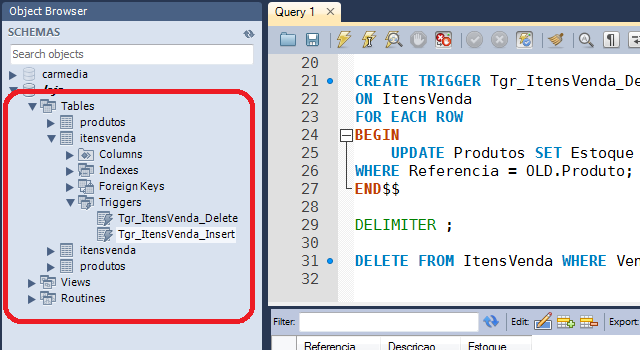
A seguir vemos a exclusão de uma trigger no MySQL:

DROP TRIGGER Tgr\_ItensVenda\_Insert

**Listagem 7.** Exclusão de trigger

**Informações adicionais**

vale o seguinte comentário: na nova versão do MySQL, no **MySQL** Workbench é possível visualizar os gatilhos relacionados a uma tabela através do Object browser, como mostra a figura a seguir.

  
**Figura 4.**Object Browser no MySQL Workbench

Em ambientes reais, [**triggers podem ser utilizados para operações mais complexas**](http://www.devmedia.com.br/desenvolvendo-triggers-em-sql-server-oracle-firebird-e-postgres/5625), por exemplo, antes de vender um item, verificar se há estoque disponível e só então permitir a saída do produto.