Implementando controle de estoque no MySQL com triggers e procedures

Veja neste artigo uma solução prática e de fácil implementação para realizar o controle de estoque (muito utilizado em aplicações para o comércio) diretamente no banco de dados MySQL, utilizando triggers e stored procedures.

Neste artigo será demonstrado como podemos controlar estoques internamente no MySQL usando triggers (gatilhos) e stored procedures.

Uma funcionalidade básica e obrigatória dos sistemas desenvolvidos para o comércio e empresas em geral é o controle do estoque dos produtos, desse modo o vendedor consegue consultar no sistema e verificar a disponibilidade de um determinado produto em tempo real. Com essas informações setores como compras e PCP conseguem planejar melhor as atividades de compra e produção.

Existem várias técnicas para se controlar os estoques, cada programador desenvolve um controle de estoque que atenda as necessidades do seu cliente. Aqui será demonstrada uma forma de implementar esse controle, tomando como exemplo os estoques de uma papelaria.

Vamos construir um pequeno banco de dados "PAPELARIA" usando o MySQL 5.5.24, nesse banco vamos criar quatro tabelas e alguns triggers e um procedure:

- PRODUTO
- ENTRADA_PRODUTO
- ESTOQUE
- SAIDA_PRODUTO

Abaixo segue uma breve descrição e Script para cada tabela:

TABELA "PRODUTO"

A tabela de "PRODUTO" vai conter o cadastro dos produtos que a papelaria vende, nesse exemplo foram criados somente os campos básicos para esse tipo de cadastro, segue abaixo o Script para criação dessa tabela.

Listagem 1: Script de criação da tabela Produtos

```
1 CREATE TABLE `produto` (
2 `id` INT(11) NOT NULL AUTO_INCREMENT,
3 `status` CHAR(1) NOT NULL DEFAULT 'A',
4 `descricao` VARCHAR(50) NULL DEFAULT NULL,
```

```
`estoque_minimo` INT(11) NULL DEFAULT NULL,
`estoque_maximo` INT(11) NULL DEFAULT NULL,
PRIMARY KEY (`id`))
```

Para adiantar um pouco foram cadastrados alguns produtos. Vale uma observação para o campo "STATUS", que indica se o cadastro está ativo "A" ou inativo "I", somente para fins didáticos.

id	status	descricao	estoque_minimo	estoque_maximo
1	A	CANETA	10	100
2	A	LAPIS	10	100
3	A	BORRACHA	5	50
4	A	LAPISEIRA	5	40
5	A	CORRETIVO	5	20

Figura 1: Lista de produtos já cadastrados na tabela PRODUTO

TABELA "ENTRADA_PRODUTO"

Nessa tabela serão gravadas todas as compras de produtos efetuadas para papelaria e através de triggers vamos controlar as inserções na tabela de "ESTOQUE", segue abaixo o Script para criação dessa tabela.

Listagem 2: Script de criação da tabela ENTRADA_PRODUTO

```
CREATE TABLE `entrada_produto` (
`id` INT(11) NOT NULL AUTO_INCREMENT,
`id_produto` INT(11) NULL DEFAULT NULL,
`qtde` INT(11) NULL DEFAULT NULL,
`valor_unitario` DECIMAL(9,2) NULL DEFAULT '0.00',
`data_entrada` DATE NULL DEFAULT NULL,
PRIMARY KEY (`id`))
```

Para esse artigo não estaremos usando Foreign Keys (Chaves Estrangeiras), notem que o campo "ID_PRODUTO" não está configurado como FK. Imaginem que todas as compras serão lançadas nessa tabela.

TABELA "ESTOQUE"

Essa tabela somente recebe os dados conforme as ações executadas nas tabelas de "ENTRADA_PRODUTO" e "SAIDA_PRODUTO". O usuário não tem interação direta como INSERÇÕES, UPDATES E EXCLUSÕES, a tabela "ESTOQUE" é somente o resultado das ações de compra e venda de produtos. Segue abaixo o script para criação dessa tabela.

Listagem 3: Script de criação da tabela ESTOQUE

```
CREATE TABLE `estoque` (
`id` INT(11) NOT NULL AUTO_INCREMENT,
`id_produto` INT(11) NULL DEFAULT NULL,
`qtde` INT(11) NULL DEFAULT NULL,
`valor_unitario` DECIMAL(9,2) NULL DEFAULT '0.00',
PRIMARY KEY (`id`))
```

TABELA "SAIDA_PRODUTO"

Nessa tabela serão gravadas todas as saídas (Vendas) de produtos e através de triggers essas ações serão refletidas na tabela de "ESTOQUE". Segue abaixo script para criação dessa tabela.

Listagem 4: Script de criação da tabela SAIDA_PRODUTO

```
CREATE TABLE `saida_produto` (
`id` INT(11) NOT NULL AUTO_INCREMENT,
```

```
`id_produto` INT(11) NULL DEFAULT NULL,
`qtde` INT(11) NULL DEFAULT NULL,
`data_saida` DATE NULL DEFAULT NULL,
`valor_unitario` DECIMAL(9,2) NULL DEFAULT '0.00',
PRIMARY KEY (`id`))
```

Agora vamos criar apenas um procedure que vai atualizar os estoques na tabela de "ESTOQUE". Notem que nas quatro tabelas criadas existem dois campos em comum "ID_PRODUTO" e "QTDE", são estes campos que serviram como parâmetros para inserção e baixa de estoque nos procedures.

Abaixo segue uma breve descrição e script para cada procedure.

PROCEDURE "SP_AtualizaEstoque"

Esse procedure recebe três parâmetros (id_prod, qtde_comprada, valor_unit) e tem a finalidade de inserir ou debitar produtos na tabela de "ESTOQUE" de acordo com o os parâmetros que são passados.

Listagem 5: Script de criação do procedure SP_AtualizaEstoque

```
DELIMITER //
  CREATE PROCEDURE `SP_AtualizaEstoque`( `id_prod` int, `qtde_comprada` int, valor_uni
BEGIN
    declare contador int(11);

    SELECT count(*) into contador FROM estoque WHERE id_produto = id_prod;

    IF contador > 0 THEN
        UPDATE estoque SET qtde=qtde + qtde_comprada, valor_unitario= valor_unit
        WHERE id_produto = id_prod;
    ELSE
        INSERT INTO estoque (id_produto, qtde, valor_unitario) values (id_prod, qtde_c
    END IF;
END //
DELIMITER ;
```

Observem que foi declarada uma variável contador para receber o valor da instrução SELECT count(*). Caso exista um produto cadastrado no estoque com o mesmo id_prod passado como parâmetro, então será inserido na variável contador o número de linhas que atendem a essa condição. Posteriormente verifica-se o valor de contador, se for maior que 0 então executa-se um UPDATE na tabela "ESTOQUE", senão é feito um "INSERT". Essa verificação pode ser feita de diversas maneiras, o leitor fique à vontade para implementar da melhor maneira possível.

Vamos agora criar os triggers que serão ativadas sempre que ocorrerem eventos de INSERT, UPDATE E DELETE. Será criado uma trigger para cada evento das tabelas "ENTRADA_PRODUTO" e "SAIDA_PRODUTO" ao todo serão 6 triggers. Infelizmente o MySQL ainda não suporta múltiplos eventos em um mesmo trigger, então teremos um pouquinho de trabalho nessa fase.

- TRG_EntradaProduto_AI;
- TRG_EntradaProduto_AU;
- TRG_EntradaProduto_AD;
- TRG_SaidaProduto_AI;
- TRG_SaidaProduto_AU;
- TRG_SaidaProduto_AD.

Observação: o padrão usado para nomenclatura dos triggers varia conforme o programador, aqui iniciaremos com "TRG", abreviação de Trigger + nome da tabela + identificação do evento em que será disparado a trigger:

- AI : After Insert (Após Inserção);
- AU: After Update (Após Atualização);
- AD: After Delete (Após Exclusão).

Basicamente cada trigger vai conter apenas uma linha de instrução, que será a chamada de um procedure. Vale lembrar que para criar triggers é importante saber trabalhar com os identificadores "NEW" e "OLD". Sendo NEW para o novo valor inserido ou atualizado e OLD para o antigo valor, que pode ser antes da atualização e após a exclusão.

Abaixo segue uma breve descrição e Script para cada trigger.

TRIGGER "TRG_EntradaProduto_AI"

Esse trigger será disparado após a inserção de um registro na tabela de "ENTRADA_PRODUTO":

Listagem 6: Script de criação do trigger TRG_EntradaProduto_AI

```
DELIMITER //
CREATE TRIGGER `TRG_EntradaProduto_AI` AFTER INSERT ON `entrada_produto`
FOR EACH ROW
BEGIN
	CALL SP_AtualizaEstoque (new.id_produto, new.qtde, new.valor_unitario);
END //
DELIMITER ;
```

TRIGGER "TRG_EntradaProduto_AU"

Esse trigger será disparado após a atualização de um registro na tabela de "ENTRADA_PRODUTO".

Listagem 7: Script de criação do trigger TRG_EntradaProduto_AU

```
DELIMITER //
CREATE TRIGGER `TRG_EntradaProduto_AU` AFTER UPDATE ON `entrada_produto`
FOR EACH ROW
BEGIN
	CALL SP_AtualizaEstoque (new.id_produto, new.qtde - old.qtde, new.valor_unitario)
END //
DELIMITER ;
```

TRIGGER "TRG_EntradaProduto_AD"

Esse trigger será disparado após a exclusão de um registro na tabela de "ENTRADA_PRODUTO".

Listagem 8: Script de criação do trigger TRG_EntradaProduto_AD

```
DELIMITER //
CREATE TRIGGER `TRG_EntradaProduto_AD` AFTER DELETE ON `entrada_produto`
FOR EACH ROW
BEGIN
	CALL SP_AtualizaEstoque (old.id_produto, old.qtde * -1, old.valor_unitario);
END //
DELIMITER ;
```

TRIGGER "TRG_SaidaProduto_AI"

Esse trigger será disparado após a inserção de um registro na tabela de "SAIDA_PRODUTO".

Listagem 9: Script de criação do trigger TRG_SaidaProduto_AI

```
DELIMITER //
CREATE TRIGGER `TRG_SaidaProduto_AI` AFTER INSERT ON `saida_produto`
FOR EACH ROW
BEGIN
      CALL SP_AtualizaEstoque (new.id_produto, new.qtde * -1, new.valor_unitario);
END //
DELIMITER ;
```

TRIGGER "TRG_ SaidaProduto _AU"

Esse trigger será disparado após a atualização de um registro na tabela "SAIDA_PRODUTO".

Listagem 10: Script de criação do trigger TRG_EntradaProduto_AU

```
DELIMITER //
CREATE TRIGGER `TRG_SaidaProduto_AU` AFTER UPDATE ON `saida_produto`
FOR EACH ROW
BEGIN
      CALL SP_AtualizaEstoque (new.id_produto, old.qtde - new.qtde, new.valor_unitario)
END //
DELIMITER ;
```

TRIGGER "TRG_ SaidaProduto _AD"

Esse trigger será disparado após a exclusão de um registro na tabela de "SAIDA_PRODUTO".

Listagem 11: Script de criação do trigger TRG_EntradaProduto_AD

```
DELIMITER //
CREATE TRIGGER `TRG_SaidaProduto_AD` AFTER DELETE ON `saida_produto`
FOR EACH ROW
BEGIN
      CALL SP_AtualizaEstoque (old.id_produto, old.qtde, old.valor_unitario);
END //
DELIMITER ;
```

Observação: observem que em algumas chamadas do procedure "SP_AtualizaEstoque", antes de passar o parâmetro "qtde" é feita a multiplicação desse valor por -1, essa operação muda o sinal matemático do valor para negativo. Dentro do procedure somamos as quantidades, mas quando passamos o sinal de negativo ocorre então uma subtração dos valores resultando em débito no estoque.

Depois de todo esse trabalho é só cadastrar as compras e as vendas nas respectivas tabelas. Como criamos um trigger para cada evento, todas as ações (INSERT, UPDATE, DELETE) na compra ou venda serão refletidas na tabela "ESTOQUE". Vamos dizer que na teoria não tem como haver "furo" nesse estoque, pelo menos no sistema.

Outra questão interessante é que se somarmos a quantidade em estoque mais a quantidade vendida de um determinado produto, vamos obter a quantidade comprada, exemplo:

- Foram compradas 10 canetas;

- No estoque constam 5 canetas;
- Foram vendidas 5 canetas;
- 5 no estoque + 5 vendidas = 10 compradas .

Vamos testar !!!

id	id_produto	qtde	valor_unitario	data_entrada
1	1	10	2,00	2012-11-08
2	1	15	2,50	2012-11-10
3	2	5	1,00	2012-11-10
4	4	10	3,00	2012-11-10

Figura 2: Produtos comprados na tabela ENTRADA_PRODUTO

id	id_produto	qtde	data_saida	valor_unitario
1	1	3	2012-11-13	2,50
2	4	2	2012-11-13	3,00
3	2	5	2012-11-11	1,00

Figura 3: Produtos vendidos na tabela SAIDA_PRODUTO

id	id_produto	descricao	qtde	valor_unitario
1	1	CANETA	22	2,50
2	2	LAPIS	0	1,00
3	4	LAPISEIRA	8	3,00

Figura 4: Produtos na tabela ESTOQUE

Para esse artigo não foi dada importância à questão de valores do produto, estamos atualizando o valor na tabela de "ESTOQUE" sempre que um produto é inserido, atualizado ou excluído. Mas existem outras formas melhores e mais seguras de controlar esse custo.

Bom pessoal, neste artigo foi demonstrada uma das várias maneiras pelas quais podemos controlar estoques direto no banco de dados sem a necessidade de desenvolver esse controle dentro da aplicação. Existem opiniões que encorajam essa prática com as regras de negócio direto no banco de dados, outros já consideram que essa prática pode ser perigosa pelo fato da aplicação ficar presa a um determinado banco de dados e caso seja necessário a migração para outro SGBD, poderão ocorrer problemas.

Cabe aos colegas decidirem qual a melhor prática!

Espero que tenham apreciado e até a próxima.

Caso surja alguma dúvida meu e-mail é wllfl@ig.com.br. Fiquem à vontade também para usar a seção de comentários.

Abraço a todos!

William

Técnico em Informática, atualmente cursando Tecnologia em Análise e Desenvolvimento de Sistemas. Conhecimentos básicos nas linguagens Delphi, C#, PHP, Java, JQuery e CSS. Banco de dados Firebrid, MySQL, PostgreSQL e SQLServer...

cursos relacionados | posts em sequência | últimos do autor

Administração do Firebird/InterBase

Curso de Administração do Microsoft SQL Server

Ferramentas Administrativas do MySQL

PL/SQL Oracle

Administração do Firebird/InterBase

[Ver todos]