Banco de Dados II

TRIGGERS

Gatilhos

Prof. Jackson Henrique

Introdução

- É muito comum, em aplicações que utilizam bancos 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;

Introdução

- 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.

Introdução

- A “melhor 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**.

Introdução

- Os **TRIGGERS** podem ser executados **ANTES** ou **DEPOIS** das operações:
  - **INSERT**
  - **UPDATE**
  - **DELETE**

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;

Prós e Contras das Triggers

- 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;
- TRIGGERS ainda não podem ser implementadas com a intenção de devolver para o usuário ou para uma aplicação mensagens de erros.

Sintaxe da Trigger

- A sintaxe dos comandos para criar um novo trigger no MySQL é a seguinte:

- CREATE TRIGGER **nome** **momento** **evento**
- ON **tabela**
- FOR EACH ROW
- BEGIN
- /*corpo do código*/
- END

Análise do Código

- 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.**
- **tabela**: nome da tabela a qual o gatilho está associado.

Observações Importantes

- **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.
- Não é possível chamar um Procedimento ou Função, que faça esta tarefa (insert/update ou delete para a mesma tabela), dentro da trigger.

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.

Os registros NEW e OLD

- 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.
- Utilizando o **Banco de Dados Mecanica**, uma trigger pode ser usada para baixar o estoque quando um produto for inserido nos Itens da Venda;
- Analise a Trigger a Seguir:

Exemplo: Baixar Estoque

- delimiter $$
- **CREATE TRIGGER** baixarEstoque **AFTER INSERT ON** itens_venda **FOR EACH ROW**
- **BEGIN**
- **UPDATE** produto **SET** quant_prod = quant_prod - **NEW**.quant_itensvend **WHERE** cod_prod = **NEW**.cod_prod;
- **END;**
- $$ delimiter ;

Utilização do trigger

- Observe que **after insert** significa que o código da trigger só será executado após o **insert** na tabela Itens_Venda
- **For each row** é padrão em todas as trigger;
- Apesar da trigger ser disparada na tabela Itens_venda não é ela que está sendo atualizada, mas sim a tabela **Produto;**

Utilização do trigger

- Observe que o estoque do produto é baixado assim que ele é inserido no itens da venda, mas e se o cliente não quiser mais um item e solicitar a sua retirada da venda?
- Podemos então fazer uma trigger para retornar o valor anterior a execução da trigger baixarEstoque utilizando o OLD;
- Analise o código a seguir:

Utilização do trigger

- delimiter $$
- **CREATE TRIGGER** cancelarBaixaEstoque **AFTER DELETE ON** itens_venda **FOR EACH ROW**
- **BEGIN**
  - **UPDATE** produto **SET** quant_prod = quant_prod + **OLD**.quant_itensvend **WHERE** cod_prod = **OLD**.cod_prod;
- **END;**
- $$ delimiter ;

Trigger com IF

- O gatilho **baixarEstoque** tem um problema. Mesmo que não exista estoque do produto, o estoque é baixado deixando o mesmo negativo na tabela PRODUTO;
- Para resolver esse problema, deve-se implementar uma estrutura de decisão para verificar o estoque antes de inserir o ITENS_VENDA;
- Observe também que o momento deverá ser **modificado** para BEFORE, pois a verificação deve acontecer antes de realizar o INSERT;

Baixar Estoque *versão 2*

- delimiter $$
- **CREATE TRIGGER** baixarEstoque2 **BEFORE INSERT ON** itens_venda **FOR EACH ROW**
- **BEGIN**
- **DECLARE** estoque **INT;**
- **SELECT** quant_prod **INTO** estoque **FROM** produto **WHERE** cod_prod = **NEW**.cod_prod;
- **IF** (estoque >= **NEW**.quant_itensvend) **THEN**
  - **UPDATE** produto **SET** quant_prod = quant_prod - **NEW**.quant_itensvend **WHERE** cod_prod = **NEW**.cod_prod;
- ELSE
  - **SET** new.quant_itensvend = estoque;
  - **UPDATE** produto **SET** quant_prod = quant_prod - **NEW**.quant_itensvend **WHERE** cod_prod = **NEW**.cod_prod;
- END IF;
- END;
- $$ delimiter ;

O ELSE define a quant_itensvend igual ao estoque, assim, só permite inserir a quantidade disponível no ESTOQUE

Exclusão de Gatilho

- Para excluir um gatilho basta utilizar o síntaxe:
- **DROP TRIGGER** nome_do_gatilho;

- **Exemplo:**
  - **DROP TRIGGER** baixarEstoque;

Exercício I – BD_Mecânica 3.0

Refaça a trigger **baixarEstoque2** adicionando as seguinte **ações:**

a) Atualize o **valor total** da VENDA, somando o valor do novo item ao valor total APÓS cada INSERT;

Exercício II

Crie uma triger para **positivar** o **estoque** do PRODUTO após ele ser inserido em **itens da compra;**

a) Atualize a **quant_prod** na tabela PRODUTO, aumentando o **estoque** do produto de acordo o **quant_intenscomp** informada;

b) Atualize o **valor_prod** da tabela PRODUTO, de acordo com o **valor_itenscomp**, adicionando um lucro de 40% em cima do **valor do produto** comprado.

c) Atualize o **valortotal_comp** da tabela **Compra_Produto** de acordo com os itens inseridos na tabela Itens_compra.