INSTITUTO FEDERAL
Rio Grande do Norte | Campus Parnamirim

Triggers (gatilhos)

Prof. Fábio Procópio

fabio.procopio@ifrn.edu.br

Tecnologia em Sistemas para Internet – Programação em Bancos de Dados

Relembrando...

- Na aula passada, falamos sobre Controle de Concorrência em Sistemas de Gerenciamento de Banco de Dados. O assunto é bastante teórico, mas conseguimos apresentar alguns cenários práticos para representar alguns conceitos.

- Nesta aula, falaremos sobre um dos assuntos que acho de extrema importância para programadores de banco de dados: gatilhos. Vamos nessa?!

Introdução

- Trigger (ou gatilho) é um tipo especial de procedimento armazenado que é **disparado**, **automaticamente, quando um evento é executado**;

- O SQL Server considera três tipos de gatilhos
  - **Gatilhos DDL** (Data Definition Language)
    - Invocados quando um evento de definição de dados ocorre no servidor ou no BD
  - **Gatilhos DML** (Data Manipulation Language)
    - Invocados quando um evento de manipulação de dados ocorre no BD
  - **Gatilhos de logon**
    - Disparados quando estabelecida uma sessão de usuário no SQL Server

Trigger DML – 1 de 2

- É disparado quando um evento DML afeta uma tabela definida pelo gatilho;
- Pode ser usado no gerenciamento de regras de negócios e de integridade referencial;
- O gatilho e a instrução que o dispara são tratados como uma transação simples que pode ser revertido dentro do próprio gatilho;
- Pode proteger os dados contra operações incorretas ou mal-intencionadas do tipo INSERT, UPDATE e DELETE;
- Pode avaliar o estado de uma tabela **antes** e **depois** da modificação dos dados e efetuar ações com base nas diferenças identificadas.

Trigger DML – 2 de 2

- Os triggers DML podem ser de três tipos:
  - FOR
  - AFTER
  - INSTEAD OF
- Um trigger **FOR** é o tipo padrão e faz com que o gatilho seja disparado junto com a ação que o invocou;
- Um trigger **AFTER** faz com que o disparo ocorra apenas depois que a ação que o gerou for concluída;
- Um trigger **INSTEAD OF** quando disparado, faz com que ele seja executado no lugar da ação que o invocou.

Sintaxe básica

- Para criar um *trigger*, usa-se o comando **CREATE TRIGGER**, cuja sintaxe básica é:

```
CREATE TRIGGER [nome_trigger]
ON [nome_tabela | nome_visão]
[FOR | AFTER | INSTEAD OF]
AS
    /*Corpo do gatilho*/
```

onde

- **[nome_trigger]**: define um nome para o gatilho a fim de identificá-lo como um objeto do banco de dados;
- **[nome_tabela | nome_visao]**: indica à qual tabela (ou visão) o gatilho está associado;
- **[FOR | AFTER | INSTEAD OF]**: define o momento em que o disparo ocorrerá.

Tabelas especiais – 1 de 3

- Durante a execução de um gatilho DML, duas tabelas especiais são criadas: **inserted** e **deleted**;
- Elas são gerenciadas, automaticamente, pelo SQL Server e estão armazenadas em memória;
- São usadas para testar efeitos de algumas modificações nos dados e possuem a mesma estrutura da tabela base do gatilho;
- **Não é possível modificar os dados** dessas tabelas e **nem executar operações DDL** sobre elas, como ALTER TABLE, CREATE INDEX, etc;
- Podem ser usadas para: 1) testar se há erros em uma transação; 2) aplicar ações com base em erros identificados; 3) identificar as diferenças de uma tabela antes e depois de uma alteração de dados; 4) executar ações com base nas diferenças detectadas.

Tabelas especiais – 2 de 3

- **Tabela inserted**
  - Armazena cópias das linhas que foram afetadas durante as instruções INSERT e UPDATE;
  - Durante uma transação de inserção ou de atualização, novas linhas são adicionadas à tabela inserted e também à tabela a qual o gatilho está associado;
  - As linhas na tabela inserted são cópias das linhas adicionadas à tabela na qual o gatilho foi definido.

- **Tabela deleted**
  - Armazena cópias das linhas que foram afetadas durante as instruções DELETE e UPDATE;
  - Durante uma transação da instrução DELETE ou UPDATE, as linhas são excluídas da tabela de gatilhos e transferidas para a tabela deleted;
  - Em geral, as linhas da tabela deleted não possuem nenhuma linha em comum.

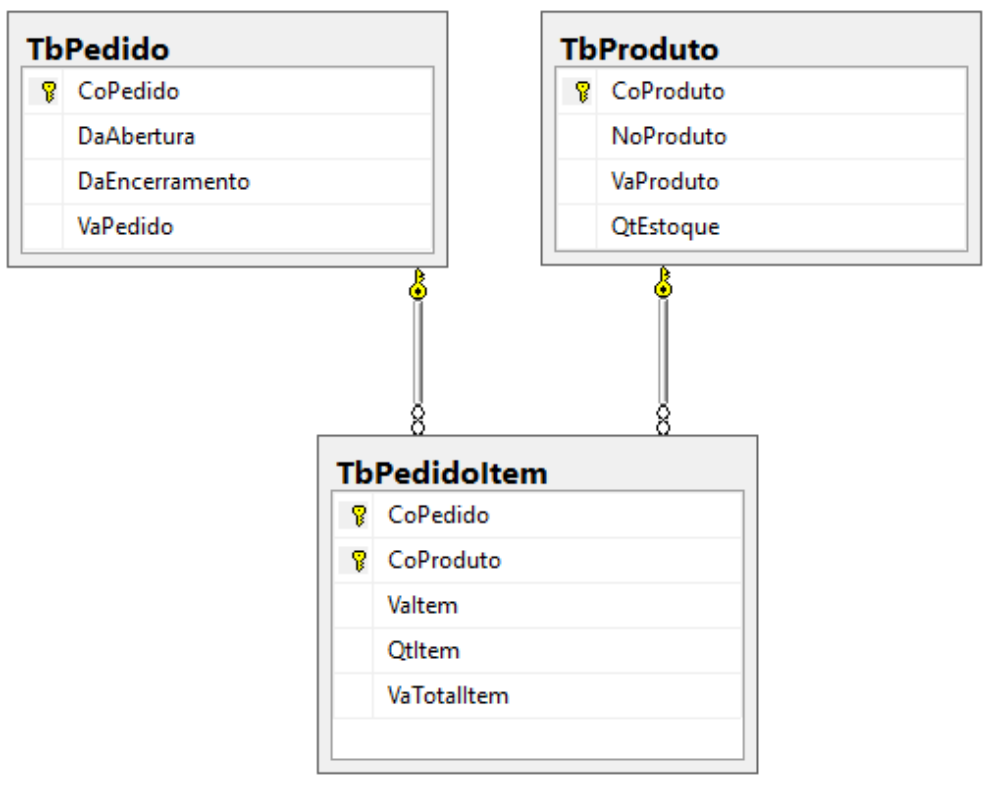
Tabelas especiais – 3 de 3

	inserted	deleted
INSERT	X	
UPDATE	X	X
DELETE		X

Dados das tabelas especiais	
INSERT	A tabela **inserted** possui os dados que estão sendo inseridos
UPDATE	A tabela **deleted** terá os dados antigos e a tabela **inserted** terá os dados novos.
DELETE	A tabela **deleted** terá dados que foram deletados.

Cenário para simulação

- Restaure o banco de dados dbVendas a partir deste arquivo de *backup*;
- O DER do banco de dados é mostrado na figura abaixo:



Cenários

1. Criar um gatilho para toda vez que um item for inserido em um pedido, o valor total do pedido seja calculado. Além disso, o estoque do produto deve ser baixado.

2. Criar um gatilho para toda vez que um item for excluído, a quantidade do item retorne para o estoque e o valor do pedido seja recalculado para um valor a menor.

3. Criar um gatilho para toda vez que um item for substituído por um outro, a quantidade do item anterior retorne para o estoque e seja dado baixa no estoque do novo item. Além disso, o gatilho deve recalcular o valor do pedido.

4. Criar um gatilho para bloquear as tentativas de exclusão de produtos, mesmo que não haja itens desse produto na tabela de itens de pedido.

5. Criar um gatilho para bloquear as tentativas de um alteração que afete todos os registros da tabela de produtos.

Cenário 1

```
CREATE TRIGGER tgIncluiItem
ON TbPedidoItem
AFTER INSERT
AS

DECLARE @CoPedido INT, @CoProduto SMALLINT,@QtItem SMALLINT, @VaItem SMALLMONEY

SELECT @CoPedido = CoPedido, @CoProduto = CoProduto,
       @QtItem = QtItem, @VaItem = VaItem
  FROM inserted

/*Calcula valor do pedido*/
UPDATE TbPedido
   SET VaPedido = VaPedido + (@QtItem  * @VaItem)
 WHERE CoPedido = @CoPedido

/*Baixa estoque do produto*/
UPDATE TbProduto
   SET QtEstoque = QtEstoque - @QtItem
 WHERE CoProduto = @CoProduto
```

Cenário 2 – 1 de 2

```
CREATE TRIGGER tgCancelaItem
ON TbPedidoItem
AFTER DELETE
AS

DECLARE @CoPedido INT, @CoProduto SMALLINT,  @QtItem SMALLINT,
        @DaEncerramento DATETIME, @VaItem SMALLMONEY

SELECT @CoPedido  = d.CoPedido, @CoProduto   = d.CoProduto,
       @QtItem      = d.QtItem, @VaItem = d.VaItem,
       @DaEncerramento  = DaEncerramento
  FROM deleted AS d
 INNER JOIN TbPedido AS p
    ON d.CoPedido = p.CoPedido
```

Cenário 2 – 2 de 2

```
IF @DaEncerramento IS NULL
    BEGIN
      UPDATE TbPedido
         SET VaPedido = VaPedido - (@QtItem  * @VaItem)
       WHERE CoPedido = @CoPedido

      UPDATE TbProduto
         SET QtEstoque = QtEstoque + @QtItem
       WHERE CoProduto = @CoProduto
   END
ELSE
   BEGIN
      RAISERROR ('Não é permitido excluir itens de um pedido encerrado.',
                 16,  1);
      ROLLBACK /*Cancela a transação */
   END
```

Cenário 3 – 1 de 2

```
CREATE TRIGGER tgAlteraItem
ON TbPedidoItem
AFTER UPDATE
AS

DECLARE @CoPedido INT, @CoProduto SMALLINT, @QtItem SMALLINT,
        @VaItem SMALLMONEY

SELECT @CoPedido  = CoPedido, @CoProduto   = CoProduto,
       @QtItem      = QtItem, @VaItem = VaItem
  FROM deleted

/*Recalcula valor do pedido*/
UPDATE TbPedido
   SET VaPedido = VaPedido - (@QtItem * @VaItem)
 WHERE CoPedido = @CoPedido
```

Cenário 3 – 2 de 2

```
/*Devolve produto ao estoque*/
UPDATE TbProduto
   SET QtEstoque = QtEstoque + @QtItem
 WHERE CoProduto = @CoProduto

SELECT @CoProduto   = CoProduto, @QtItem = QtItem, @VaItem = VaItem
  FROM inserted

/*Recalcula valor do pedido*/
UPDATE TbPedido
   SET VaPedido = VaPedido + (@QtItem * @VaItem)
 WHERE CoPedido = @CoPedido

/*Baixa estoque do produto*/
UPDATE TbProduto
   SET QtEstoque = QtEstoque - @QtItem
 WHERE CoProduto = @CoProduto
```

Cenário 4

```
CREATE TRIGGER tgProdutoExclui
ON TbProduto
INSTEAD OF DELETE
AS

  RAISERROR ('Você não tem permissão para excluir produtos.', 16, 1);
```

Exemplo 5

```
CREATE TRIGGER tgProdutoAltera
ON TbProduto
AFTER UPDATE
AS
  IF (SELECT COUNT(*) FROM deleted) =
     (SELECT COUNT(*) FROM TbProduto) AND
     (SELECT COUNT(*) FROM TbProduto) > 1
      BEGIN
       RAISERROR ('Você não tem permissão para alterar todos os
                   registros de uma só vez.',  16, 1);
       ROLLBACK
      END
```

Ativar e desativar gatilhos

- A sintaxe para desativar um gatilho é a seguinte:

```
ALTER TABLE [nome_tabela] DISABLE TRIGGER [nome_trigger]
```

- A sintaxe para ativar um gatilho é a seguinte:

```
ALTER TABLE [nome_tabela] ENABLE TRIGGER [nome_trigger]
```

Exercícios de fixação

Usando o banco de dados dbVendas:

1. Na tabela TbPedidoItem, crie um gatilho para toda vez que um item for inserido (ou alterado), o campo VaTotalItem seja (re)calculado.

2. Em TbProduto, crie um gatilho para não permitir estoques inferiores a zero.

3. Ainda na tabela TbProduto, adicione um campo chamado QtEstoqueMinimo e atualize-o definindo um valor que você considerar adequado. Em seguida, crie um gatilho para mostrar uma mensagem todas as vezes que o estoque foi alterado para um valor inferior ao campo QtEstoqueMinimo.

Referências

1. DEVMEDIA. **Triggers no SQL Server: teoria e prática aplicada em uma situação real.** Disponível em: http://www.devmedia.com.br/triggers-no-sql-server-teoria-e-pratica-aplicada-em-uma-situacao-real/28194. Acessado em: 22 out. 2019.

2. MSDN. **Gatilhos DDL.** Disponível em: https://msdn.microsoft.com/pt-br/library/ms175941.aspx. Acessado em: 22 out. 2019.

3. MICROSOFT. **CREATE TRIGGER.** Disponível em: https://msdn.microsoft.com/pt-BR/library/ms189799.aspx. Acessado em: 22 out. 2019.

4. MICROSOFT. **Usar as tabelas incluída e excluída.** Disponível em: https://msdn.microsoft.com/pt-br/library/ms191300.aspx. Acessado em: 22 out. 2019.

5. MICROSOFT. **Usando RAISERROR.** Disponível em: https://technet.microsoft.com/pt-br/library/ms177497(v=sql.105).aspx. Acessado em: 22 out. 2019.