Banco de Dados

Triggers (*Gatilhos*)

Prof. Fabiano Papaiz
IFRN

Triggers

- Uma Trigger (ou gatilho) é uma estrutura presente na maioria dos SGBD's que permite que um determinado evento (código SQL) seja executado automaticamente mediante alguma ação ocorrida em uma tabela no banco de dados
  - Geralmente em ações de *insert*, *update* ou *delete*.
- Geralmente utilizamos triggers quando desejamos realizar:
  - Validações de dados
  - Garantir a consistência dos dados
  - Restrições de acesso
  - Registros das atividades dos usuários (Log de sistema) etc

TRIGGERS

- A utilização de triggers envolve duas etapas:

1. A criação da trigger através de um comando SQL
   - *CREATE TRIGGER*

2. A ação que irá executar (disparar) o código da trigger
   - Envio de comandos INSERT, UPDATE ou DELETE para o banco de dados (edição de registros)

TRIGGERS

- A sintaxe para criação de triggers no *SQL Server* é a seguinte:

```
CREATE TRIGGER nome_da_trigger ON nome_da_tabela
FOR INSERT | UPDATE | DELETE  AS...
```

```
CREATE TRIGGER nome_da_trigger ON nome_da_tabela
AFTER INSERT | UPDATE | DELETE AS...
```

```
CREATE TRIGGER nome_da_trigger ON nome_da_tabela
INSTEAD OF INSERT | UPDATE | DELETE AS...
```

Triggers

- FOR / AFTER / INSTEAD OF
  - FOR: é o valor padrão e faz com o que o código seja disparado antes da ação que o disparou (ex. validação de dados)
  - AFTER: faz com que o código seja disparado somente após a ação que o gerou ser concluída (ex. log de atividades)
  - INSTEAD OF: faz com que o código seja executado no lugar da ação (*insert*, *update* ou *delete*) que o disparou
- INSERT / UPDATE / DELETE
  - Uma ou várias dessas opções (separadas por vírgula) devem ser definidas para indicar qual será a ação que irá disparar a trigger

Triggers

- Exemplo prático:
- Criar um novo BD e as tabelas a seguir.

```
create table produto (
    id int identity not null,
    nome varchar(50) not null,
    quantidade_em_estoque int,
    primary key(id)
)

create table compra (
    id int identity not null,
    data date not null,
    produto_id int not null,
    quantidade int not null,
    primary key(id),
    foreign key (produto_id) references produto(id)
)
```

Triggers

- Agora iremos criar uma trigger que irá atualizar a quantidade em estoque do produto para o qual foi realizado uma compra
- A ideia aqui é que a cada registro de compra que for inserido no BD, a quantidade atual em estoque do produto comprado deverá ser automaticamente acrescida da quantidade comprada

TRIGGERS

- Código de criação da trigger

```
create trigger atualizar_estoque on compra
after insert
as
    --variavel para armazenar a qtd comprada e o id do produto
    declare @qtd_comprada int
    declare @produto_id int

    --guarda na variavel a quantidade comprada
    select @qtd_comprada = quantidade
          ,@produto_id = produto_id
    from inserted

    --atualiza a qtd em estoque na tabela Produto
    update produto
    set quantidade_em_estoque = quantidade_em_estoque + @qtd_comprada
    where id = @produto_id
```

TRIGGERS

- Testando a trigger

```
--inclui produtos
insert into produto(nome, quantidade_em_estoque)
       values('Arroz', 0)
insert into produto(nome, quantidade_em_estoque)
       values('Carne', 0)

--inclui compras
insert into compra(data, produto_id, quantidade)
       values( getdate(), 1, 5)
insert into compra(data, produto_id, quantidade)
       values( getdate(), 2, 10)
```

- Após executar o código anterior, verifique as quantidades em estoques dos produtos inseridos

TRIGGERS

- Explicando a trigger

```
create trigger atualizar_estoque on compra
after insert
as
    --variavel para armazenar a qtd comprada e o id do produto
    declare @qtd_comprada int
    declare @produto_id int

    --guarda na variavel a quantida
    select @qtd_comprada = quantida
          ,@produto_id = produto_id
    from inserted

    --atualiza a qtd em estoque na tabela Produto
    update produto
    set quantidade_em_estoque = quantidade_em_estoque + @qtd_comprada
    where id = @produto_id
```

Primeiro declaramos variáveis para armazenarmos o id do produto e a quantidade comprada. Devemos utilizar o @ no início do nome das variáveis

TRIGGERS

- Explicando a trigger

```
create trigger atualizar_estoque on compra
after insert
as
    --variave
    declare @
    declare @

    --guarda na variavel a quantidade comprada
    select @qtd_comprada = quantidade
          ,@produto_id = produto_id
    from inserted

    --atualiza a qtd em estoque na tabela Produto
    update produto
    set quantidade_em_estoque = quantidade_em_estoque + @qtd_comprada
    where id = @produto_id
```

Depois obtemos o id do produto e a quantidade que foram inseridos na tabela *Compra* através da tabela temporária chamada *INSERTED*. Esta tabela existe somente dentro da trigger e possui apenas uma linha contendo os dados que acabaram de ser inseridos na tabela *Compra*.

TRIGGERS

- Explicando a trigger

```
create trigger atualizar_estoque on compra
after insert
as
    --variavel para armazenar a qtd comprada e o id do produto
    declare @qtd_comprada int
    declare @produto_id int

    --guarda na variavel a quantidade comprada
```

Agora basta atualizar a quantidade atual em estoque do produto comprado na tabela *Produto*.

```
    --atualiza a qtd em estoque na tabela Produto
    update produto
    set quantidade_em_estoque = quantidade_em_estoque + @qtd_comprada
    where id = @produto_id
```

Triggers

- Agora iremos criar uma nova trigger que irá atualizar a quantidade em estoque do produto quando um registro de compra for excluído da tabela *Compra*.
- A ideia aqui é que a cada registro de compra que for excluído no BD, a quantidade atual em estoque do produto comprado deverá ser automaticamente subtraída da quantidade comprada que foi excluída.

TRIGGERS

- Código de criação da trigger

```
create trigger atualizar_estoque_exclusao on compra
after delete
as
    --variavel para armazenar a qtd comprada e o id do produto
    declare @qtd_comprada int
    declare @produto_id int

    --guarda na variavel a quantidade comprada que foi excluida
    select @qtd_comprada = quantidade
          ,@produto_id = produto_id
      from deleted

    --atualiza a qtd em estoque na tabela Produto
    update produto
    set quantidade_em_estoque = quantidade_em_estoque - @qtd_comprada
    where id = @produto_id
```

Triggers

- Testando a trigger

```
--exclui a compra com id=1
delete from compra where id = 1

--exclui a compra com id=2
delete from compra where id = 2
```

- Após executar o código anterior, verifique as quantidades em estoques dos produtos inseridos

TRIGGERS

- DESAFIO!!!
- Agora crie uma nova trigger que irá atualizar a quantidade em estoque do produto quando um registro de compra for alterado na tabela *Compra*.
- Importante:
  - Para que a quantidade atual seja calculada corretamente devemos subtrair a quantidade anterior e adicionar a quantidade alterada
  - Na trigger de UPDATE, podemos obter os valores anteriores através da tabela temporária DELETED e os valores alterados através da tabela temporária INSERTED

FIM