Bài 10. Lập trình Trigger trong SQL Server.

- Mục đích, yêu cầu: Cung cấp cho sinh viên kiến thức về lập trình trigger trong SQL Server.

- Hình thức tổ chức dạy học: Lý thuyết, trực tiếp + trực tuyến + tự học

- Thời gian: Lý thuyết(trên lớp: 3; online: 2) Tự học, tự nghiên cứu: 10

- Nội dung chính:

Lập trình Trigger trong SQL Server

I. Trigger

Ta đã biết các ràng buộc được sử dụng để đảm bảo tính toàn vẹn dữ liệu trong cơ sở dữ liệu. Một đối tượng khác cũng thường được sử dụng trong các cơ sở dữ liệu cũng với mục đích này là các trigger. Cũng tương tự như thủ tục lưu trữ, một trigger là một đối tượng chứa một tập các câu lệnh SQL và tập các câu lệnh này sẽ được thực thi khi trigger được gọi. Điểm khác biệt giữa thủ tục lưu trữ và trigger là: các thủ tục lưu trữ được thực thi khi người sử dụng có lời gọi đến chúng còn các trigger lại được "gọi" tự động khi xảy ra những giao dịch làm thay đổi dữ liệu trong các bảng.

Mỗi một trigger được tạo ra và gắn liền với một bảng nào đó trong cơ sở dữ liệu. Khi dữ liệu trong bảng bị thay đổi (tức là khi bảng chịu tác động của các câu lệnh INSERT, UPDATE hay DELETE) thì trigger sẽ được tự đông kích hoạt.

Sử dụng trigger một cách hợp lý trong cơ sở dữ liệu sẽ có tác động rất lớn trong việc tăng hiệu năng của cơ sở dữ liệu. Các trigger thực sự hữu dụng với những khả năng sau:

- Một trigger có thể nhận biết, ngăn chặn và huỷ bỏ được những thao tác làm thay đổi trái phép dữ liệu trong cơ sở dữ liệu.

- Các thao tác trên dữ liệu (xoá, cập nhật và bổ sung) có thể được trigger phát hiện ra và tự động thực hiện một loạt các thao tác khác trên cơ sở dữ liệu nhằm đảm bảo tính hợp lệ của dữ liệu.

- Thông qua trigger, ta có thể tạo và kiểm tra được những mối quan hệ phức tạp hơn giữa các bảng trong cơ sở dữ liệu mà bản thân các ràng buộc không thể thực hiện được.

1.1. Định nghĩa trigger

Một trigger là một đối tượng gắn liền với một bảng và được tự động kích hoạt khi xảy ra những giao dịch làm thay đổi dữ liệu trong bảng. Định nghĩa một trigger bao gồm các yếu tố sau:

• Trigger sẽ được áp dụng đối với bảng nào?

• Trigger được kích hoạt khi câu lệnh nào được thực thi trên bảng: `INSERT`, `UPDATE`, `DELETE`?

• Trigger sẽ làm gì khi được kích hoạt?

Câu lệnh `CREATE TRIGGER` được sử dụng để đinh nghĩa trigger và có cú pháp như sau:

```
CREATE TRIGGER tên_trigger ON tên_bảng FOR
{[INSERT][,][UPDATE][,][DELETE]} AS [IF UPDATE(tên_cột)
[AND UPDATE(tên_cột)|OR UPDATE(tên_cột)]...]
các_câu_lệnh_của_trigger
```

Ta định nghĩa các bảng như sau:

Bảng MATHANG lưu trữ dữ liệu về các mặt hàng:

```
CREATE TABLE mathang (
mahang NVARCHAR(5) PRIMARY KEY,/*mã hàng*/
tenhang NVARCHAR(50) NOT NULL, /*tên hàng*/
soluong INT, /*số lượng hàng hiện có*/
)
```

Bảng NHATKYBANHANG lưu trữ thông tin về các lần bán hàng

```
CREATE TABLE nhatkybanhang (
stt INT IDENTITY PRIMARY KEY,
ngay DATETIME, /*ngày bán hàng*/
```

```
nguoimua NVARCHAR(30),/*tên người mua hàng*/
mahang NVARCHAR(5)/*mã mặt hàng được bán*/ FOREIGN KEY
REFERENCES mathang(mahang),
soluong INT, /*số lượng hàng được bán*/
giaban MONEY /*giá bán hàng*/)
```

Câu lệnh dưới đây định nghĩa trigger *trg_nhatkybanhang_insert*. Trigger này có chức năng tự động giảm số lượng hàng hiện có khi một mặt hàng nào đó được bán (tức là khi câu lệnh INSERT được thực thi trên bảng NHATKYBANHANG).

```
CREATE TRIGGER trg_nhatkybanhang_insert
ON nhatkybanhang
FOR INSERT
AS
UPDATE mathang SET
mathang.soluong=mathang.soluonginserted.soluong FROM
mathang INNER JOIN inserted ON
mathang.mahang=inserted.mahang
```

Với trigger vừa tạo ở trên, nếu dữ liệu trong bảng MATHANG là:

MAHANG	TENHANG	SOLUONG
H1	Xà phòng	30
H2	Kem đánh răng	45

thì sau khi ta thực hiện câu lênh:

```
INSERT INTO
nhatkybanhang(ngay,nguoimua,mahang,soluong,giaban)
VALUES('5/5/2004','Tran Ngoc Thanh','H1',10,5200)
```

dữ liệu trong bảng MATHANG sẽ như sau:

MAHANG	TENHANG	SOLUONG
H1	Xà phòng	20
H2	Kem đánh răng	45

Trong câu lệnh `CREATE TRIGGER` ở ví dụ trên, sau mệnh đề ON là tên của bảng mà trigger cần tạo sẽ tác động đến. Mệnh đề tiếp theo chỉ định câu lệnh sẽ kích

hoạt trigger FOR INSERT). Ngoài INSERT, ta còn có thể chỉ định UPDATE hoặc DELETE cho mệnh đề này, hoặc có thể kết hợp chúng lại với nhau. Phần thân của trigger nằm sau từ khoá AS bao gồm các câu lệnh mà trigger sẽ thực thi khi được kích hoạt.

Chuẩn SQL định nghĩa hai bảng logic INSERTED và DELETED để sử dụng trong các trigger. Cấu trúc của hai bảng này tương tự như cấu trúc của bảng mà trigger tác động. Dữ liệu trong hai bảng này tuỳ thuộc vào câu lệnh tác động lên bảng làm kích hoạt trigger; cụ thể trong các trường hợp sau:

• Khi câu lệnh DELETE được thực thi trên bảng, các dòng dữ liệu bị xoá sẽ được sao chép vào trong bảng DELETED. Bảng INSERTED trong trường hợp này không có dữ liệu.

• Dữ liệu trong bảng INSERTED sẽ là dòng dữ liệu được bổ sung vào bảng gây nên sự kích hoạt đối với trigger bằng câu lệnh INSERT. Bảng DELETED trong trường hợp này không có dữ liệu.

• Khi câu lệnh UPDATE được thực thi trên bảng, các dòng dữ liệu cũ chịu sự tác động của câu lệnh sẽ được sao chép vào bảng DELETED, còn trong bảng INSERTED sẽ là các dòng sau khi đã được cập nhật.

1.2. Sử dụng mệnh đề IF UPDATE trong trigger

Thay vì chỉ định một trigger được kích hoạt trên một bảng, ta có thể chỉ định trigger được kích hoạt và thực hiện những thao tác cụ thể khi việc thay đổi dữ liệu chỉ liên quan đến một số cột nhất định nào đó của cột. Trong trường hợp này, ta sử dụng mệnh đề IF UPDATE trong trigger. IF UPDATE không sử dụng được đối với câu lệnh DELETE.

Xét lại ví dụ với hai bảng MATHANG và NHATKYBANHANG, trigger dưới đây được kích hoạt khi ta tiến hành cập nhật cột SOLUONG cho một bản ghi của bảng NHATKYBANHANG (lưu ý là chỉ cập nhật đúng một bản ghi)

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong
ON nhatkybanhang
FOR UPDATE
AS
IF UPDATE(soluong)
    UPDATE mathang SET mathang.soluong=mathang.soluong-
    (inserted.soluong-deleted.soluong)
      FROM (deleted INNER JOIN inserted ON deleted.stt=
    inserted.stt) INNER JOIN mathang ON mathang.mahang
    = deleted.mahang
```

Với trigger ở ví dụ trên, câu lệnh:

```
UPDATE nhatkybanhang SET soluong=soluong+20 WHERE stt=1
```

sẽ kích hoạt trigger ứng với mệnh đề IF UPDATE (soluong) và câu lệnh `UPDATE` trong trigger sẽ được thực thi. Tuy nhiên câu lệnh:

```
UPDATE nhatkybanhang SET nguoimua='Mai Hữu Toàn' WHERE
stt=3
```

lại không kích hoạt trigger này.

Mệnh đề IF UPDATE có thể xuất hiện nhiều lần trong phần thân của trigger. Khi đó, mệnh đề IF UPDATE nào đúng thì phần câu lệnh của mệnh đề đó sẽ được thực thi khi trigger được kích hoạt.

Giả sử ta định nghĩa bảng R như sau:

```
CREATE TABLE R ( A INT, B INT, C INT )
```

và trigger `trg_R_update`cho bảng R:

```
CREATE TRIGGER trg_R_test
ON R
FOR UPDATE
AS IF UPDATE(A)
Print 'A updated' IF UPDATE(C) Print 'C updated'
```

Câu lệnh

```
UPDATE R SET A=100 WHERE A=1
```

sẽ kích hoạt trigger và cho kết quả là: A updated và câu lệnh:

```
UPDATE R SET C=100 WHERE C=2
```

cũng kích hoạt trigger và cho kết quả là: C updated còn câu lệnh:

```
UPDATE R SET B=100 WHERE B=3
```

hiển nhiên sẽ không kích hoạt trigger

1.3. ROLLBACK TRANSACTION và trigger

Một trigger có khả năng nhận biết được sự thay đổi về mặt dữ liệu trên bảng dữ liệu, từ đó có thể phát hiện và huỷ bỏ những thao tác không đảm bảo tính toàn vẹn dữ liệu. Trong một trigger, để huỷ bỏ tác dụng của câu lệnh làm kích hoạt trigger, ta sử dụng câu lệnh(1):

```
ROLLBACK TRANSACTION
```

Nếu trên bảng MATHANG, ta tạo một trigger như sau:

```
CREATE TRIGGER trg_mathang_delete
ON mathang
FOR DELETE
AS
ROLLBACK TRANSACTION
```

Thì câu lệnh `DELETE` sẽ không thể có tác dụng đối với bảng `MATHANG`. Hay nói cách khác, ta không thể xoá được dữ liệu trong bảng.

Trigger dưới đây được kích hoạt khi câu lệnh INSERT được sử dụng để bổ sung một bản ghi mới cho bảng NHATKYBANHANG. Trong trigger này kiểm tra điều kiện hợp lệ của dữ liệu là số lượng hàng bán ra phải nhỏ hơn hoặc bằng số lượng hàng hiện có.

Nếu điều kiện này không thoả mãn thì huỷ bỏ thao tác bổ sung dữ liệu.

```
CREATE TRIGGER trg_nhatkybanhang_insert
ON NHATKYBANHANG
FOR INSERT
AS
DECLARE @sl_co int /*Số lượng hàng hiện có */
DECLARE @sl_ban int /* Số lượng hàng được bán */
DECLARE @mahang nvarchar(5) /* Mã hàng được bán */
SELECT @mahang=mahang,@sl_ban=soluong FROM inserted
SELECT @sl_co = soluong FROM mathang where
mahang=@mahang /*Nếu số lượng hàng hiện có nhỏ hơn số
lượng bán thì huỷ bỏ thao tác bổ sung dữ liệu */
IF @sl_co<@sl_ban
    ROLLBACK TRANSACTION /* Nếu dữ liệu hợp lệ
    thì giảm số lượng hàng hiện có */
ELSE
    UPDATE mathang SET soluong=soluong-@sl_ban WHERE
mahang=@mahang
```

1.4. Sử dụng trigger trong trường hợp câu lệnh INSERT, UPDATE và DELETE có tác động đến nhiều dòng dữ liệu

Trong các ví dụ trước, các trigger chỉ thực sự hoạt động đúng mục đích khi các câu lệnh kích hoạt trigger chỉ có tác dụng đối với đúng một dòng dữ liêu. Ta có thể nhận thấy là câu lệnh UPDATE và DELETE thường có tác dụng trên nhiều dòng, câu lệnh INSERT mặc dù ít rơi vào trường hợp này nhưng không phải là không gặp; đó là khi ta sử dụng câu lệnh có dạng INSERT INTO ... SELECT ... Vậy làm thế nào để trigger hoạt động đúng trong trường hợp những câu lệnh có tác động lên nhiều dòng dữ liệu?

Có hai giải pháp có thể sử dụng đối với vấn đề này:

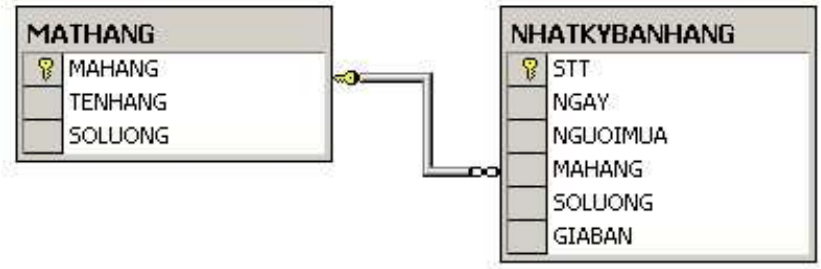
• Sử dụng truy vấn con.

• Sử dụng biến con trỏ.

1.5. Sử dụng truy vấn con

Ta hình dung vấn đề này và cách khắc phục qua ví dụ dưới đây:

Ta xét lại trường hợp của hai bảng MATHANG và NHATKYBANHANG như sơ đồ dưới đây:



MAHANG	TENHANG	SOLUONG
H1	Xà phòng	30
H2	Kem đánh răng	45

STT	NGAY	NGUOIMUA	MAHANG	SOLUONG	GIABAN
1	1-1-2004	Ha	H1	10	10000.0000
2	2-2-2004	Phong	H2	20	5000.0000
3	3-3-2004	Thuy	H2	30	6000.0000

Trigger dưới đây cập nhật lại số lượng hàng của bảng `MATHANG` khi câu lệnh `UPDATE` được sử dụng để cập nhật cột `SOLUONG` của bảng `NHATKYBANHANG`.

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong
ON nhatkybanhang
FOR UPDATE
AS
IF UPDATE(soluong)
    UPDATE mathang SET mathang.soluong=mathang.soluong-
      (inserted.soluong-deleted.soluong) FROM (deleted
    INNER JOIN inserted ON deleted.stt = inserted.stt)
    INNER JOIN mathang ON mathang.mahang=deleted.mahang
```

thì dữ liệu trong hai bảng `MATHANG` và `NHATKYBANHANG` sẽ là:

MAHANG	TENHANG	SOLUONG
H1	Xà phòng	20
H2	Kem đánh răng	45

Bảng MATHANG

STT	NGAY	NGUOIMUA	MAHANG	SOLUONG	GIABAN
1	1-1-2004	Ha	H1	20	10000.0000
2	2-2-2004	Phong	H2	20	5000.0000
3	3-3-2004	Thuy	H2	30	6000.0000
4	4-4-2004	Dung	H1	40	9000.0000

Bảng NHATKYBANHANG

Tức là số lượng của mặt hàng có mã *H1*đã được giảm đi *10*. Nhưng nếu thực hiện tiếp câu lệnh:

```
UPDATE nhatkybanhang SET soluong=soluong + 5 WHERE
mahang='H2'
```

dữ liệu trong hai bảng sau khi câu lệnh thực hiện xong sẽ như sau:

MAHANG	TENHANG	SOLUONG
H1	Xà phòng	20
H2	Kem đánh răng	40

Bảng MATHANG

STT	NGAY	NGUOIMUA	MAHANG	SOLUONG	GIABAN
1	1-1-2004	Ha	H1	20	10000.0000
2	2-2-2004	Phong	H2	25	5000.0000
3	3-3-2004	Thuy	H2	35	6000.0000

Bảng NHATKYBANHANG

Ta có thể nhận thấy số lượng của mặt hàng có mã *H2*còn lại *40*(giảm đi 5) trong khi đúng ra phải là *35*(tức là phải giảm 10). Như vậy, trigger ở trên không hoạt động đúng trong trường hợp này.

Để khắc phục lỗi gặp phải như trên, ta định nghĩa lại trigger như sau:

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong
ON nhatkybanhang
FOR UPDATE
AS
IF UPDATE(soluong)
    UPDATE mathang SET mathang.soluong=mathang.soluong-
(SELECT SUM(inserted.soluong-deleted.soluong) FROM
inserted INNER JOIN deleted ON inserted.stt=deleted.stt
```

```
WHERE inserted.mahang = mathang.mahang) WHERE
mathang.mahang IN (SELECT mahang FROM inserted)
```

hoặc:

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong
ON nhatkybanhang
FOR UPDATE
AS
IF UPDATE(soluong) /* Nếu số lượng dòng được cập nhật
bằng 1 */
    IF @@ROWCOUNT = 1
      BEGIN
        UPDATE mathang SET
        mathang.soluong=mathang.soluong -
(inserted.soluong-deleted.soluong) FROM (deleted INNER
JOIN inserted ON deleted.stt = inserted.stt) INNER JOIN
mathang ON mathang.mahang = deleted.mahang
      END
    ELSE
      BEGIN
        UPDATE mathang SET mathang.soluong=
        mathang.soluong -
(SELECT SUM(inserted.soluong-deleted.soluong) FROM
inserted INNER JOIN deleted ON inserted.stt=deleted.stt
WHERE inserted.mahang = mathang.mahang) WHERE
mathang.mahang IN (SELECT mahang FROM inserted)

END
```

II. Sử dụng biến con trỏ

Một cách khác để khắc phục lỗi xảy ra như trong ví dụ 5.17 là sử dụng con trỏ để duyệt qua các dòng dữ liệu và kiểm tra trên từng dòng. Tuy nhiên, sử dụng biến con trỏ trong trigger là giải pháp nên chọn trong trường hợp thực sự cần thiết.

Một biến con trỏ được sử dụng để duyệt qua các dòng dữ liệu trong kết quả

của một truy vấn và được khai báo theo cú pháp như sau:

```
DECLARE tên_con_trỏ CURSOR FOR câu_lệnh_SELECT
```

Trong đó câu lệnh SELECT phải có kết quả dưới dạng bảng. Tức là trong câu lệnh không sử dụng mệnh đề COMPUTE và INTO.

Để mở một biến con trỏ ta sử dụng câu lệnh:

```
OPEN tên_con_trỏ
```

Để sử dụng biến con trỏ duyệt qua các dòng dữ liệu của truy vấn, ta sử dụng câu lệnh FETCH. Giá trị của biến trạng thái @@FETCH_STATUS bằng không nếu chưa duyệt hết các dòng trong kết quả truy vấn.

Câu lệnh FETCH có cú pháp như sau:

```
FETCH [[NEXT|PRIOR|FIST|LAST] FROM] tên_con_trỏ [INTO
danh_sách_biến ]
```

Trong đó các biến trong danh sách biến được sử dụng để chứa các giá trị của các trường ứng với dòng dữ liệu mà con trỏ trỏ đến. Số lượng các biến phải bằng với số lượng các cột của kết quả truy vấn trong câu lệnh DECLARE CURSOR.

Tập các câu lệnh trong ví dụ dưới đây minh hoạ cách sử dụng biến con trỏ để duyệt qua các dòng trong kết quả của câu lệnh SELECT

```
DECLARE contro CURSOR FOR SELECT mahang,tenhang,soluong
FROM mathang
OPEN contro
DECLARE @mahang NVARCHAR(10)
DECLARE @tenhang NVARCHAR(10)
DECLARE @soluong INT
/*Bắt đầu duyệt qua các dòng trong kết quả truy vấn*/
FETCH NEXT FROM contro INTO @mahang,@tenhang,@soluong
WHILE @@FETCH_STATUS=0 BEGIN PRINT 'Ma hang:'+@mahang
```

```
PRINT 'Ten hang:'+@tenhang
PRINT 'So luong:'+STR(@soluong)
FETCH NEXT FROM contro INTO @mahang,@tenhang,@soluong
END
/*Đóng con trỏ và giải phóng vùng nhớ*/
CLOSE contro
DEALLOCATE contro
```

Trigger dưới đây là một cách giải quyết khác của trường hợp được đề cập trên

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong
ON nhatkybanhang
FOR UPDATE
AS
IF UPDATE(soluong)
BEGIN
DECLARE @mahang NVARCHAR(10)
DECLARE @soluong INT
DECLARE contro CURSOR FOR SELECT inserted.mahang,
inserted.soluongdeleted.soluong AS soluong FROM
inserted INNER JOIN deleted ON inserted.stt=deleted.stt
OPEN contro
FETCH NEXT FROM contro INTO @mahang,@soluong
WHILE @@FETCH_STATUS=0
BEGIN
UPDATE mathang SET soluong=soluong@soluong
WHERE mahang=@mahang
FETCH NEXT FROM contro INTO @mahang,@soluong
END
CLOSE contro
DEALLOCATE contro
END
END
```

Tài liệu tham khảo:

[1]. Giáo trình SQL Server – Đỗ Ngọc Sơn, Phan Văn Viên - Tài liệu lưu hành nội bộ của Trường Đại học Công nghiệp Hà Nội, 2015.

[2]. Giáo trình hệ quản trị cơ sở dữ liệu - Đỗ Ngọc Sơn; Phan Văn Viên; Nguyễn Phương Nga - NXB Khoa học Kỹ thuật

[3]. Bài tập Hệ quản trị Cơ sở dữ liệu – Phạm Văn Hà, Trần Thanh Hùng, Đỗ Ngọc Sơn, Nguyễn Thị Thanh Huyền – Trường Đại học Công nghiệp Hà Nội, 2020.