Bài 5:
HÀM NGƯỜI DÙNG ĐỊNH NGHĨA & VIEW & TRIGGER

Hệ thống bài cũ

- Các nội dung đã học trong bài trước
  - Stored Procedure
  - Giao dịch

Mục tiêu bài học

1. Hàm người dùng định nghĩa

2. View

3. Trigger

HÀM NGƯỜI DÙNG ĐỊNH NGHĨA

Hàm người dùng tự định nghĩa

- Hàm người dùng tự định nghĩa
  - Là một đối tượng CSDL chứa các câu lệnh SQL, được biên dịch sẵn và lưu trữ trong CSDL, **thực hiện một hành động như các tính toán phức tạp và trả về kết quả là một giá trị.**
  - Giá trị trả về có thể là
    - Giá trị vô hướng
    - Một bảng

Hàm người dùng tự định nghĩa

- Hàm người dùng tự định nghĩa
  - Tương tự như Stored Procedure.
    - Là một đối tượng CSDL chứa các câu lệnh SQL, được biên dịch sẵn và lưu trữ trong CSDL.
  - Khác với Stored Procedure.
    - Các hàm luôn phải trả về một giá trị, sử dụng câu lệnh RETURN
    - Hàm không có tham số đầu ra
    - Không được chứa các câu lệnh INSERT, UPDATE, DELETE một bảng hoặc view đang tồn tại trong CSDL
    - Có thể tạo bảng, bảng tạm, biến bảng và thực hiện các câu lệnh INSERT, UPDATE, DELETE trên các bảng, bảng tạm, biến bảng vừa tạo trong thân hàm

Một ví dụ về hàm người dùng định nghĩa

Một ví dụ về hàm trả về giá trị vô hướng

```
CREATE FUNCTION fnVendorID
        (@VendorName varchar(50))
  RETURNS int
BEGIN
  RETURN (SELECT VendorID FROM Vendors WHERE VendorName = @VendorName)
END
```

Câu lệnh gọi hàm

```
SELECT InvoiceDate, InvoiceTotal
FROM Invoices
WHERE VendorID = dbo.fnVendorID('IBM')
```

Results | Messages

	InvoiceDate	InvoiceTotal
1	2008-05-07 00:00:00	116.54
2	2008-06-09 00:00:00	1083.58

Các loại hàm người dùng tự định nghĩa

Kiểu hàm	Mô tả
Hàm giá trị vô hướng	Trả về giá trị đơn của mọi kiểu dữ liệu T-SQL.
Hàm giá trị bảng đơn giản	Trả về bảng, là kết quả của một câu lệnh SELECT đơn.
Hàm giá trị bảng nhiều câu lệnh	Trả về bảng, là kết quả của nhiều câu lệnh.

Tạo hàm giá trị vô hướng

- **Cú pháp tạo hàm giá trị vô hướng**

```
CREATE FUNCTION [<tên schema>.] <tên hàm>
([@<tên tham số> <kiểu dữ liệu> [= <Giá trị mặc định>]] [, ...])
RETURNS <kiểu dữ liệu>
[WITH [ENCRYPTION] [, SCHEMABINDING] [, <Mệnh đề EXECUTE AS>]]
[AS]
BEGIN
   [<Câu lệnh SQL>]
   RETURN <Biểu thức vô hướng>
END
```

- Chú ý:
  - Câu lệnh gọi hàm:
    - Không thể truyền tham số theo tên
    - Truyền đầy đủ các tham số theo vị trí. Kể cả tham số tùy chọn, nếu muốn sử dụng giá trị mặc định, phải đặt từ khóa DEFAULT tại đúng vị trí tham số tùy chọn đó.

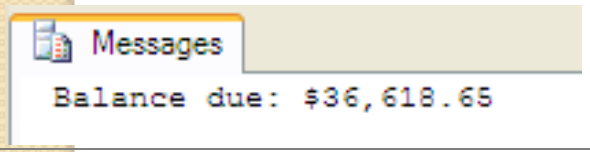
Ví dụ về hàm giá trị vô hướng

Câu lệnh tạo hàm giá trị vô hướng trả về tổng số tiền đáo hạn của các hóa đơn

```
CREATE FUNCTION fnBalanceDue()
   RETURNS money
BEGIN
   RETURN (SELECT SUM(InvoiceTotal - PaymentTotal - CreditTotal)
                  FROM  Invoices
                  WHERE InvoiceTotal - PaymentTotal -
                        CreditTotal > 0)
END
```

Câu lệnh tạo hàm giá trị vô hướng trả về tổng số tiền đáo hạn của các hóa đơn

```
PRINT 'Balance due: $' +      CONVERT(varchar,dbo.fnBalanceDue(),1)
```



Hàm giá trị bảng đơn giản

- Cú pháp câu lệnh tạo hàm giá trị bảng đơn giản

CREATE FUNCTION [<tên schema>.] <tên hàm>

([@<tên tham số> <kiểu dữ liệu> [= <Giá trị mặc định>]] [, ...])

RETURNS TABLE

[WITH {ENCRYPTION|SCHEMABINDING|ENCRYPTION, SCHEMABINDING}]

[AS]

RETURN [(] <Câu lệnh SELECT> [)]

- Chú ý

- Hàm giá trị bảng đơn giản còn gọi là hàm giá trị bảng nội tuyến
- Hàm giá trị bảng đơn giản có thể được dùng trong câu lệnh truy vấn thay thế cho tên bảng hoặc tên view.

Demo
Hàm giá trị bảng đơn giản

Câu lệnh tạo hàm giá trị bảng đơn giản

```
CREATE   FUNCTION   fnTopVendorsDue
                        (@CutOff money = 0)
   RETURNS  table
RETURN
   (SELECT  VendorName, SUM(InvoiceTotal) AS TotalDue
      FROM   Vendors JOIN Invoices ON Vendors.VendorID =   Invoices.VendorID
    WHERE    InvoiceTotal - CreditTotal - PaymentTotal > 0
    GROUP    BY VendorName
    HAVING SUM(InvoiceTotal) >= @CutOff)
```

Câu lệnh gọi hàm

```
SELECT * FROM dbo.fnTopVendorsDue(5000)
```

Câu lệnh SELECT sử dụng hàm trong phép kết nối

```
SELECT   Vendors.VendorName, VendorCity, TotalDue
   FROM  Vendors JOIN dbo.fnTopVendorsDue(DEFAULT)    AS TopVendors
       ON Vendors.VendorName = TopVendors.VendorName
```

Hàm giá trị bảng đa câu lệnh

- Cú pháp:

```
CREATE FUNCTION [<tên schema>].<tên hàm>
    ([@<tên tham số> <tên kiểu dữ liệu> [= <Giá trị mặc định>]] [,...])
    RETURNS @<tên biến trả về> TABLE
    (<tên cột 1> <kiểu dữ liệu> [<Các thuộc tính cột>]
    [, <tên cột 1> <kiểu dữ liệu> [<Các thuộc tính cột>]]...)
    [WITH [ENCRYPTION] [, SCHEMABINDING] [, <mệnh đề EXECUTE AS>]]
    [AS]
BEGIN
<Các câu lệnh SQL>
RETURN
END
```

Demo
Hàm giá trị bảng đa câu lệnh

Ví dụ câu lệnh tạo hàm giá trị bảng đa câu lệnh

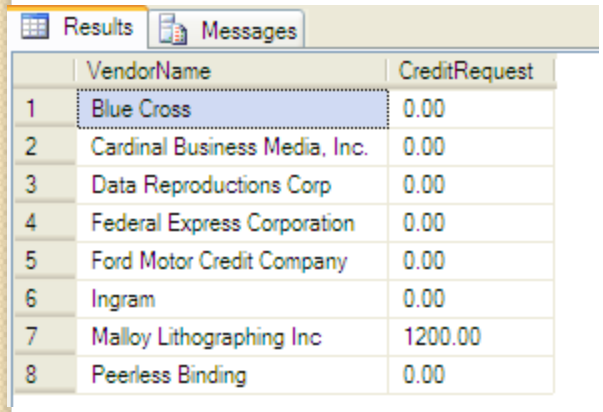
```
CREATE  FUNCTION    fnCreditAdj    (@HowMuch money)
   RETURNS     @OutTable          table
                          (InvoiceID int, VendorID int, InvoiceNumber varchar(50),
                           InvoiceDate smalldatetime, InvoiceTotal money,
                          PaymentTotal money, CreditTotal money)
BEGIN
   INSERT  @OutTable
      SELECT   InvoiceID, VendorID, InvoiceNumber, InvoiceDate,
                    InvoiceTotal, PaymentTotal, CreditTotal
        FROM    Invoices
       WHERE   InvoiceTotal - CreditTotal - PaymentTotal > 0

WHILE (SELECT  SUM(InvoiceTotal - CreditTotal - PaymentTotal)   FROM  @OutTable) >= @HowMuch
      BEGIN
         UPDATE @OutTable
         SET CreditTotal = CreditTotal + .01
         WHERE InvoiceTotal - CreditTotal - PaymentTotal > 0
      END
      RETURN
END
```

Demo
Hàm giá trị bảng đa câu lệnh

Câu lệnh SELECT sử dụng hàm

```
SELECT   VendorName, SUM(CreditTotal) AS CreditRequest
  FROM   Vendors JOIN dbo.fnCreditAdj(50000) AS CreditTable
    ON   Vendors.VendorID = CreditTable.VendorID
 GROUP   BY VendorName
```


Results Messages

	VendorName	CreditRequest
1	Blue Cross	0.00
2	Cardinal Business Media, Inc.	0.00
3	Data Reproductions Corp	0.00
4	Federal Express Corporation	0.00
5	Ford Motor Credit Company	0.00
6	Ingram	0.00
7	Malloy Lithographing Inc	1200.00
8	Peerless Binding	0.00

Xóa và chỉnh sửa hàm

- **Cú pháp của câu lệnh DROP FUNCTION**

  DROP FUNCTION [<tên schema>.] <tên hàm> [, ...]

- **Cú pháp của câu lệnh ALTER FUNCTION cho hàm giá trị vô hướng**
  - Cú pháp tương tự câu lệnh tạo hàm. Thay từ khóa CREATE bởi từ khóa ALTER

VIEW

Khung nhìn - View

- View là một bảng ảo (virtual table) được tạo ra để cho phép người dùng truy cập đến các cột được chỉ định của một bảng.
- Thực chất VIEW là một câu lệnh truy vấn được biên dịch sẵn và lưu trữ như là một đối tượng trong CSDL.
- View có thể bao gồm dữ liệu từ nhiều cột của các bảng khác nhau. Các bảng này được gọi là bảng cơ sở

Lợi ích của View

- Một số lợi ích khi sử dụng View:
  - Che dấu và bảo mật dữ liệu
    - Không cho phép người dùng xem toàn bộ dữ liệu chứa trong các bảng
    - Bằng cách chỉ định các cột trong View, các dữ liệu quan trọng chứa trong một số cột của bảng có thể được che dấu.
  - Hiển thị dữ liệu một cách tùy biến
    - Với mỗi người dùng khác nhau, có thể tạo các View khác nhau phù hợp với nhu cầu xem thông tin của từng người dùng.

Lợi ích của View

- Một số lợi ích khi sử dụng View:
  - Lưu trữ câu lệnh truy vấn phức tạp và thường xuyên sử dụng.
  - Thực thi nhanh hơn các câu lệnh truy vấn do đã được biên dịch sẵn.
  - Đảm bảo tính toàn vẹn dữ liệu
    - Khi sử dụng View để cập nhật dữ liệu trong các bảng cơ sở, SQL Server sẽ tự động kiểm tra các ràng buộc toàn vẹn trên các bảng.

Tạo View

- **Cú pháp của câu lệnh CREATE VIEW**

CREATE VIEW <tên view> [(<tên cột 1> [, <tên cột 2>]...)]

AS

<Câu lệnh SELECT>

Tạo View

- Chú ý:
  - Tên view không được trùng với tên bảng hoặc view đã tồn tại
  - Câu lệnh SELECT tạo VIEW
    - Không được chứa mệnh đề INTO, hoặc ORDER BY trừ khi chứa từ khóa TOP
  - Đặt tên cột
    - Cột chứa giá trị được tính toán từ nhiều cột khác phải được đặt tên
    - Nếu cột không được đặt tên, tên cột sẽ được mặc định giống tên cột của bảng cơ sở.

Ví dụ về View

- Ví dụ 1

```
CREATE   VIEW  VendorInvoices
AS
SELECT          VendorName, InvoiceNumber, InvoiceDate,
                InvoiceTotal
FROM            Vendors JOIN Invoices ON Vendors.VendorID =
                Invoices.VendorID
```

- Ví dụ 2: **Câu lệnh CREATE VIEW sử dụng mệnh đề TOP và ORDER BY**

```
CREATE   VIEW   TopVendors
AS
SELECT          TOP 5   PERCENT Invoices.VendorID,
InvoiceTotal
FROM            Invoices JOIN Vendors ON Vendors.VendorId =
                Invoices.VendorId
ORDER BY        InvoiceTotal DESC
```

Ví dụ về View

- Ví dụ 3: **Câu lệnh đặt tên toàn bộ cột view trong mệnh đề CREATE VIEW**

```
CREATE  VIEW      OutstandingInvoices (InvoiceNumber,
                                       InvoiceDate, InvoiceTotal, BalanceDue)
AS
SELECT  InvoiceNumber,     InvoiceDate,         InvoiceTotal,
          InvoiceTotal - PaymentTotal - CreditTotal
FROM    Invoices
WHERE   InvoiceTotal - PaymentTotal - CreditTotal > 0
```

- **Ví dụ 4: Câu lệnh đặt tên chỉ mình cột được tính toán trong mệnh đề SELECT**

```
CREATE  VIEW      OutstandingInvoices
AS
SELECT  InvoiceNumber,     InvoiceDate,         InvoiceTotal,
          InvoiceTotal - PaymentTotal – CreditTotal      AS      BalanceDue
FROM    Invoices
WHERE   InvoiceTotal - PaymentTotal - CreditTotal > 0
```

Hai loại View

- Hai loại VIEW:
  - VIEW chỉ đọc (read-only view)
    - View này chỉ dùng để xem dữ liệu
  - VIEW có thể cập nhật (updatable view)
    - Xem dữ liệu
    - Có thể sử dụng câu lệnh INSERT, UPDATE, DELETE để cập nhật dữ liệu trong các bảng cơ sở qua View

View có thể cập nhật

- **Các yêu cầu để tạo view có thể cập nhật**
  - Câu lệnh SELECT không được chứa
    - Mệnh đề DISTINCT hoặc TOP.
    - Một hàm kết tập (Aggregate function)
    - Một giá trị được tính toán.
    - Mệnh đề GROUP BY và HAVING.
    - Toán tử UNION.
- Nếu câu lệnh tạo View vi phạm một trong số điều kiện trên. VIEW được tạo ra là VIEW chỉ đọc.

Ví dụ View có thể cập nhật

- **Câu lệnh CREATE VIEW tạo view có thể cập nhật**

```
CREATE  VIEW      InvoiceCredit
AS
SELECT  InvoiceNumber, InvoiceDate, InvoiceTotal,    PaymentTotal, CreditTotal
FROM    Invoices
WHERE   InvoiceTotal - PaymentTotal - CreditTotal > 0
```

- **Câu lệnh UPDATE cập nhật view**

```
UPDATE  InvoiceCredit
SET     CreditTotal    =  CreditTotal + 200
WHERE   InvoiceTotal  –  PaymentTotal – CreditTotal >= 200
```

- **Câu lệnh CREATE VIEW tạo view chỉ đọc**

```
CREATE VIEW OutstandingInvoices
AS
SELECT InvoiceNumber, InvoiceDate, InvoiceTotal,
   InvoiceTotal - PaymentTotal - CreditTotal
   AS BalanceDue
FROM Invoices
WHERE InvoiceTotal - PaymentTotal - CreditTotal > 0
```

Xóa và chỉnh sửa View

- Cú pháp câu lệnh xóa View

DROP VIEW <tên View>

- Cú pháp câu lệnh chỉnh sửa View

ALTER VIEW <tên View> [(<tên cột 1> [,<tên cột 2>]...)]

[WITH {ENCRYPTION|SCHEMABINDING|ENCRYPTION,SCHEMABINDING}]

AS <câu lệnh SELECT>

[WITH CHECK OPTION]

Ví dụ
Xóa và chỉnh sửa View

- **Câu lệnh tạo view**

```
CREATE  VIEW  Vendors_SW
AS
SELECT *  FROM  Vendors
WHERE     VendorState IN ('CA','AZ','NV','NM')
```

- **Câu lệnh chỉnh sửa view**

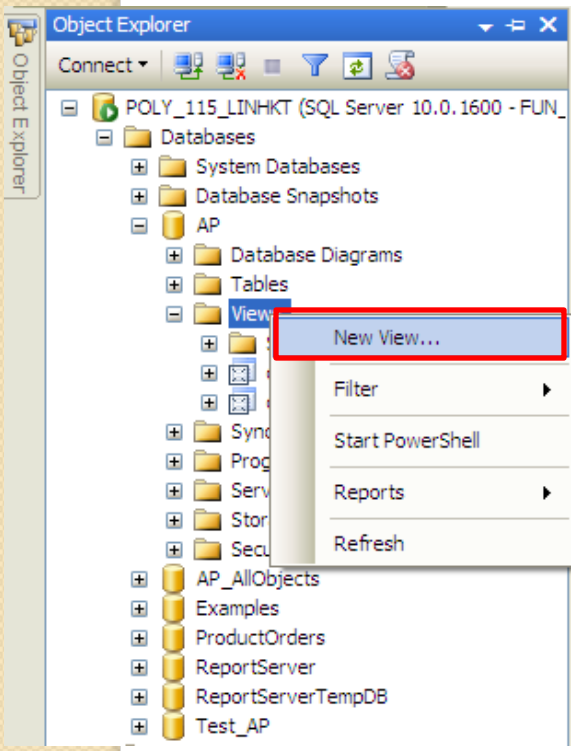
```
ALTER   VIEW     Vendors_SW
AS
SELECT *
FROM    Vendors
WHERE  VendorState  IN ('CA','AZ','NV','NM','UT','CO')
```

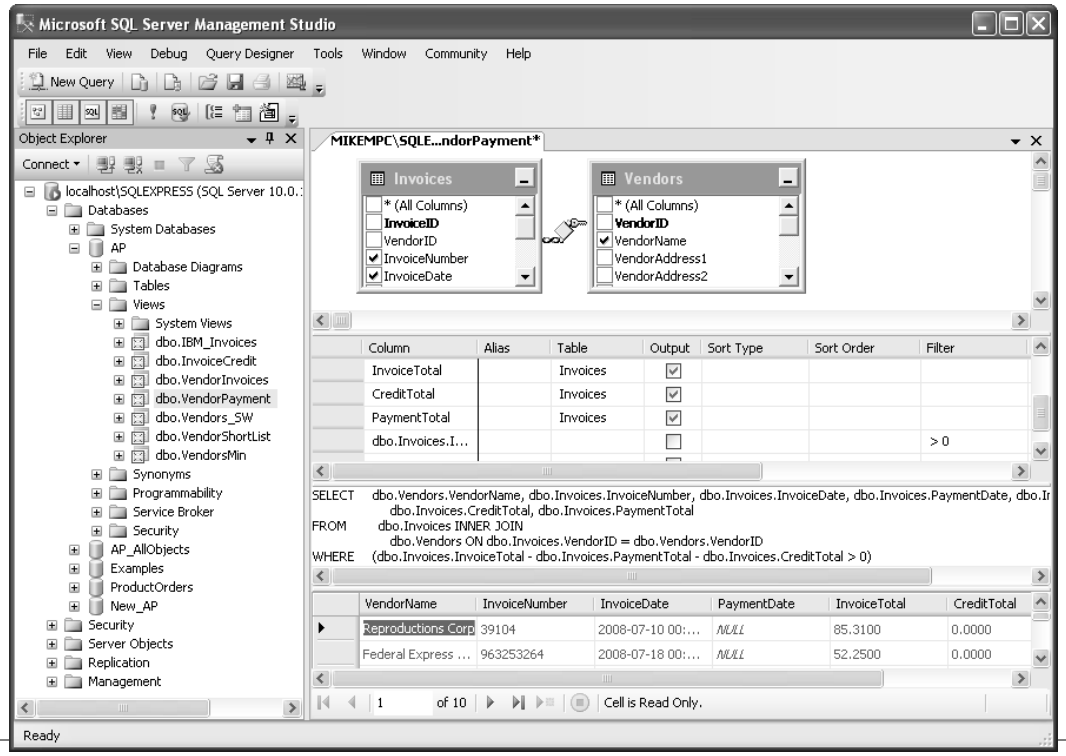
- **Câu lệnh xóa view**

```
DROP   VIEW     Vendors_SW
```

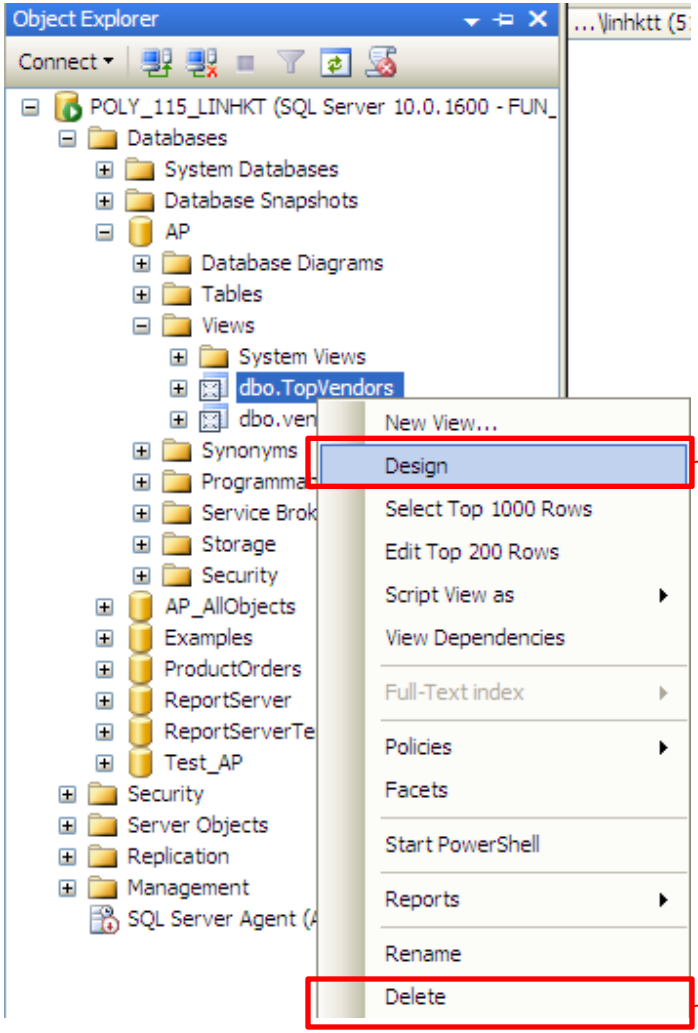
Hướng dẫn sử dụng View Designer
Tạo View

- Management Studio cung cấp công cụ View Designer để làm việc với View
- Cách sử dụng công cụ này tương tự như Query Designer





Hướng dẫn sử dụng View Designer
Xóa và chỉnh sửa View



Chỉnh sửa View

Xóa View

TRIGGER

- Là một đối tượng chứa tập câu lệnh SQL, được thực thi tự động khi có thay đổi CSDL (SP được thực hiện khi có lời gọi).
- Mỗi trigger được tạo và gắn liền với một bảng
- Sử dụng trigger hiệu quả, hiệu năng của CSDL:
  - Trigger nhận biết và ngăn chặn, hủy bỏ những thao tác làm thay đổi trái phép dữ liệu trong CSDL
  - Thông qua trigger, có thể tạo và kiểm tra những mối quan hệ phức tạp hơn giữa các bảng trong CSDL mà bản thân các ràng buộc không thực hiện được

TRIGGER

- Cú pháp

```
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
```

TRIGGER

- Ví dụ:

```
Create Trigger trg_tblMH_Insert
on tblMuahang
For Insert
As
   Update tblSach
   Set tblSach.Soluong = tblSach.Soluong
                - inserted.Soluong
   From tblSach inner join inserted
        on tblSach.MaSach =
inserted.MaSach
```

TRIGGER

- Ví dụ: Sửa số lượng trong tblMuahang

```
Create Trigger trg_tblMH_Update
on tblMuahang
For Update
As
    If UPDATE(soluong)
    Update tblSach
    Set tblSach.Soluong = tblSach.Soluong -
           (inserted.Soluong - deleted.Soluong)
    From (deleted inner join inserted on
           deleted.MaSach = inserted.MaSach)
           inner join tblSach on
                  tblSach.MaSach = deleted.MaSach
```

TRIGGER

- Ví dụ: Sửa số lượng trong tblMuahang

```
Update tblMuaHang
Set Soluong = 2
Where MaKh = 'KH03' and MaSach='KT01'
--Trường hợp Update chỉ có MaKH
Update tblMuaHang
Set Soluong = 2
Where MaKh = 'KH03'
```

TRIGGER

- Ví dụ: Sửa số lượng trong tblMuahang

```
Update tblMuaHang
Set Soluong = 2
Where MaKh = 'KH03' and MaSach='KT01'
--Trường hợp Update chỉ có MaKH
Update tblMuaHang
Set Soluong = 2
Where MaKh = 'KH03'
```

TRIGGER

- Ví dụ:

```
Insert into tblMuaHang
Values('KH03','KT01','02/02/2013',2)
```

- SQL định nghĩa hai bảng Logic Inserted và Deleted để sử dụng trong các trigger. Cấu trúc tương tự cấu trúc của bảng mà trigger tác động
  - Khi câu lệnh Delete được thực thi trên bảng, các dòng dữ liệu bị xóa sao chép vào bảng Deleted. Bảng Inserted không có dữ liệu. Tương tự với trường hợp Insert.
  - Khi câu lệnh Update được thực thi trên bảng, các dòng dữ liệu chịu tác động của câu lệnh sẽ được sao chép vào bảng Deleted, còn trong bảng Inserted sẽ là các dòng sau khi cập nhật

TRIGGER

```
Create Trigger trg_tblMuahang_Insert1
On tblMuahang
For Insert
As
      Declare @slBan int
      Declare @slHienco int
      Declare @masach Char(10)
      Select @masach = MaSach, @slBan = Soluong
      From inserted

      Select @slHienco = Soluong
      From tblSach Where MaSach = @masach

      if @slHienco<@slBan
                Rollback Tran
      Else
                --Ban
                Update tblSach
                Set Soluong = Soluong - @slBan
                Where MaSach = @masach
```