THỦ TỤC LƯU TRỮ, HÀM VÀ TRIGGER

Tài liệu: Trang 109-132

Nội dung

- Lập trình PL/SQL
- Thủ tục / Procedure
- Hàm / Function
- Trigger

PL / SQL

Giới thiệu

- **PL/SQL** is a **p**rocedural **l**anguage extension to

  Structured **Q**uery **L**anguage

- Là một ngôn ngữ lập trình hướng thủ tục, mở rộng cho SQL
- Có thể sử dụng các biến như trong ngôn ngữ lập trình
- Các cấu trúc điều khiển (IF, WHILE, …)
- Có thể nhận các tham số truyền vào cũng như có thể trả về các giá trị thông qua các tham số (như trong các ngôn ngữ lập trình)

Khối lệnh

```
DECLARE

-- khai báo biến

BEGIN

-- các lệnh điều khiển

END;
```

Biến và kiểu

- Biến phải được xác định kiểu và khai báo tại declare
- Có thể gán giá trị ban đầu khi khai báo bằng =
- Bắt đầu bằng @ và tuân theo quy tắc đặt tên biến trong lập trình
- Lệnh gán giá trị trong lập trình là set @bien =

Biến và kiểu

```
DECLARE
   @So_sv INT;
BEGIN
   USE QLDiem;
   SELECT @So_sv=Count(*) from sinhvien
   print @So_sv
END
```

Cấu trúc lệnh điều khiển: IF

```
IF điều kiện
    BEGIN
        các công việc / câu lệnh ;
    END;
    [ELSIF điều kiện
        BEGIN
                các công việc / câu lệnh ;
        END;]
    [ELSE
        BEGIN
                các công việc / câu lệnh ;
        END;]
```

Cấu trúc lệnh điều khiển: IF

```
DECLARE
    @So_sv INT;
BEGIN
    USE QLDiem;
    SELECT @So_sv=Count(*) from sinhvien
    IF @So_sv<30
        Begin
            print N'Không đủ mở lớp’
        End
    Else
        if @So_sv<60
            Begin
                print N'Đủ mở lớp’
            End
        Else
            Begin
                print N'Cần chia nhóm’
            End
END
```

Cấu trúc lệnh điều khiển: WHILE

```
WHILE điều kiện
    BEGIN
        các công việc / câu lệnh ;
    END;
```

```
DECLARE
@so int = 10
BEGIN
WHILE (@so>0 )
    begin
        print @so;
        set @so  = @so-1;
    end;
END;
```

PROCEDURE

FUNCTION

TRIGGER

Công dụng

- Đặt tên, lưu trữ lại các câu lệnh PL/SQL lên server
- Tăng tốc xử lý cho các lần gọi thực thi vì đã được biên dịch
- Dễ bảo trì trong quá trình sử dụng
- Tách rời lập trình xử lý dữ liệu và thiết kế giao diện theo mô hình MVC

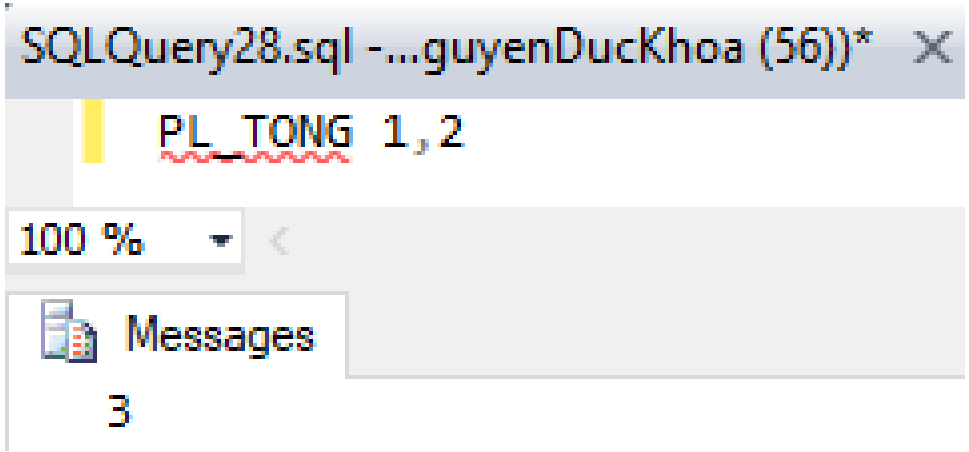
Procedure

```
CREATE|ALTER PROC tên thủ tục
        (danh sách tham số vào và kiểu)
AS
[DECLARE ……]
BEGIN
        các công việc / câu lệnh
END

EXEC tên thủ tục
```

Procedure

Tham số vào

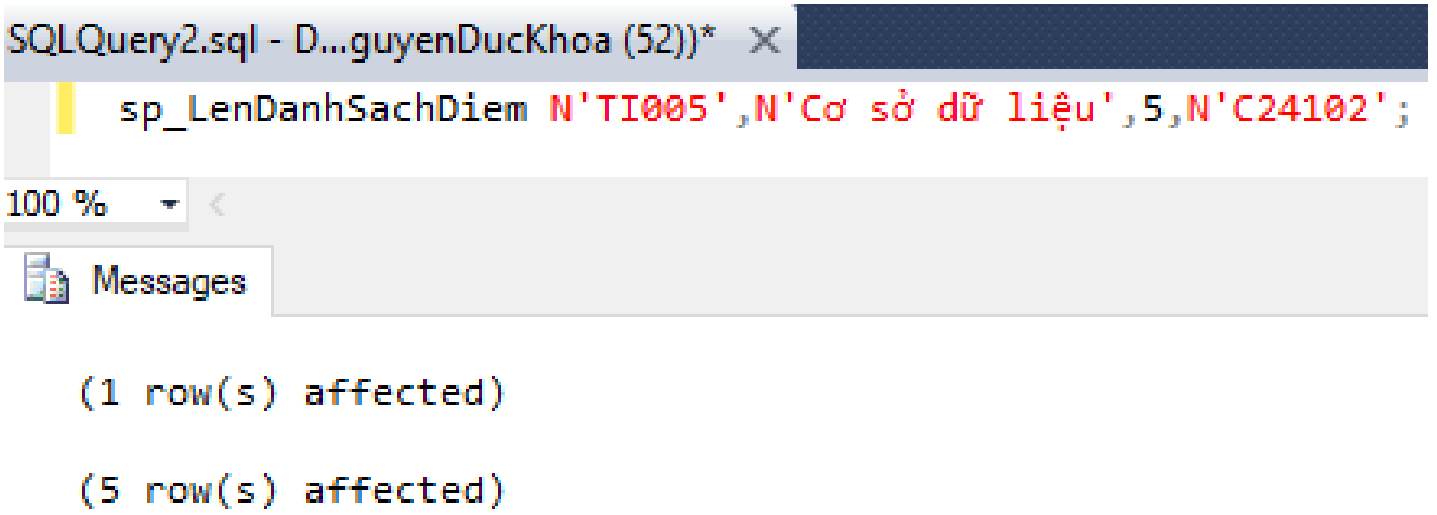
```
CREATE PROC PL_TONG(@bien1 int, @bien2 int)
AS
DECLARE
   @tong INT;
BEGIN
   set @tong=@bien1+@bien2;
   print @tong;
END
--PL_TONG 1,2
```



Procedure

Tham số vào

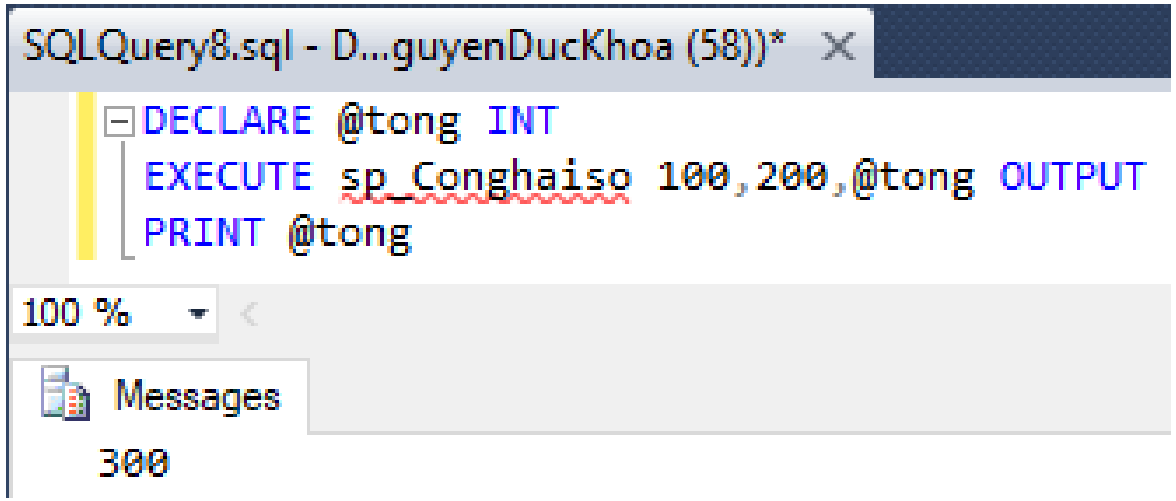
```
CREATE PROC sp_LenDanhSachDiem(@mamonhoc NVARCHAR(10),
    @tenmonhoc NVARCHAR(50),@sodvht SMALLINT, @malop NVARCHAR(10))
AS
BEGIN
    INSERT INTO monhoc VALUES(@mamonhoc,@tenmonhoc,@sodvht);
    INSERT INTO diemthi(mamonhoc,masv) SELECT @mamonhoc,masv FROM sinhvien
                                        WHERE malop=@malop
END
--sp_LenDanhSachDiem N'TI005',N'Cơ sở dữ liệu',5,N'C24102';
```



Procedure

Tham số vào và ra

```
CREATE PROCEDURE sp_Conghaiso(@a INT, @b INT, @c INT OUTPUT)
AS SET @c=@a+@b

--DECLARE @tong INT
--EXECUTE sp_Conghaiso 100,200,@tong OUTPUT
--PRINT @tong
```



Procedure

Tham số với giá trị mặc định

```
CREATE PROC sp_TestDefault(@tenlop NVARCHAR(30)=NULL, @noisinh NVARCHAR(100)=N'Huế')
AS
DECLARE
    @ns nvarchar(100) ='%'+@noisinh+'%’,
    @tl nvarchar(100) ='%'+@tenlop+'%';
BEGIN
    IF @tenlop IS NULL
        SELECT hodem,ten FROM sinhvien INNER JOIN lop ON
            sinhvien.malop=lop.malop WHERE noisinh LIKE @ns
    ELSE
        SELECT hodem,ten FROM sinhvien INNER JOIN lop ON
            sinhvien.malop=lop.malop WHERE noisinh LIKE @ns AND tenlop LIKE @tl
END

--4 CÁCH GỌI
--sp_testdefault
--sp_testdefault @tenlop =N'Tin K24'
--sp_testDefault @noisinh=N'Nghệ An'
--sp_testdefault @tenlop=N'Tin K24', @noisinh=N'Quảng Trị'
```

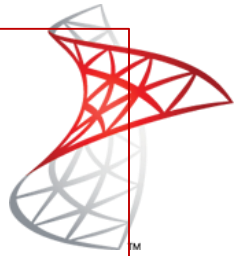
Procedure

Chỉnh sửa và xóa

```
ALTER PROCEDURE tên_thủ_tục [(danh_sách_tham_số)]
AS
BEGIN
   Các_câu_lệnh_Của_thủ_tục
END


DROP PROCEDURE tên_thủ_tục
```

Function

```
CREATE|ALTER FUNCTION tên hàm
        (danh sách tham số vào và kiểu)
RETURNS kiểu dữ liệu rả về
AS
BEGIN
[DECLARE ……]
        các công việc / câu lệnh
END

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

DROP FUNCTION tên hàm
```



FUNCTION

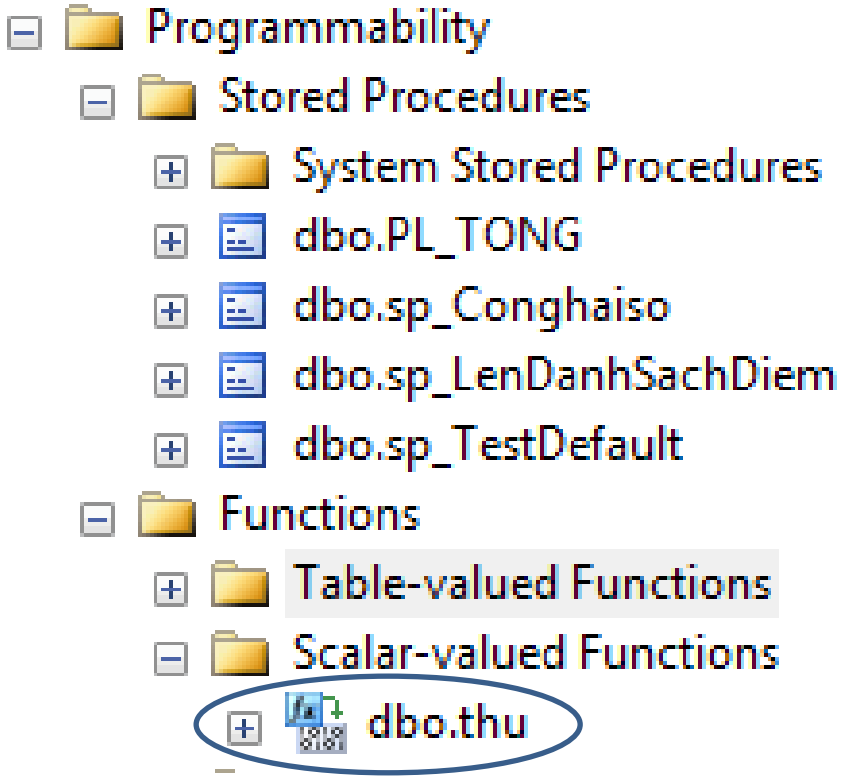
Đặc điểm

- Điểm khác biệt giữa hàm và thủ tục là hàm trả về một giá trị thông qua tên hàm còn thủ tục thì không
- Ngoài những hàm do hệ quản trị cơ sở dữ liệu cung cấp sẵn, người sử dụng có thể định nghĩa thêm các hàm nhằm phục vụ cho mục đích riêng của mình
- Có thể sử dụng hàm như là một thành phần của một biêu thức

FUNCTION

Ví dụ hàm trả về thứ

```
CREATE FUNCTION thu(@ngay DATETIME) RETURNS NVARCHAR(10)
AS
BEGIN
   DECLARE @st NVARCHAR(10)
   SELECT @st=CASE DATEPART(DW,@ngay)
                WHEN 1 THEN N'Chủ nhật'
                WHEN 2 THEN N'Thứ hai'
                WHEN 3 THEN N'Thứ ba'
                WHEN 4 THEN N'Thứ tư'
                WHEN 5 THEN N'Thứ năm'
                WHEN 6 THEN N'Thứ sáu'
                ELSE N'Thứ bảy'
              END
   RETURN (@st) /* Trị trả về của hàm */
END
```



FUNCTION

Lời gọi hàm

```
SELECT dbo.thu(getdate())

PRINT dbo.thu(getdate())

SELECT masv,hodem,ten,dbo.thu(ngaysinh) AS thu, ngaysinh FROM sinhvien WHERE malop='C24102'
```

FUNCTION

Hàm với giá trị trả về là dữ liệu kiểu bảng *(c*òn được gọi là *hàm nội tuyến* - inline function)

```
CREATE FUNCTION tên_hàm ([danh_sách_tham_số])
RETURNS TABLE
AS RETURN (câu_lệnh_select)
```

Cú pháp của hàm nội tuyến phải tuân theo các qui tắc sau:

- Kiểu trả về của hàm phải được chỉ định bởi mệnh đề RETURNS TABLE.
- Trong phần thân của hàm chỉ có duy nhất một câu lệnh RETURN xác định giá trị trả về của hàm thông qua duy nhất một câu lệnh SELECT. Ngoài ra, không sử dụng bất kỳ câu lệnh nào khác trong phần thân của hàm.

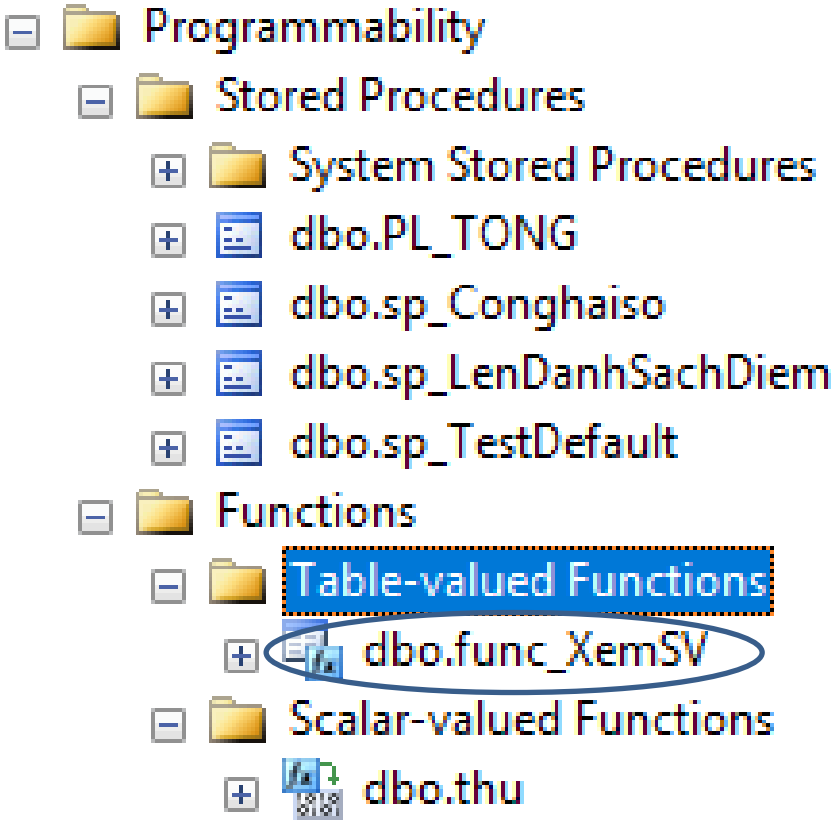
FUNCTION

Hàm với giá trị trả về là dữ liệu kiểu bảng *(còn được gọi là hàm nội tuyến - inline function)*

```
CREATE FUNCTION func_XemSV(@khoa SMALLINT)
RETURNS TABLE
AS RETURN
(SELECT masv,hodem,ten,ngaysinh
FROM sinhvien INNER JOIN lop ON
sinhvien.malop=lop.malop
WHERE khoa=@khoa)
```

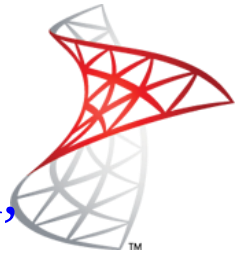
LỜI GỌI HÀM

```
SELECT * FROM func_XemSV(24)
```



FUNCTION

Hàm với giá trị trả về là dữ liệu kiểu bảng *(còn được gọi là hàm nội tuyến - inline function)*



Trong trường hợp cần phải sử dụng đến nhiều câu lệnh trong phần than của hàm, ta sử dụng cú pháp như sau để định nghĩa hàm:

```
CREATE FUNCTION tên_hàm([danh_sách_tham_số])
RETURNS @biến_bảng TABLE định_nghĩa_bảng
AS
BEGIN
    Các_câu_lệnh_trong_thân_hàm gán cho @biến_bảng
    RETURN
END
```

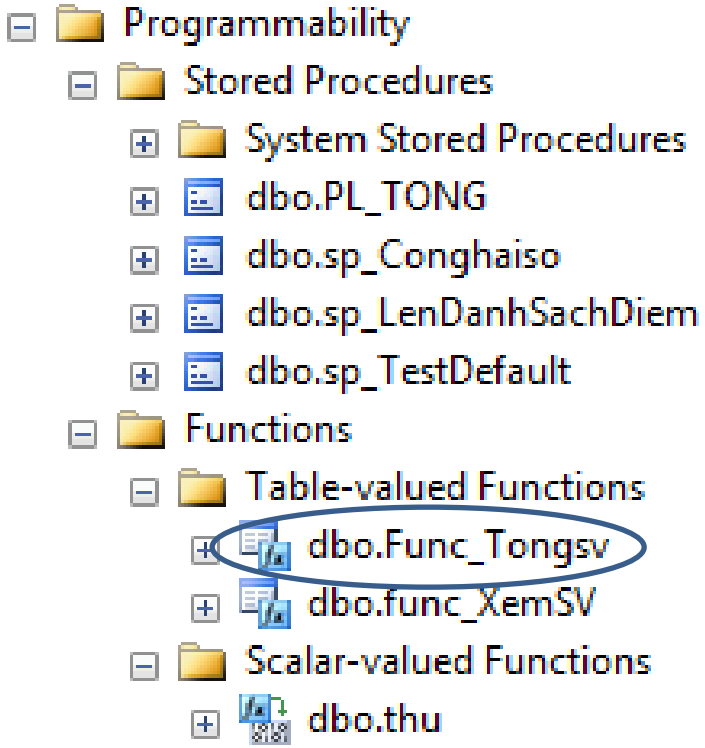
Lưu ý:

- Biến @biến_bảng trong mệnh đề RETURNS được sử dụng như một bảng.
- Câu lệnh RETURN trong thân hàm không chỉ định giá trị trả về. Giá trị trả về của hàm chính là @biếnbảng nhận giá trị gì trong mệnh đề RETURNS.

FUNCTION

Hàm với giá trị trả về là dữ liệu kiểu bảng *(còn được gọi là hàm nội tuyến - inline function)*

```
CREATE FUNCTION Func_Tongsv(@khoa SMALLINT)
RETURNS @bangthongke TABLE
    (makhoa NVARCHAR(5), tenkhoa NVARCHAR(50), tongsosv INT)
AS
BEGIN
    IF @khoa=0
        INSERT INTO @bangthongke
        SELECT khoa.makhoa,tenkhoa,COUNT(masv)
        FROM (khoa INNER JOIN lop ON khoa.makhoa=lop.makhoa) INNER
            JOIN sinhvien on lop.malop=sinhvien.malop
        GROUP BY khoa.makhoa,tenkhoa
    ELSE
        INSERT INTO @bangthongke
        SELECT khoa.makhoa,tenkhoa,COUNT(masv)
        FROM (khoa INNER JOIN lop ON khoa.makhoa=lop.makhoa)
            INNER JOIN sinhvien ON lop.malop=sinhvien.malop
        WHERE khoa=@khoa
        GROUP BY khoa.makhoa,tenkhoa
    RETURN /*Trả kết quả về cho hàm*/
END     //SELECT * FROM func_TongSV(24)
```



TRIGGER

Ý nghĩa

- Chứa các câu lệnh và được thực thi khi có lời gọi.
- Được gắn vào một bảng dữ liệu xác định.
- 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.
- Ngăn chặn các thao tác trái luật.
- Gọi các lệnh cần thiết liên quan thao tác.

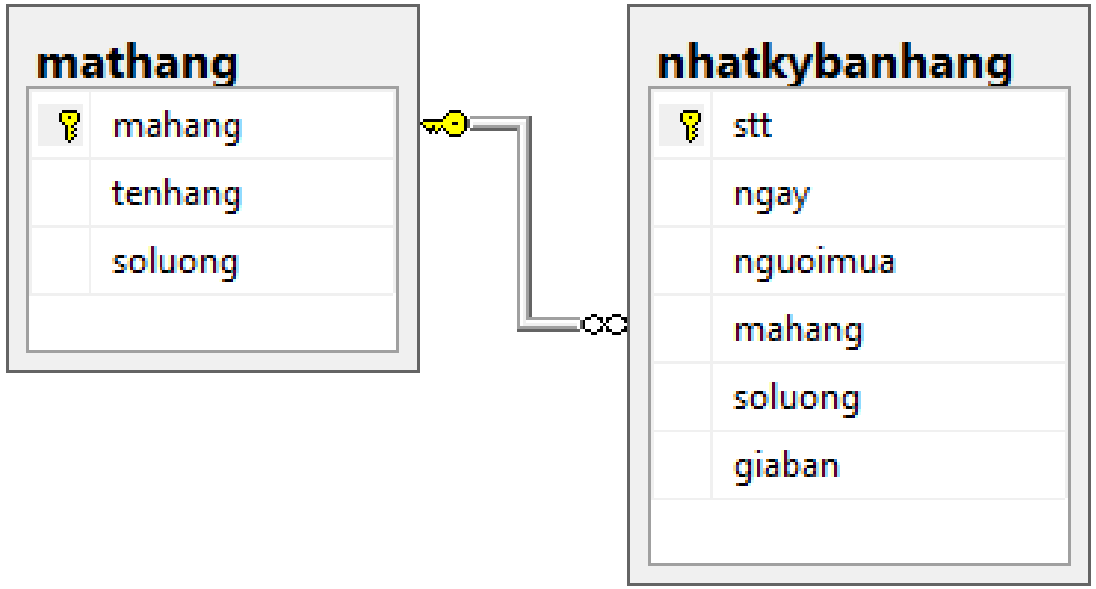
TRIGGER

Cấu trúc

```
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: FOR INSERT

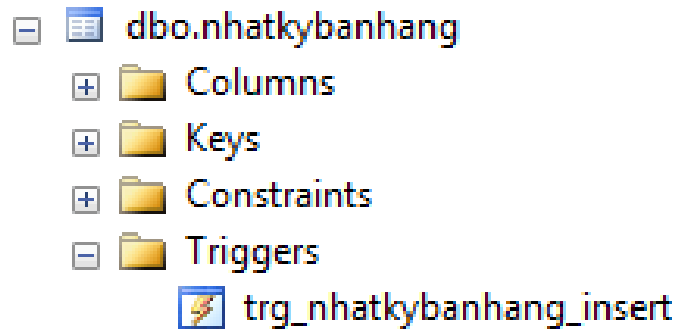


Tạo bảng MATHANG:

```
CREATE TABLE mathang
(mahang NVARCHAR(5) PRIMARY KEY,
tenhang NVARCHAR(50) NOT NULL,
soluong INT)
```

Tạo bảng NHATKYBANHANG

```
CREATE TABLE nhatkybanhang
(stt INT IDENTITY PRIMARY KEY,
ngay DATETIME, nguoimua NVARCHAR(30),
mahang NVARCHAR(5) FOREIGN KEY REFERENCES mathang(mahang),
soluong INT, giaban MONEY)
```

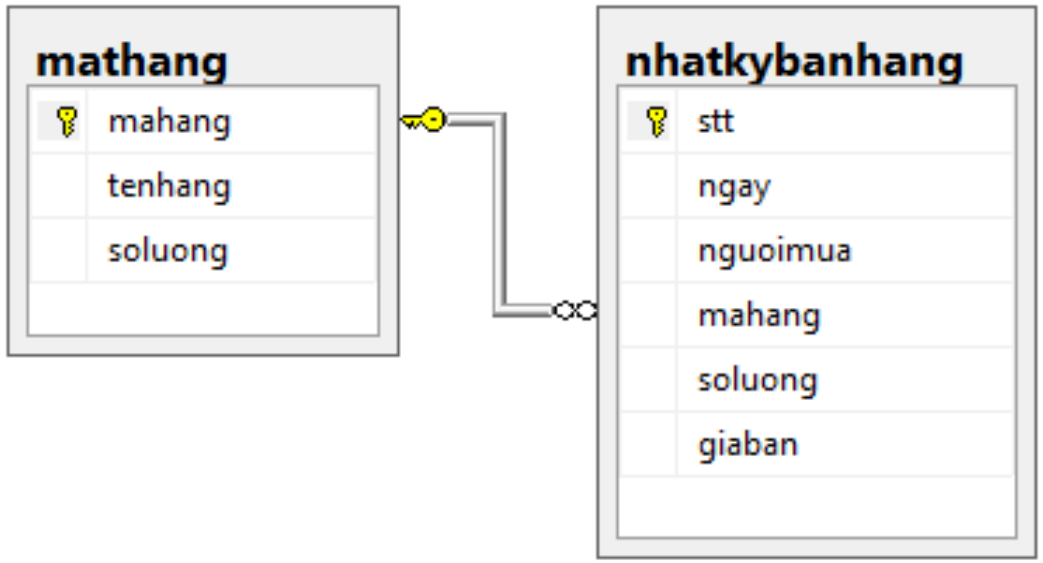


Tạo Trigger cho bảng nhatkybanhang

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

TRIGGER

Ví dụ: FOR INSERT



Thêm vào bảng mathang:

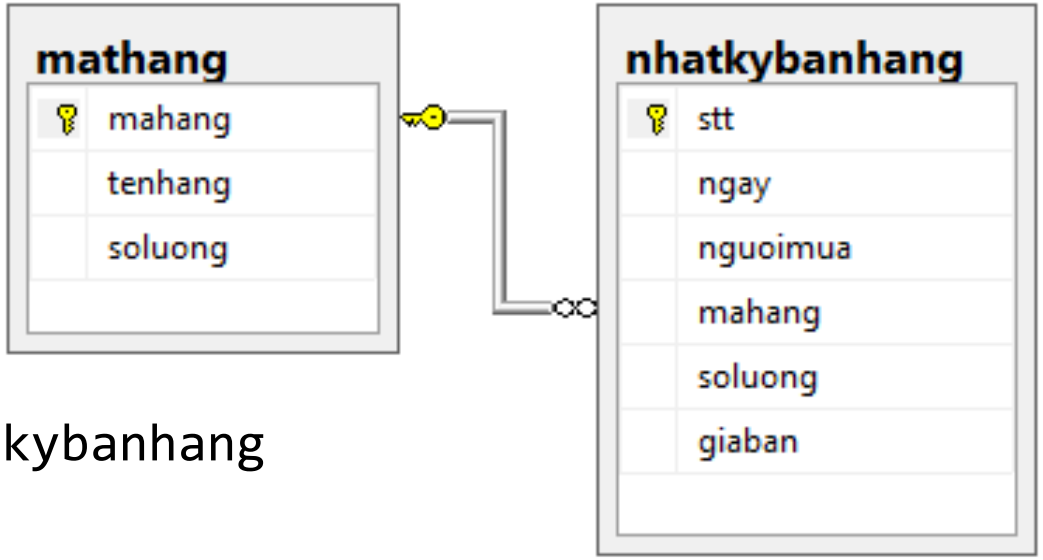
```
INSERT INTO mathang(mahang,tenhang, soluong)
VALUES(N'H1',N'Xà phòng',30)
INSERT INTO mathang(mahang,tenhang, soluong)
VALUES(N'H2',N'Kem đánh răng',45)
```

Thêm vào bảng nhatkybanhang làm trigger: trg_nhatkybanhang_insert **hoạt động:**

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

TRIGGER

Ví dụ: FOR DELETE



Tạo Trigger cho bảng nhatkybanhang:

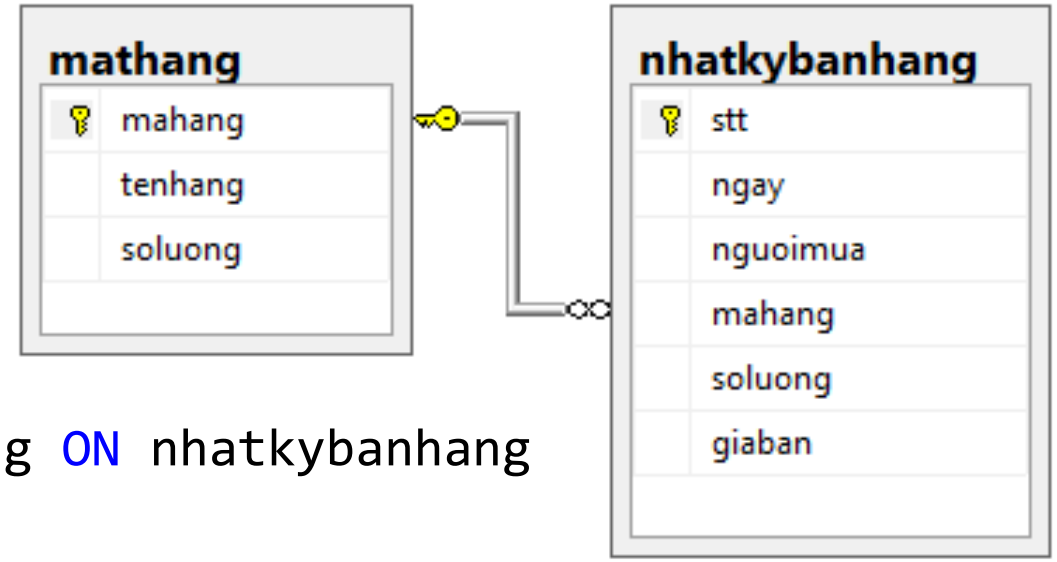
```
CREATE TRIGGER trg_nhatkybanhang_delete ON nhatkybanhang
FOR DELETE
AS
    UPDATE mathang SET mathang.soluong= mathang.soluong+deleted.soluong FROM
        mathang INNER JOIN deleted ON mathang.mahang=deleted.mahang
```

Xóa mẫu tin trong bảng nhatkybanhang làm trigger: trg_nhatkybanhang_delete **hoạt động:**

```
DELETE FROM nhatkybanhang WHERE mahang=N'H1'
```

TRIGGER

Ví dụ: FOR UPDATE và IF UPDATE(field)



Tạo Trigger:

```
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
```

Cập nhật bảng nhatkybanhang làm trigger: trg_nhatkybanhang_insert **hoạt động:**

```
UPDATE nhatkybanhang SET soluong=soluong+20 WHERE mahang=N'H1’
```

Nhận xét: có thể xảy ra soluong trong bảng mathang sẽ bị âm

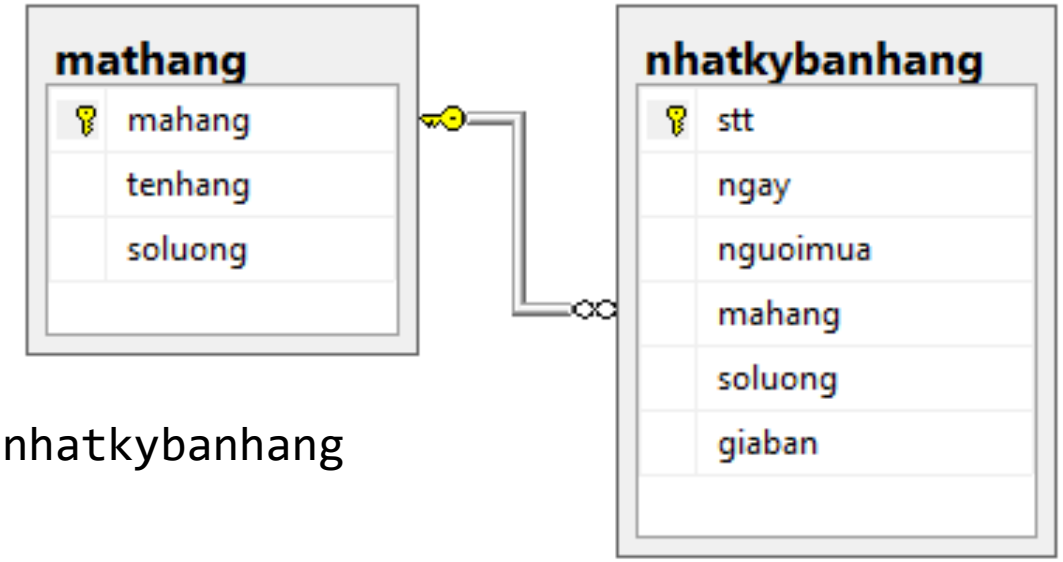
TRIGGER

ROLLBACK TRANSACTION

- 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 lệnh: ROLLBACK TRANSACTION.

TRIGGER

Ví dụ: Sửa trg_nhatkybanhang_update_soluong



Tạo Trigger:

```
CREATE TRIGGER trg_nhatkybanhang_update_soluong ON nhatkybanhang
FOR UPDATE
AS
    DECLARE @sl int
    IF UPDATE(soluong)
        SELECT @sl=mathang.soluong-(inserted.soluong-deleted.soluong)
        FROM (deleted INNER JOIN inserted ON deleted.stt = inserted.stt)
        INNER JOIN mathang ON mathang.mahang = deleted.mahang
    IF @sl<0
        ROLLBACK TRANSACTION
    ELSE
        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
```

Trigger

- Cài trigger đảm bảo điểm lần một hoặc lần 2 hoặc là null hoặc phải từ 0-10
- Cài trigger đảm bảo không cho sửa điểm lần một nếu như đã có điểm lần 2
- Cài trigger đảm bảo năm sinh của sinh viên luôn nhỏ hơn năm hiện tại
- Cài trigger tăng, giảm sỉ số sinh viên trong lớp học mỗi khi thêm mới hoặc xóa sinh viên

TRIGGER

Khi bị tác động đến nhiều mẫu tin

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

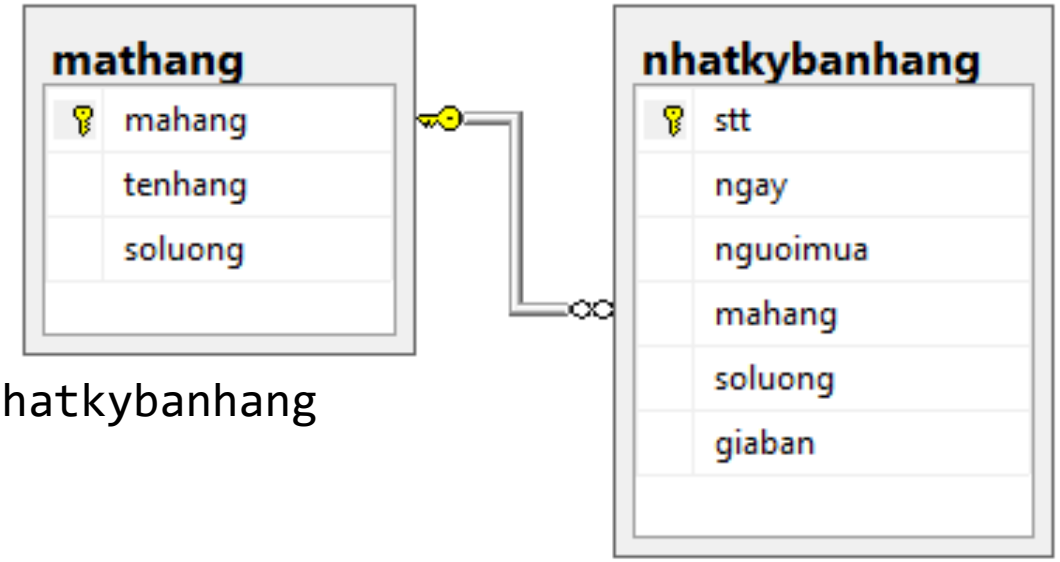
Bảng MATHANG

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

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



Xét lại Trigger đã tạo trước đây:

```
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
```

Nếu update cùng lúc nhiều mẫu tin thì Trigger bị sai:

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

TRIGGER

Sử dụng truy vấn con để khắc phục

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

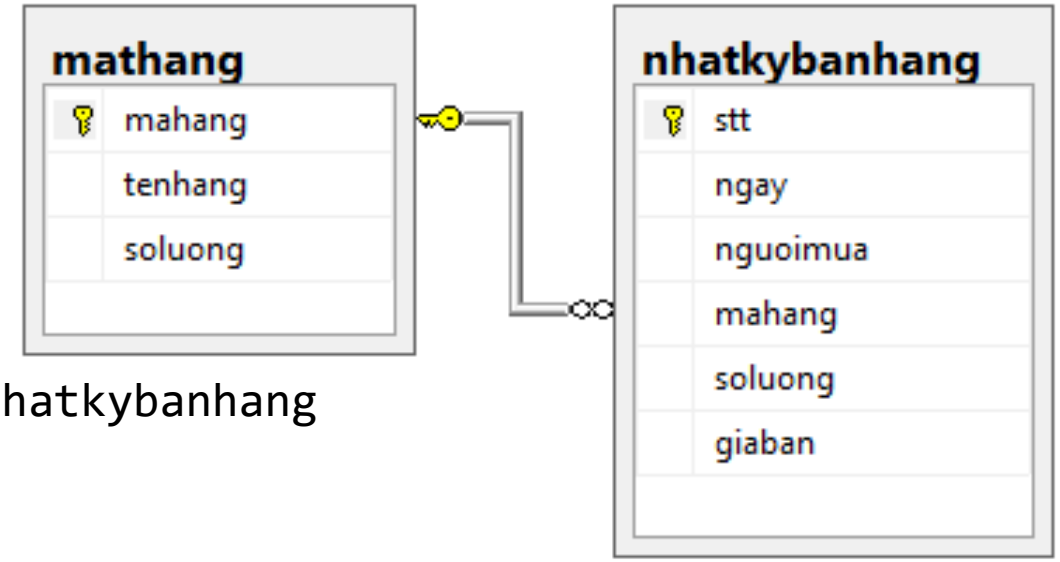
Bảng MATHANG

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

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



Lúc này Trigger phải sửa lại là:

```
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)
```

TRIGGER

Sử dụng biến con trỏ để duyệt từng dòng trong câu lệnh Select

Khai báo:

```
DECLARE tên_con_trỏ CURSOR FOR câu_lệnh_SELECT
```

Mở 1 biến con trỏ:

```
OPEN tên_con_trỏ
```

Duyệt qua biến con trỏ:

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

Chú ý:

- Khi duyệt dựa vào giá trị của biến @@FETCH_STATUS, nếu =0 thì chưa duyệt hết danh sách.
- Số lượng các biến trong danh_sách_biến phải bằng với số lượng các cột trong câu_lệnh_SELECT của DECLARE CURSOR.

TRIGGER

Ví dụ mẫu về sử dụng biến con trỏ để duyệt

```
DECLARE biến con trỏ (C) CURSOR FOR SELECT ……
OPEN (C)
DECLARE biến chứa dữ liệu lấy trên từng dòng (A)
FETCH NEXT FROM biến con trỏ INTO (A);
WHILE @@FETCH_STATUS=0
    BEGIN
        …. Lệnh xử lý
        FETCH NEXT FROM contro INTO (A);
    END;
CLOSE (C);
DEALLOCATE biến con trỏ ;
```

TRIGGER

Ví dụ về sử dụng biến con trỏ để duyệt

```
DECLARE contro CURSOR FOR SELECT mahang, tenhang, soluong FROM mathang
OPEN contro
DECLARE @mahang NVARCHAR(5)
DECLARE @tenhang NVARCHAR(50)
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

Ví dụ sửa lại trigger trên như sau

```
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.soluong-deleted.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
```

BÀI TẬP THỰC HÀNH

Thủ tục - Procedure

- Viết thủ tục bổ sung thêm một mẫu tin mới cho bảng MATHANG (thủ tục phải thực hiện kiểm tra tính hợp lệ của dữ liệu cần bổ sung: không trùng khoá chính và đảm bảo toàn vẹn tham chiếu).
- Tạo thủ tục lưu trữ có chức năng thống kê tổng số lượng hàng bán được của một mặt hàng có mã bất kỳ (mã mặt hàng cần thống kê là tham số của thủ tục).
- Viết thủ tục liệt kê chi tiết một đơn đặt hàng bất kỳ nào đó.
- Viết thủ tục tính trị giá từng đơn đặt hàng của từng khách hàng nào đó.
- Viết thủ tục cho biết số đơn đặt hàng, tổng trị giá của các đơn đặt hàng mà một nhân viên nào đó đã thực hiện.

BÀI TẬP THỰC HÀNH

Hàm - Function

- Viết hàm trả về một bảng trong đó cho biết tổng số lượng hàng bán được của mỗi mặt hàng. Sử dụng hàm này để thống kê xem tổng số lượng hàng (hiện có và đã bán) của mỗi mặt hàng là bao nhiêu.
- Viết hàm tính số mặt hàng đã cung cấp của một nhà cung cấp nào đó.
- Viết hàm cho biết một loại hàng nào đó có bao nhiêu mặt hàng.
- Viết hàm cho biết tổng giá trị của một đơn đặt hàng cụ thể nào đó.
- Viết hàm cho biết một khách hàng nào đó có bao nhiêu đơn đặt hàng.
- Viết hàm cho biết tổng giá trị các đơn đặt hàng mà nhân viên nào đó đã lập.

BÀI TẬP THỰC HÀNH

Trigger

Viết trigger cho bảng CHITIETDATHANG theo yêu cầu sau:

- Khi một mẫu mới được bổ sung vào bảng này thì giảm số lượng hàng hiện có nếu số lượng hàng hiện có lớn hơn hoặc bằng số lượng hàng được bán ra. Ngược lại thì huỷ bỏ thao tác thêm này.
- Khi cập nhật lại số lượng hàng được bán, kiểm tra số lượng hàng được cập nhật lại có phù hợp hay không (số lượng hàng bán ra không được vượt quá số lượng hàng hiện có và không được nhỏ hơn 1). Nếu dữ liệu hợp lệ thì giảm (hoặc tăng) số lượng hàng hiện có trong công ty, ngược lại thì huỷ bỏ thao tác cập nhật.
- Khi thêm hoặc cập nhật trường "giaban" thì chỉ chấp nhận giá hàng bán ra phải nhỏ hơn hoặc bằng giá gốc (giá của mặt hàng trong bảng MATHANG).