Chương 1 (tt) T-SQL –p.1-

- Biến cục bộ
- Biến hệ thống
- Các câu lệnh truy vấn dữ liệu
- Cấu trúc điều khiển
- Biến kiểu dữ liệu cursor
- Các hàm thường dùng

Biến cục bộ (tt)

- **Cú pháp:**

  DECLARE @Tên_biến kiểu_dữ_liệu[,…]

- *Ví dụ:*

  DECLARE @ten_ncc char(50), @ngayxh DATETIME

- Gán giá trị cho biến: ***SET hoặc SELECT cùng với phép gán (=)***

  (*Lệnh SET chỉ để gán giá trị cụ thể hoặc các biểu thức tính toán hoặc giá trị tính toán từ các biến khác, ngược lại lệnh **SELECT** dùng để gán các giá trị được lấy ra hoặc tính toán từ dữ liệu của các cột bên trong các bảng dữ liệu.*)

Biến cục bộ (tt)

- *Ví dụ 1: để gán giá trị ngày 25/1/2007 vào biến ngày xuất hàng*

```
DECLARE     @ngayxh DATETIME
SET         @ngayxh = '25-1-2007'
```

- *Ví dụ 2:*

```
DECLARE     @TongSLDat      int
SELECT      @TongSLDat = SUM(SLDAT)
FROM        CTHDON
```

- *Ví dụ 3:*

```
DECLARE     @MinSLDat int, @MaxSLDat int
SELECT @MinSLDat = MIN(SLDAT),
            @MaxSLDat = MAX(SLDAT),
FROM    CTHDON
```

Biến cục bộ (tt)

- **Xem giá trị hiện hành của biến**
- ***Cú pháp:***

  PRINT @Tên_biến | Biểu_thức_chuỗi

  hoặc
- *Ví dụ:*

```
DECLARE     @MinSLDat   int, @MaxSLDat   int
SELECT  @MinSLDat = MIN(SLDAT), @MaxSLDat =
                                    MAX(SLDAT),
FROM        CTHDON
PRINT       “Số lượng đặt thấp nhất là: ”
PRINT       @MinSLDat
PRINT       “Số lượng đặt cao nhất là: ”
PRINT       @MaxSLDat
```

Biến cục bộ (tt)

- Phạm vi hoạt động của biến
  - Trong Transaction-SQL phạm vi hoạt động của biến chỉ nằm trong một thủ tục nội tại (stored procedure) hoặc một lô (batch) chứa các câu lệnh mà biến đã được khai báo bên trong đó.
  - Lô được xem như một nhóm tập hợp của một hoặc nhiều câu lệnh Transaction-SQL sẽ được biên dịch đồng thời cùng lúc tại SQL Server và sau đó hệ thống sẽ thực thi các câu lệnh này ngay sau khi đã biên dịch thành công
  - Một từ khoá **GO** chỉ định kết thúc 1 lô

SELECT: không làm thay đổi CSDL, dùng để trích lọc, thống kê.
UPDATE, INSERT, DELETE làm thay đổi CSDL.

SELECT có 02 loại: thêm 1 dòng & thêm nhiều dòng.

Biến cục bộ (tt)

- Do các câu lệnh trong một lô được biên dịch tại SQL Server vì thế khi có ít nhất 1 lệnh bên trong lô có lỗi về cú pháp lúc biên dịch thì hệ thống sẽ không có lệnh nào được thực thi bên trong lô đó.

- Ví dụ:

```
SELECT * FROM     NHACC
ORDER BY TenNhaCC

INSERT INTO NHACC
('C01', 'Nguyen Van A', '87 Ly Tu Trong', '0903.123456')

SELECT * FROM     VATTU
ORDER BY Tenvtu DESC
GO
```

(thiếu từ khoá VALUES) thì các lệnh SELECT bên trong lô này không được thực hiện.

Biến cục bộ (tt)

- Đối với các lỗi khi thực hiện (run-time) bên trong 1 lô nếu trường hợp các lỗi vi phạm ràng buộc toàn vẹn dữ liệu thì hệ thống SQL Server chỉ ngưng lại câu lệnh gây lỗi và thực hiện tiếp các lệnh bên trong lô đó.
- Ví dụ:

```
SELECT * FROM     NHACC
ORDER BY TenNhaCC

INSERT INTO NHACC
VALUES ('C01', 'Nguyen Van A', '87 Ly Tu Trong', '0903.123456')

SELECT * FROM     VATTU
ORDER BY Tenvtu DESC

GO
```

(mặc dù vi phạm ràng buộc toàn vẹn trong INSERT (giả sử trùng khoá chính ở cột MaNCC) nhưng các lệnh SELECT bên trong lô này vẫn được thực hiện bình thường.

Biến cục bộ (tt)

- Ví dụ:

```
DECLARE    @NgayDH    datetime
SELECT     @NgayDH = MAX(NGAYDH)
FROM       DONDH
GO
PRINT “ngay dat hang gan nhat: ” + convert(char(12),
    @ngaydh)
GO
```

HT sẽ báo lỗi vì có thêm từ khoá GO ở giữa 2 lệnh SELECT và PRINT. Bởi vì khi đó các lệnh này được chia làm 2 lô và lô thứ hai sẽ không hiểu biến @ngaydh đã được khai báo trong lô thứ 1.

Biến hệ thống

Biến chỉ đọc & dùng (Read-only)

- Các biến hệ thống trong SQL Server luôn bắt đầu bằng 2 chữ @@ và giá trị mà chúng ta đang lưu trữ do hệ thống SQL cung cấp, người lập trình không can thiệp trực tiếp để gán giá trị vào các biến hệ thống.

Biến hệ thống (tt)

Tên biến	kiểu trả về	Dùng để trả về
connections	số nguyên	Tổng số các kết nối vào SQL Server từ khi nó được khởi động
Error	số nguyên	số mã lỗi của câu lệnh thực hiện gần nhất. Khi một lệnh thực hiện thành công thì biến này có giá trị là 0
Language	chuỗi	Tên ngôn ngữ mà hệ thống SQL đag sử dụng. Mặc định là US_English
RowCount	số nguyên	Tổng số mẩu tin được tác động vào câu lệnh truy vấn gần nhất
ServerName	chuỗi	Tên của máy tính cục bộ được cài đặt trong SQL Server
ServiceName	chuỗi	Tên dịch vụ kèm theo bên dưới SQL Server
Fetch_Status	số nguyên	Trạng thái của việc đọc dữ liệu trong bảng theo cơ chế dòng mẩu tin (cursor). Khi dữ liệu đọc mẩu tin thành công thì biến này có giá trị là 0
Version	chuỗi	Phiên bản, ngày của phẩm SQL Server và loại CPU

Biến hệ thống (tt)

- *Ví dụ:*

  SELECT * FROM NHACC

  SELECT @@ROWCOUNT

  (trả về tổng số mẩu tin đang hiện có trong bảng NHACC)

- *Ví dụ:*

  UPDATE VATTU

  SET PHANTRAM = PHANTRAM + 5

  WHERE MAVTU like “TV%”

  SELECT @@ROWCOUNT

  (Trả về tổng số mẩu tin có MAVTU bắt đầu bằng chữ “TV” trong bảng VATTU)

Các câu lệnh truy vấn dữ liệu

- *Truy vấn con*: chỉ là một câu lệnh truy vấn lựa chọn (SELECT) được lồng vào các câu lệnh truy vấn khác nhằm thực hiện các truy vấn tính toán phức tạp. Khi sử dụng đến truy vấn con chúng ta cần lưu tâm đến một vài yếu tố sau:
  - Cần mở và đóng ngoặc đơn cho câu lệnh truy vấn con.
  - Chúng ta chỉ được phép tham chiếu đến tên một cột hoặc một biểu thức sẽ trả về giá trị trong truy vấn con.
  - Kết quả truy vấn con có thể trả về là một giá trị đơn lẻ hoặc một danh sách các giá trị.
  - Cấp độ lồng nhau của các truy vấn con bên trong SQL Server là không giới hạn.

Lồng tương quan

Các câu lệnh truy vấn dữ liệu (tt)

- *Truy vấn con trả về một giá trị đơn:* là truy vấn mà kết qủa trả về của nó luôn đảm bảo chỉ là một giá trị đơn.

- Ví dụ: để biết được danh sách các đơn đặt hàng gần đây nhất.

```
SELECT      MAX(NGAYDH)
FROM        DONDH
```

Kết quả trả về: 2017-01-25 00:00:00

```
SELECT      *
FROM        DONDH
WHERE       NGAYDH = “2017-01-25”
```

- Kết hợp 2 câu truy vấn trên

```
SELECT      *
FROM        DONDH
WHERE       NGAYDH = (SELECT    MAX(NGAYDH)
                      FROM          DONDH)
```

Các câu lệnh truy vấn dữ liệu (tt)

- *Truy vấn con trả về danh sách các giá trị:* trả về của nó là danh sách các giá trị hay còn gọi là một tập hợp các phần tử. Toán tử IN sẽ được sử dụng để so sánh truy vấn con dạng này
- ***Ví dụ* 1**: để biết nhà cung cấp nào mà công ty đã đặt hàng trong tháng 01/2017.

```
SELECT    MaNCC
FROM      DONDH
WHERE     Convert(char(7), NgayDH, 21) = "2017-01"
```

  - Kết quả trả về

    MaNCC
    C03
    C01

```
SELECT    TenNCC, DienThoai
FROM      NHACC
WHERE     MaNCC IN("C01", "C03")
```

Các câu lệnh truy vấn dữ liệu (tt)

- Đâu đảm bảo rằng trong tháng 01/2017 công ty chỉ đặt hàng cho 2 nhà cung cấp C01 và C03. Do đó để luôn luôn có được danh sách họ tên các nhà cung cấp mà công ty đã đặt trong tháng 01-2017 chúng ta thực hiện truy vấn con sau:

```
SELECT      TenNCC, DienThoai
FROM        NHACC
WHERE       MaNCC IN(SELECT MaNCC
                     FROM   DONDH
                     WHERE Covert(char(7), NgayDH, 21) = “2017-01” )
```

- *Hoặc dùng EXISTS*

```
SELECT      TenNCC, DienThoai
FROM        NHACC
WHERE       EXISTS (SELECT *
                    FROM   DONDH
                    WHERE Covert(char(7), NgayDH, 21) = “2017-01”
                    AND    NHACC.MaNCC = DONDH.MaNCC)
```

Các câu lệnh truy vấn dữ liệu (tt)

- *Ví dụ 2*: Để biết danh sách các nhà cung cấp nào mà công ty chưa bao giờ đặt hàng. Chúng ta có thể thực hiện câu truy vấn như sau:

```
SELECT    TenNhaCC, DienThoai
FROM              NHACC
WHERE     MaNCC NOT IN (SELECT Distinct MaNCC
                        FROM          DONDH)
```

- *Hoặc*

```
SELECT    TenNhaCC, DienThoai
FROM              NHACC
WHERE     MaNCC <> ALL (SELECT Distinct MaNCC
                        FROM          DONDH)
```

- ***Lưu ý:***

IN tương đương =ANY

NOT IN tương đượng <>ALL

Lệnh INSERT

- *Cách 1: Thêm trực tiếp một bộ*

  INSERT INTO bảng[<cột 1>, <cột 2>,…, cột n]
  VALUES(<giá trị 1>, <giá trị 2>,…, <giá trị n>)

- *Ví dụ*: thêm dữ liệu vào **HOCVIEN**

  HOCVIEN(MaHV, TenHV, Phai, NgaySinh, Malop)

  INSERT INTO HOCVIEN
  VALUES(‘001’, ‘Nguyen Van Tam’, 1, ‘05/06/1986’, ‘CT01’)

- *Hay*

  INSERT INTO HOCVIEN(MaHV, TenHV, NgaySinh, Malop)
  VALUES(‘001’, ‘Nguyen Van Tam’, ‘05/06/1986’, ‘CT01’)

Lệnh INSERT (tt)

- *Cách 2: Thêm nhiều bộ giá trị lấy từ các bộ giá trị của các bảng của CSDL*

  INSERT INTO bảng[<cột 1>, <cột 2>,…, <cột n>]

  SELECT … FROM …. WHERE

- *Ví dụ:*

  LOP-HV(TenLop, SiSo)

  INSERT INTO LOP-HV(TenLop, SiSo)
  SELECT TenLop, count(*)
  FORM HOCVIEN, LOP
  WHERE HOCVIEN.MaLop=LOP.MaLop
  GROUP BY TenLop

Lệnh UPDATE

- *Cú pháp:*

  UPDATE <tên bảng>
  SET <cột 1> = <giá trị 1>,
  <cột 2> = <giá trị 2>,
  ……,
  <cột n> = <giá trị n>
  [**WHERE** <điều kiện>]

- Ví dụ: Sửa nhân viên Dư Thanh Loan ở phòng HC sang phòng KD

  UPDATE NHANVIEN
  SET Phong= “KD”
  WHERE HoTen = “Dư Thanh Loan”

Lệnh DELETE

- ***Cú pháp:***

  DELETE [FROM] Bảng

  [**WHERE** <điều kiện>]

- *Ví dụ*: Xóa tất cả các nhân viên có LCB<700

  DELETE FROM NHANVIEN

  WHERE LCB<700

Tương tự như cấu trúc Switch-Case

Biểu thức CASE

- **Cú pháp**:

```
Case    biểu_thức:
        When    giá_tri_1|Bt_logic_1  Then Biểu_thức_kết_quả_1
        [When   giá_tri_2|Bt_logic_2  Then Biểu_thức_kết_quả_2
        …]
        [ELSE   biểu_thức_kết_quả_N]
End
```

- *Ví dụ 1:* hiển thị danh sách các vật tư có trong bảng VATTU theo từng loại hàng

```
SELECT MAVTU, TenVTU, Loai = CASE LEFT(MAVTU,2)
                                When "DD" THEN"Đầu DVD"
                                When "VD" THEN"Đầu VCD"
                                When "TV" THEN "Tivi"
                                When "TL" THEN "Tủ lạnh"
                                When "LO" THEN"Loa thùng"
                                ELSE  "chưa phân loại"
                             End, DVTinh
FROM                         VATTU
```

Biểu thức CASE (tt)

- *Ví dụ 2:* Hiển thị danh sách các vật tư trong bảng VATTU, thông tin bổ sung thêm chuỗi ghi chú, tuỳ thuộc vào giá trị của cột phần trăm giá bán.

```
SELECT      MaVT, TenVT, DVTinh, PhanTram,
                GhiChu = CASE
                            WHEN PhanTram < 20 Then “Lời ít”
                            WHEN PhanTram Between 20 And 40 Then “Lời nhiều”
                            ELSE “Rất lời”
                         END
FROM        VATTU
```

- *Ví dụ 3:* Giảm giá bán hàng trong tháng 2-2017 theo quy tắc sau: Nếu số lượng hàng <= 2 thì không giảm giá, Nếu số lượng hàng từ 3 đến 10 thì giảm 10%, Nếu số lượng hàng > 10 thì giảm 20%

```
UPDATE      CTPXUAT
SET DGXuat = CASE
             WHEN    SLXUAT <=2 THEN           DGXuat
             WHEN    SLXUAT BETWEEN 3 AND 10 THEN  DGXuat * 0.9
             ELSE    DGXuat*0.8
        END
FROM                 CTPXUAT, PXUAT
WHERE       CTPXUAT.SoPX =  PXUAT.SoPX
AND         Convert(char(7), NgayXuat, 21) = “2017-02”
```

Cấu trúc điều khiển

- Cấu trúc rẽ nhánh IF…ELSE

```
IF Biểu_thức_luận_lý
        Câu_lệnh1|khối_lệnh1
ELSE
        Câu_lệnh2|khối_lệnh2
```

- *Ví dụ*: Cho biết vật tư nào đã bán ra với số lượng nhiều hơn 4 không? Nếu có thì hiển thị danh sách đó ra, ngược lại thì thông báo chưa bán vật tư nào nhiều hơn 4

```
IF (SELECT COUNT(*) FROM CTPXUAT WHERE SLXUAT>4)>0
BEGIN
        Print “Danh sách các hàng hoá bán ra với số lượng lớn hơn 4”
        SELECT CTPXUAT.MAVT, TENVT, SLXUAT
        FROM            CTPXUAT, VATTU
        WHERE CTPXUAT.MaVT = VATTU.MaVT
        AND             SLXUAT>4
END
ELSE
        Print “chưa bán hàng nào với số lượng lớn hơn 4”
```

Cấu trúc điều khiển (tt)

- Cú pháp **IF** kết hợp với từ khoá **EXISTS** dùng để kiểm tra sự tồn tại của các dòng dữ liệu bên trong bảng.
- **Cú pháp:**

```
IF EXISTS(Câu_lệnh_SELECT)
        Câu_lệnh1|khối_lệnh1
ELSE
        Câu_lệnh2|khối_lệnh2
```

- *Ví dụ:*

```
IF EXISTS(SELECT COUNT(*) FROM CTPXUAT WHERE SLXUAT>4)
BEGIN
        Print “Danh sách các hàng hoá bán ra với số lượng lớn hơn 4”
        SELECT CTPXUAT.MAVT, TENVT, SLXUAT
        FROM            CTPXUAT, VATTU
        WHERE CTPXUAT.MaVT = VATTU.MaVT
        AND             SLXUAT>4
END
ELSE
        Print “chưa bán hàng nào với số lượng lớn hơn 4”
```

Cấu trúc lặp WHILE

- **Cú pháp:**

```
WHILE Biểu_thức_luận_lý
BEGIN
    Các_lệnh_lặp
END
```

- *Ví dụ 1:* Để in ra 10 số nguyên dương bắt đầu từ 100.

```
DECLARE        @Songuyen INT
SET            @Songuyen = 100
WHILE (@Songuyen < 110)
BEGIN
    Print “Số nguyên: ” + convert(char(3), @songuyen)
    SET @Songuyen = @Songuyen +1
END
```

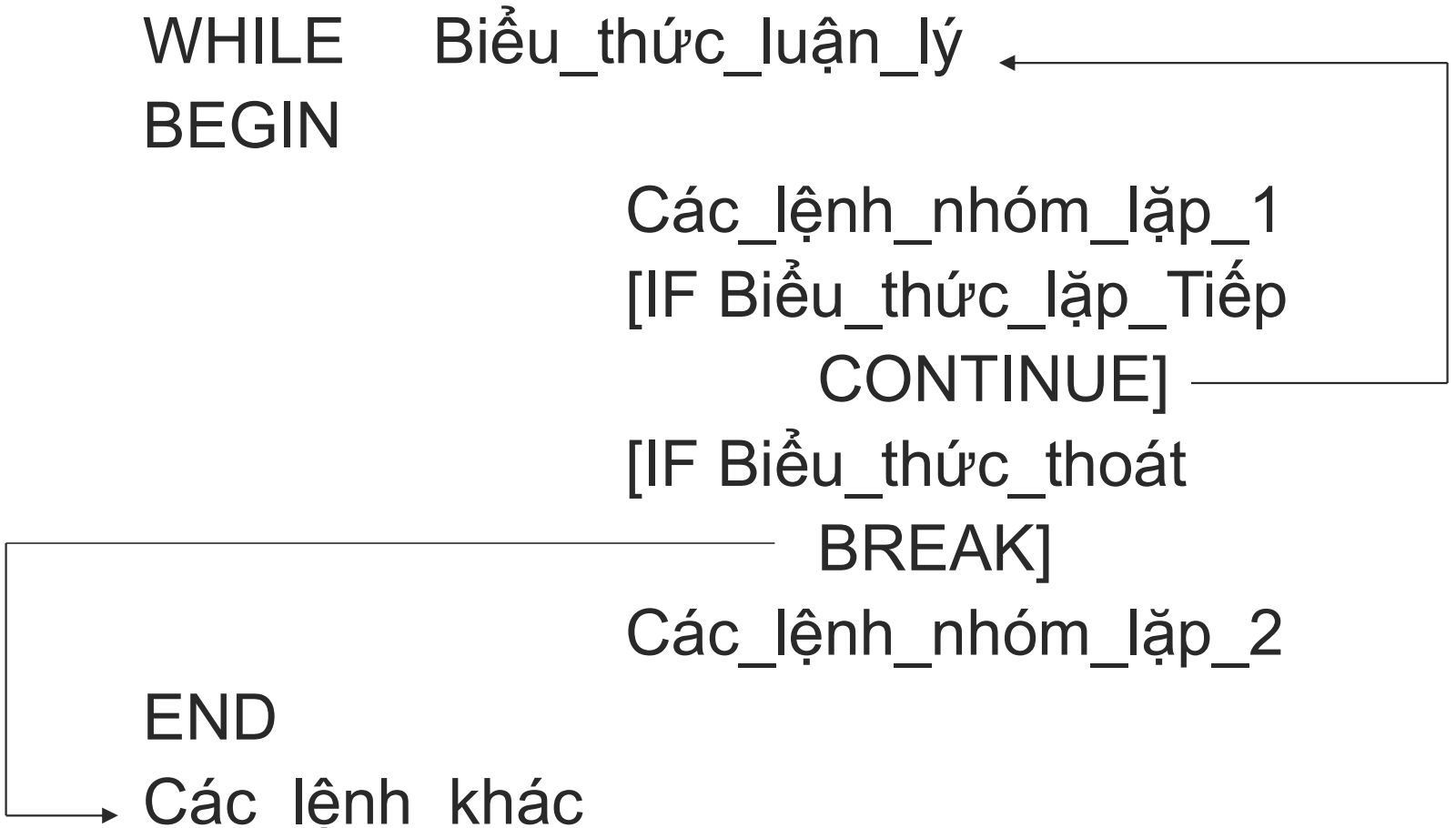
Tổng tích lũy (x = x+1)

Cấu trúc lặp WHILE (tt)

- Từ khoá BREAK lồng vào cấu trúc WHILE để có thể kết thúc việc lặp của các lệnh bên trong vòng lặp

```
DECLARE        @Songuyen int
SET            @Songuyen = 100
WHILE (@Songuyen < 110)
BEGIN
    Print "So nguyen: " + Convert(char(3), @songuyen)
    IF @Songuyen = 105
        Break
    SET   @Songuyen = @Songuyen +1
END
```

Cấu trúc lặp WHILE (tt)



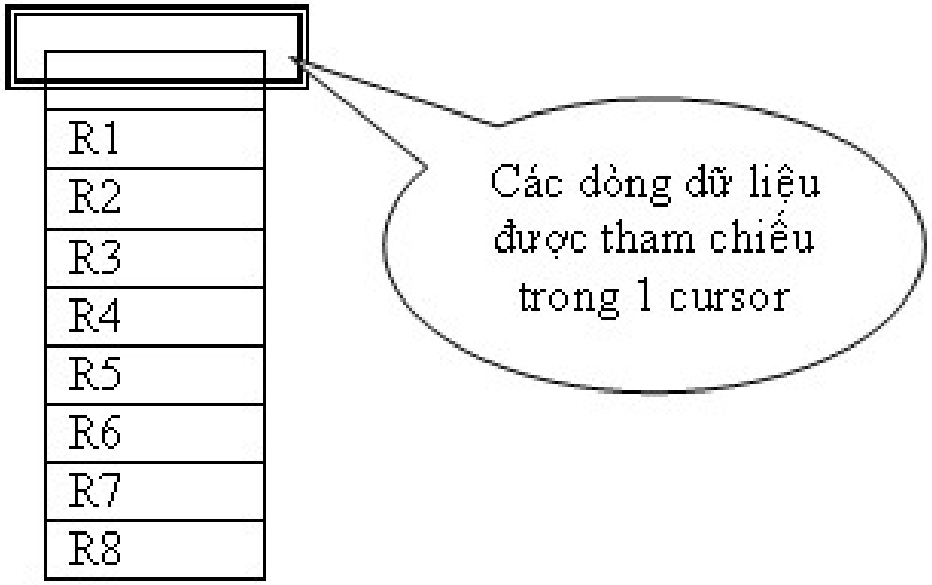
Cấu trúc lặp WHILE (tt)

- Thực hiện giống ví dụ trước, nhưng muốn in xót số nguyên 105. Chúng ta sử dụng cấu trúc lặp WHILE như sau

```
DECLARE        @Songuyen int
SET            @Songuyen = 100
WHILE (@Soguyen < 110)
BEGIN
    SET @Songuyen = @Songuyen + 1
    IF @Songuyen = 105
            CONTINUE
    Print “Số nguyên: ” + Convert(char(3), @songuyen)
END
```

Biến kiểu dữ liệu cursor

- CSDL quan hệ thường làm việc trên dữ liệu của nhiều dòng mẩu tin – còn gọi là các bộ mẩu tin. Ví dụ lệnh SELECT kết quả luôn trả về nhiều dòng dữ liệu hơn là một dòng dữ liệu. Tuy nhiên có một số ngôn ngữ lập trình việc xử lý và tính toán dữ liệu trên từng dòng riêng lẻ. Để đáp ứng được yêu cầu này SQL Server tạo ra một kiểu dữ liệu đó chính là kiểu cursor.



Biến kiểu dữ liệu cursor (tt)

- *Các bước sử dụng kiểu dữ liệu cursor*
  - Định nghĩa biến kiểu cursor bằng lệnh DECLARE.
  - Sử dụng lệnh OPEN để mở ra cursor đã định nghĩa trước đó.
  - Đọc và xử lý trên từng dòng dữ liệu bên trong cursor
  - Đóng cursor bằng lệnh CLOSE và DEALLOCATE.

Biến kiểu dữ liệu cursor (tt)

- Cú pháp định nghĩa biến có kiểu cursor

```
DECLARE    Tên_cursor        CURSOR
[LOCAL | GLOBAL]
[FORWARD_ONLY | SCROLL]
[STATIC | DYNAMIC | KEYSET]
[READ_ONLY | SCROLL_LOCK]
FOR Câu_lệnh SELECT
[FOR UPDATE [OF danh_sách_cột_n]]
```

- Trong đó:
  - Tên cursor: tên của biến kiểu cursor
  - Từ khoá LOCAL | GLOBAL: dùng chỉ phạm vi hoạt động của biến cursor hoặc là cục bộ (local) bên trong một thủ tục.
  - FORWARD_ONLY: đọc dữ liệu trong cursor theo chiều đi tới duyệt từ đầu mẫu tin đầu tiên đến mẫu tin cuối cùng.

Biến kiểu dữ liệu cursor (tt)

- SCROLL: đọc dữ liệu trong cursor được phép di chuyển tới lui, qua lại các dòng mẩu tin bên trong cursor tuỳ thích.
- STATIC: đọc dữ liệu bên trong cursor tĩnh. Khi đó nếu những người dùng khác có thay đổi bên dưới dữ liệu gốc thì các thay đổi đó sẽ không được cập nhật tự động trong dữ liệu của cursor. Bởi vì khi đó dữ liệu trong cursor chính là dữ liệu của bảng tạm đã được hệ thống sao chép và lưu trữ trong CSDL tempdb của hệ thống khi định nghĩa cursor.
- DYNAMIC: dùng chỉ định dữ liệu trong cursor là động. Khi đó việc cập nhật dữ liệu trong bảng cơ sở bởi những người dùng khác sẽ được cập nhật tự động trong dữ liệu cursor có kiểu là DYNAMIC.
- KEYSET: hoạt động giống với kiểu DYNAMIC, các thay đổi dữ liệu trên các cột không là khoá chính trong bảng cơ sở bởi những người dùng khác sẽ được cập nhật trong dữ liệu cursor. Tuy nhiên đối với mẩu tin vừa thêm mới hoặc các mẩu tin đã bị huỷ bỏ bởi những người dùng khác sẽ không được hiển thị trong dữ liệu cursor có kiểu là KEYSET.

Biến kiểu dữ liệu cursor (tt)

- READ_ONLY: chỉ định dữ liệu trong cursor chỉ đọc nhằm hạn chế việc sửa đổi dữ liệu bên trong cursor. Khi khai báo cursor với kiểu dữ liệu tĩnh (STATIC) thì dữ liệu trong cursor xem như chỉ đọc.
- SCROLL_LOCK: chỉ định hệ thống SQL Server tự động khoá các dòng mẩu tin cần phải thay đổi giá trị hoặc huỷ bỏ bên trong bảng nhằm bảo đảm các hành động cập nhật luôn thành công.
- SELECT: dùng để chỉ đến các cột bên trong bảng mà chúng ta cần đọc dữ liệu.
- Danh sách các cột cập nhật: chỉ định danh sách tên các cột sẽ được phép thay đổi giá trị trong cursor.

Biến kiểu dữ liệu cursor (tt)

- *Ví dụ 1*: để định nghĩa một biến cursor chứa toàn bộ các dòng dữ liệu bên trong bảng MAT_HANG, các dòng dữ liệu trong cursor cho phép được cập nhật.

```
DECLARE   Cur_MAT_HANG          CURSOR
DYNAMIC
FOR          SELECT   *      FROM   MATHANG
```

- *Ví dụ 2*: Định nghĩa một biến cursor chứa toàn bộ các dòng dữ liệu bên trong bảng NHACC, các dữ liệu trong cursor chỉ được phép đọc và việc đọc dữ liệu trong cursor chỉ theo một chiều đi tới.

```
DECLARE          Cur_NhaCC  CURSOR
FORWARD_ONLY
STATIC
READ_ONLY
FOR    SELECT  *       FROM NHACC
```

Biến kiểu dữ liệu cursor (tt)

- Mở Cursor
- **Cú pháp:**

  OPEN Tên_cursor
- *Trong đó:*

  Tên cursor: tên của biến cursor đã được định nghĩa trước đó bằng lệnh DECLARE
- *Ví dụ*: Mở các cursor đã định nghĩa ở ví dụ 1 trên. Chúng ta sử dụng lệnh OPEN như sau:

  OPEN cur_MAT_HANG

Biến kiểu dữ liệu cursor (tt)

- **Đọc và xử lý dữ liệu trong cursor**

  FETCH [Next | Prior | First | Last | Absolute n | Relative n]

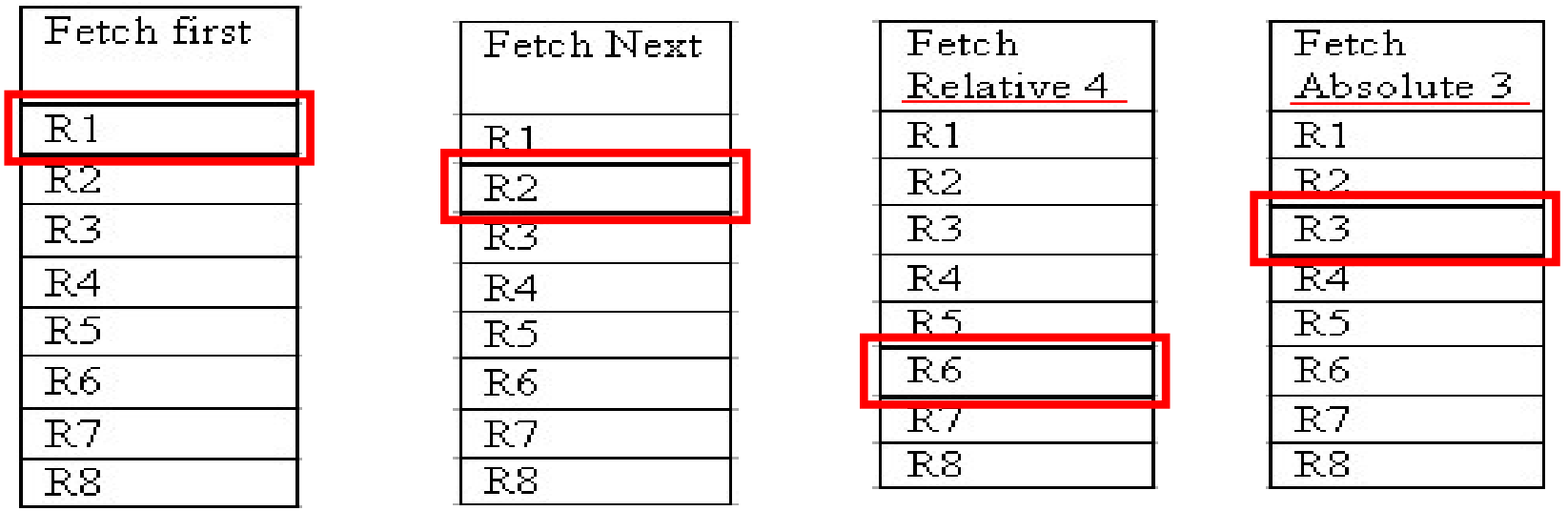
  FROM Tên_cursor

  [INTO danh_sách_biến]

- *Trong đó:*
  - ***Next, Prior, First, Last:*** *dùng để đọc dữ liệu kế tiếp, trước, đầu, sau cùng.*
  - ***Absolute:*** *dữ liệu chính xác thứ n trong cursor. N>0 chỉ định việc đọc dữ liệu tại dòng thứ n đếm từ dòng đầu tiên, n<0 dùng chỉ định việc đọc dữ liệu tại dòng thứ n được đếm ngược từ dòng cuối trở lên.*
  - ***Relative:*** *dùng chỉ định việc đọc dữ liệu tại một dòng tương đối so với dòng dữ liệu hiện hành. N là một số nguyên có thể dương có thể âm để chỉ định theo chiều tới hoặc lui so với dòng dữ liệu hiện hành.*

Biến kiểu dữ liệu cursor (tt)

- **Tên_cursor:** tên của biến cursor đã định nghĩa trước đó bằng lệnh DECLARE.
- **Danh sách biến:** danh sách tên các biến cục bộ đã được định nghĩa trước đó. Các biến này sẽ lưu trữ các giá trị dữ liệu được đọc từ lệnh FETCH.



Biến kiểu dữ liệu cursor (tt)

- **Đóng cursor**
- **Cú pháp:**

  CLOSE Tên_cursor

  DEALLOCATE Tên_cursor
- Trong đó
  - CLOSE giải phóng các dòng dữ liệu tham chiếu bên trong cursor.
  - Lệnh DEALLOCATE giải phóng thật sự biến cursor ra khỏi bộ nhớ

Biến kiểu dữ liệu cursor (tt)

- SQL Server cung cấp một biến hệ thống @@FETCH_STATUS dùng để kiểm tra trình trạng đọc dữ liệu thành công hay thất bại. Giá trị trả về 0 khi việc đọc dữ liệu là thành công.
- Cho lược đồ quan hệ như sau:
  - MAT_HANG(MaMH, TenMH, DVT, MaNCC)
  - PNHAP(MaPN, NgayNhap, ThanhTien)
  - CTPNHAP(MaMH, MaPN, SLNhap, DonGia)

ThanhTien = SUM (SLNhap * DonGia)

Biến kiểu dữ liệu cursor (tt)

- *Ví dụ 1*: Đọc dữ liệu cursor của bảng MAT_HANG chỉ đọc các vật tư là Tivi
- -- ***Khai báo biến cursor***

```
DECLARE       cur_MatHang   CURSOR
KEYSET
FOR
        SELECT  *  FROM   MAT_HANG
        WHERE   MaMH like 'TV%'
        ORDER BY            MaMH
```

- -- ***Mở cursor***

```
OPEN                        cur_MatHang
```

- -- ***Đọc dữ liệu***

```
FETCH NEXT  FROM cur_MatHang
WHILE @@FETCH_STATUS = 0
BEGIN
        -- Đọc tiếp dòng kế
        FETCH NEXT  FROM cur_MatHang
END
```

- -- ***Đóng cursor***

```
CLOSE           cur_MatHang
DEALLOCATE cur_MatHang
```

Biến kiểu dữ liệu cursor (tt)

- *Ví dụ 2*: cập nhật dữ liệu cho cột ThanhTien trong bảng PNHAP bằng cách duyệt qua từng phiếu nhập, tính ra trị giá nhập của từng phiếu căn cứ vào số lượng nhập và đơn giá nhập của từng vật tư trong bảng CTPNHAP, sau cùng cập nhật vào cột ThanhTien.

- ***-- Khai báo biến cursor, các biến cục bộ***

```
DECLARE    @Sopn char(4), @TongTT          Money
DECLARE    cur_Pnhap       CURSOR
FORWARD_ONLY
FOR
      SELECT          SOPN
      FROM            PNHAP
```

dữ liệu trả về là một ma trận m dòng 1 cột

- ***-- Mở cursor***

```
OPEN                  cur_Pnhap
```

Biến kiểu dữ liệu cursor (

- ***-- Đọc dữ liệu và cập nhật giá trị***

```
FETCH NEXT FROM  cur_Pnhap  INTO @SoPN
WHILE          @@FETCH_STATUS = 0
BEGIN
      SELECT @Tongtt = SUM(SLNhap*dongia)
       FROM                CTPNHAP
      WHERE MaPN = @SoPN
      Print 'dang cap nhat phieu nhap: ' + @SoPN
      UPDATEPNHAP
      SET                 Thanhtien = @TongTT
      Where               Current OF cur_Pnhap// sopn=@SOPN
      -- dịch  con trỏ đến dòng kế tiếp
      FETCH NEXT FROM        cur_Pnhap  INTO @Sopn
END
```

- ***-- Đóng cursor***

```
CLOSE cur_Pnhap
DEALLOCATE cur_Pnhap
```

Biến kiểu dữ liệu cursor (tt)

```
DECLARE         @Sopn char(4), @TongTT          Money
DECLARE         cur_Pnhap CURSOR
FORWARD_ONLY
FOR
        SELECT    SOPN
        FROM                    PNHAP
-- Mở cursor
OPEN                            cur_Pnhap
-- Đọc dữ liệu và cập nhật giá trị
WHILE           0 = 0
BEGIN
FETCH NEXT FROM         cur_Pnhap  INTO @Sopn
        IF    @@FETCH_STATUS<>0
                BREAK
        SELECT @Tongtg = SUM(SLNhap*dongia)
        FROM                    CTPNHAP
        WHERE     MaPN = @SoPN
        Print 'dang cap nhat phieu nhap: ' + @SoPN
        UPDATE    PNHAP
        SET                     Thanhtien = @TongTT
        Where                   Current OF cur_Pnhap
END
-- Đóng cursor
CLOSE cur_Pnhap
DEALLOCATE cur_Pnhap
```

Biến kiểu dữ liệu cursor (tt)

- ***--1. Khai báo biến cursor***

```
DECLARE      Tên_cursor          CURSOR
{kiểu đọc | cập nhật dữ liệu}
FOR
      Câu lệnh SELECT
```

- ***--2. Mở cursor***

```
OPEN                        Tên_cursor
```

- ***--3. Đọc dữ liệu và cập nhật giá trị***

```
WHILE 0=0
Begin
      FETCH NEXT FROM <Tên_cursor>
      [INTO danh_sách_biến]
      IF @@FETCH_STATUS <> 0
            Break
--cập nhật dữ liệu trong cursor
End
```

- ***--4. Đóng cursor***

```
CLOSE Tên_cursor
DEALLOCATE Tên_cursor
```

Biến kiểu dữ liệu cursor (tt)

- Khi nào chúng ta cần sử dụng kiểu dữ liệu cursor trong Transaction-SQL để giải quyết các vấn đề:
  - SQL Server là một hệ quản trị CSDL quan hệ (Relational Database Management System) do đó chúng ta nên chọn giải pháp làm việc trên các bộ mẩu tin.
  - Khi cần giải quyết vấn đề cập nhật dữ liệu thì luôn ưu tiên chọn các hướng giải quyết trên bộ mẩu tin bởi vì khi đó làm cho các bộ xử lý được nhanh hơn.
  - Sau cùng là hướng giải quyết theo kiểu cursor là giải pháp sau cùng nhất để chọn lựa khi không còn giải pháp nào tốt hơn

Các Hàm thường dùng

- *Các hàm chuyển đổi kiểu dữ liệu*
- ***Hàm CAST:** chuyển đổi một biểu thức nào đó sang một kiểu dữ liệu mong muốn.*
- **Cú pháp:**

  CAST(Biểu_thức AS kiểu_dữ_liệu)

- *Ví dụ:*

```
SELECT      MaVTU, TenVT,
            TyLe = CAST(PHANTRAM AS VARCHAR(3)) + "%"
FROM              VATTU
```

Các Hàm thường dùng (tt)

- ***Hàm CONVERT***: chuyển đổi một biểu thức nào đó sang một kiểu dữ liệu bất kỳ mong muốn nhưng có thể theo một định dạng nào đó.
- ***Cú pháp:***

  CONVERT (Kiểu_dữ_liệu, Biểu_thức[, định_dạng])
- Ví dụ:

```
SELECT SoHD,
      CONVERT(char(10),NgayHD, 103) AS   NGAYHD
FROM              DONDH
```

Các Hàm thường dùng (tt)

STT	Định dạng năm (yyyy)	Hiển thị dữ liệu
1	101	Mm/dd/yy
2	102	yy.mm.dd
3	103	Dd/mm/yy
4	104	dd.mm.yy
5	105	dd-mm-yy
6	106	Dd mon yy
7	107	Mon dd, yy
8	108	Hh:mm:ss
9	109	Mon dd yyyy hh:mm:ss
10	110	mm-dd-yy
11	111	Yy/mm/dd
12	112	Yymmdd
13	113	Dd mon yyyy hh:mm:ss
14	114	Hh:mm:ss:mmm
15	21 hoặc 121	Yyyy-mm-dd hh:mi:ss.mmm
16	20 hoặc 120	Yyyy-mm-dd hh:mi:ss

Các làm thường dùng (tt)

- ***Hàm STR:*** chuyển đổi kiểu dữ liệu số sang kiểu dữ liệu chuỗi. Phải đảm bảo đủ khoảng trắng để chứa các ký số khi chuyển sang kiểu dữ liệu chuỗi.
- ***Cú pháp:***

  STR(Số_thực, Số_ký_tự[, Số_lẻ])
- *Ví dụ:*

```
SELECT        TenVT, SLNhap =
    STR(SLNhap, 5) + " " + DVTinh
FROM          VATTU, CTPNHAP
WHERE VATTU.MaVT =  CTPNHAP.MaVT
```

Các làm thường dùng (tt)

- **Các hàm về ngày**
- **DATEDIFF**: trả về 1 số nguyên khoảng cách của hai ngày theo một đơn vị thời gian bất kỳ

  DATEDIFF(don_vi, ngay1, ngay2)
- *Ví dụ:*

```
SELECT  MADH, SONGAY = DATEDIFF(DD,
                                NGAYDH,NGAYGH)
FROM DONDH
```

- **DATENAME**: trả về một chuỗi thời gian đại diện của 1 ngày chỉ định theo một đơn vị thời gian bất kỳ

  DATENAME(Don_vi, ngay)
- *Ví dụ:*

```
SELECT  MADH, THU = DATENAME(DW, NGAYDH)
FROM    DONDH
```

Các làm thường dùng (tt)

- **GETDATE**: trả về ngày giờ hiện hành của hệ thống

  GETDATE()
- **DATEPART**: trả về 1 số nguyên chỉ định thời gian đại diện của 1 ngày theo một đơn vị thời gian bất kỳ

  DATEPART(Don_vi, ngay)
- *Ví dụ:*

  SELECT SODH, THANG = DATEPART(MM, NGAYDH)

  FROM DONDH

STORE PROCEDURE (THỦ TỤC) -p.2-

- Khái niệm
- Thủ tục
- Thủ tục với tham số đầu vào
- Thủ tục với tham số đầu ra
- Thủ tục có lệnh trả về Return
- Sử dụng bảng tạm trong thủ tục
- Tham số cursor bên trong thủ tục

Khái niệm

- Thủ tục nội tại là một tập hợp chứa các dòng lệnh, các biến và các cấu trúc điều khiển trong ngôn ngữ Transaction-SQL dùng để thực hiện một hành động nào đó.
- Các nét đặc trưng
  - Tên thủ tục
  - Tham số truyền giá trị vào
  - Tham số đón nhận giá trị ra
  - Trong thủ tục nội tại được phép gọi thực thi một thủ tục nội tại khác
  - Có tính cục bộ bên trong một cơ sở dữ liệu lưu trữ thủ tục đó
  - Có thể gọi thực hiện trong môi trường không phải Microsoft SQL Server.

Khái niệm (tt)

- Lợi ích của thủ tục
  - Tốc độ xử lý của các thủ tục nội tại rất nhanh.
  - Việc tổ chức và phân chia các xử lý thành hai nơi khác nhau: tại máy chủ hoặc tại máy trạm sẽ giúp giảm thời gian xây dựng ứng dụng.
- Thủ tục hệ thống
  - Bắt đầu bằng chữ **sp_** và hầu hết tất cả các thủ tục hệ thống được lưu trữ bên trong CSDL Master.

Thủ tục

- ***Tạo mới thủ tục***
- ***Cú pháp:***

CREATE PROC[EDURE] Tên_thủ_tục
AS
[Declare biến_cục_bộ]
các_lệnh

Thủ tục (tt)

- *Ví dụ*: cho lược đồ CSDL như sau:

MAT_HANG(MaMH, TenMH, DVT, MaNCC)

PHIEU_XUAT(SoPX, NgayXuat, #SoDH)

CTPX(Ma_MH, SoPX, SLXuat, DGXuat)

HOA_DONDH(MaDH, NgayDat)

CTDH(MaDH, MaMH, SLDH, DonGia)

Thủ tục (tt)

- *Ví dụ: Cho biết mặt hàng nào có doanh số bán cao nhất trong tháng 01/2017.*

```
CREATE    PROC  sp_MaxSLHang
AS
      Declare          @TenMH varchar(50), @MaxSL int
      Select   @TenMH=RTRIM(TenMH), @MaxSL=SLXuat
      From    CTPX, PHIEU_XUAT, MAT_HANG
      Where  CTPX.SoPX= PHIEU_XUAT.SoPX
      And      MAT_HANG.MaMH=CTPX.MaMH
      And      convert(char(7), NgayXuat, 21)= '2017-01'
      And      SLXuat = (Select Max(SLXuat)
                         From CTPX, PHIEU_XUAT
                         Where   CTPX.SoPX=PHIEU_XUAT.SoPX
                         And convert(char(7), NgayXuat, 21)= '2017-01')
Print @TenMH + 'Co doanh so cao nhat la' + Cast(@MaxSL as
char(10))
```

Thủ tục (tt)

- ***Gọi thực hiện thủ tục:***
  Cú pháp:
  EXEC[UTE] Tên_thủ_tục
  Ví dụ:
  EXEC sp_MaxSLHang
- ***Thay đổi nội dung thủ tục***
  Cú pháp:
  ALTER PROC[EDURE] Tên_thủ_tục
  AS
  [Declare biến_cục_bộ]
  Các_lệnh.

Thủ tục (tt)

```
ALTER PROC sp_MaxSLHang
AS
Declare    @TenMH varchar(50), @MaxSL int
IF NOT EXISTS(Select Ma_mh
              From CTPX, PHIEU_XUAT
              Where CTPX.SoPX= PHIEU_XUAT.SoPX
              And convert(char(7), NgayXuat, 21)= '2007-01')
Begin
      Print 'thang 01 nam 2007 chưa bán mặt hàng nào cả'
      Return
End
      Select @TenMH=RTRIM(TenMH), @MaxSL=SLXuat
      From              CTPX, PHIEU_XUAT, MAT_HANG
      Where CTPX.SoPX= PHIEU_XUAT.SoPX
      And   MAT_HANG.MaMH=CTPX.MaMH
      And  convert(char(7), NgayXuat, 21)= '2007-01'
      And  SLXuat = (Select Max(SLXuat)
      From CTPX, PHIEU_XUAT
      Where             CTPX.SoPX=PHIEU_XUAT.SoPX
      And convert(char(7), NgayXuat, 21)= '2007-01')
      Print @TenMH + 'Co doanh so cao nhat la' + Cast(@MaxSL as char(10))
```

Thủ tục với tham số đầu vào

- **Cú pháp:**

```
CREATE PROC[EDURE] Tên_thủ_tục
    @Tên_tham_số kiểu_dữ_liệu [= giá_trị]
AS
    [Declare biến_cục_bộ]
        các_lệnh
```

Thủ tục với tham số đầu vào (tt)

- *Ví dụ 1*: Tạo thủ tục tính tổng giá trị của một phiếu xuất hàng hoá có một tham số vào là số phiếu xuất với kiểu dữ liệu là chuỗi.

```
CREATE PROC   sp_TongTGXuat    @SoPX char(4)
AS
    Declare        @TongTG      money
    Select  @TongTG=SUM(SLXuat*DGXuat)
    From    CTPX
    Where @SoPX=SoPX
    Print 'Tri gia phieu xuat' + CAST(@SoPX AS char(4))
    Print 'là: ' + CAST(@TongTG as Varchar(15))
```

- *Gọi thực hiện thủ tục*

```
Exec  sp_TongTGXuat 'PX01'
```

- *Hoặc:*

```
Exec  sp_TongTGXuat @SoPX='PX01'
```

Thủ tục với tham số đầu vào (tt)

- ***Ví dụ 2***: tạo thủ tục tính số đặt hàng của một mặt hàng trong một đơn đặt hàng có 2 tham số vào là số đặt hàng và mã mặt hàng.

```
CREATE PROC      sp_TinhSLDat      @Sodh char(4), @MaMH char(4)
AS
        Declare @Sldat int
        IF NOT EXISTS(Select MoDH From CTDH
                                Where MaDH=@SoDH And MaMH=@MaMH)
        Begin
                Print 'khong hop le, xem lai don dat hang'
                Return
        End
        Select @SLDat = SLDat
        From CTDH
        Where SoDH = @SoDH And MaMH = @MaMH
        Print 'Don dat hang ' + @SoDH
        Print 'Voi ma mat hang ' + @MaMH
        Print 'Co so luong dat la: ' + Cast(@SLDat as varchar(10))
```

- *Gọi thực hiện thủ tục:*

```
        Exec      sp_TinhSLDat 'DH01', 'Fe'
```

- *Hoặc*

```
        Exec      sp_TinhSLDat  @MaMH = 'Fe', @SoDH = 'DH01'
```

Thủ tục với tham số đầu vào (tt)

- *Ví dụ:* tạo thủ tục thêm mới dữ liệu vào bảng MAT_HANG với tên sp_MATHANG_Them gồm có 4 tham số vào chính là các giá trị thêm mới cho các cột trong bảng MAT_HANG: mã mặt hàng, tên mặt hàng, đơn vị tính… Trong đó cần kiểm tra các ràng buộc dữ liệu phải hợp lệ trước khi thực hiện lệnh INSERT INTO để thêm dữ liệu vào bảng MAT_HANG.
  - Mã mặt hàng phải duy nhất
  - Tỷ lệ phần trăm phải nằm trong miền giá trị 0 đến 100

Thủ tục với tham số đầu vào (tt)

```
CREATE          PROC  SP_MATHANG_Them
    @MaMH char(4), @TenMH varchar(50), @DVT varchar(50),
    @PhanTram INT
AS
    --Định nghĩa chuỗi lỗi
    DECLARE @ErrMsg varchar(200)
    --Kiểm tra có mặt hàng chưa?
    IF EXISTS(SELECT MaMH FROM MAT_HANG WHERE
                                        MaMH=@MaMH)
    BEGIN
        SET @ErrMsg = 'Mã mặt hàng [' + @MaMH + '] đã có'
        RAISERROR(@ErrMsg, 16, 1)
        RETURN
    END
```

Thủ tục với tham số đầu vào (tt)

```
--Kiểm tra tỷ lệ phần trăm nằm ngoài 0..100
    IF @PhamTram NOT BETWEEN 0 AND 100
    BEGIN
        SET @ErrMsg = 'Tỷ lệ phần trăm nằm trong đoạn [0, 100]'
        RAISERROR(@ErrMsg, 16, 1)
        Return
    END
--Khi các RBTV hợp lệ thì thêm dữ liệu vào bảng MatHang
    INSERT INTO MAT_HANG(MaMH, TenMH, DVT, PhanTram)
    VALUES(@MaMH, @TenMH, @DvTinh, @PhanTram)
```

Thủ tục với tham số đầu ra

- **Cú pháp:**

```
CREATE    PROC        Tên_thủ_tục
    @Tên_tham_số kiểu_dữ_liệu OUTPUT [,…]
AS
    [Declare Biến cục bộ]
    Các_lệnh
```

Thủ tục với tham số đầu ra (tt)

- ***Ví dụ: tạo thủ tục tính số đặt hàng của một mặt hàng trong một đơn đặt hàng có 2 tham số vào là số đặt hàng và mã mặt hàng, trả ra số lượng đặt hàng của một vật tư tương ứng trong đơn đặt hàng thông qua tham số đầu ra.***

```
CREATE PROC                    sp_TinhSLDat
    @Sodh char(4), @MaMH char(4), @SLDat int OUTPUT
AS
    IF NOT EXISTS(Select MaDH From CTDH
            Where MaDH=@SoDH And MaMH=@MaMH)
    Begin
            Print 'khong hop le, xem lai don dat hang'
            Return
    End
    Select @SLDat = SLDH
    From CTDH
    Where MaDH = @SoDH And MaMH = @MaMH
```

Thủ tục với tham số đầu ra (tt)

- Gọi thực hiện thủ tục

```
DECLARE @SLDatHang int
EXEC    sp_TinhSLDat  @MaMH = 'Fe', @SoDH = 'DH01',
      @SLDat = @SLDatHang OUTPUT
Print 'Don dat hang DH01 với mặt hàng Fe'
Print 'co so luong dat la: ' + CAST(@SLDatHang AS
varchar(10))
```

Thủ tục có lệnh trả về Return

- Return không có giá trị chỉ định thì thủ tục sẽ trả về giá trị là không (0).
- Return [Số_nguyên]
- *Ví dụ:* Tạo thủ tục tính tổng số lượng đặt hàng của một mặt hàng đối với một nhà cung cấp chỉ định, kiểm tra xem giá trị của mặt hàng và mã nhà cung cấp mà người dùng truyền vào thủ tục có đúng hay không? Qui định thủ tục trả về 1 khi mã mặt hàng không tồn tại, trả về 2 khi mã nhà cung cấp không tồn tại.

Thủ tục có lệnh trả về Return (tt)

```
CREATE PROC          sp_TinhTongSLDat
    @MaNCC   char(3), @MaMH char(4),
    @TongSLdat                  INT      OUTPUT
AS
    IF NOT   EXISTS(Select * From Mat_Hang Where
                        MaMH=@MaMH)
        Return 1
    IF NOT   EXISTS(Select * From Mat_Hang Where
                              MaNCC=@MaNCC)
        Return 2
    Select @TongSLdat = SUM(SLDat)
    From HoaDon_DH, CTDH
    Where       HoaDon_DH.MaDH = CTDH.MaDH
    And         MaNCC=@MaNCC And MaVTu=@MaVTu
    IF @TongSLdat IS NULL
        Set @TongSLdat=0
    Return
```

Thủ tục có lệnh trả về Return(tt)

- **Gọi thực hiện thủ tục:**

```
Declare    @TongSLD INT, @Ketqua INT
EXEC       @ketqua = sp_TinhTongSLDat 'NCCA', 'Fe',
@TongSLdat=@TongSLD output
IF @ketqua =1
     Print 'Mã mặt hàng không hợp lệ'
ELSE IF   @ketqua=2
     Print 'Mã nhà cung cấp không hợp lệ'
ELSE
     Print 'Tổng số lượng đặt là: ' + CAST(@TongSLD as
char(10))
```

Sử dụng bảng tạm trong thủ tục

- **Cú pháp:**

  SELECT danh_sách_các_cột INTO #Tên_bảng_tạm

  FROM Tên_bảng_dữ_liệu

  (#): tạo ra các bảng tạm cục bộ

  (##): tạo ra các bảng tạm toàn cục

Sử dụng bảng tạm trong thủ tục (tt)

- *Ví dụ: Tạo thủ tục tính mặt hàng nào có doanh thu bán ra cao nhất trong một năm tháng bất kỳ.*

```
CREATE PROC      sp_TinhDTCaoNhat
    @namThang    char(7),
    @TenMH       char(50) OUTPUT, @TongTien Money    OUTPUT
AS
    Select MH.MaMH, TenMH, Sum(SLXuat*DGXuat) AS TT
    INTO #DoanhThu
    From    Phieu_xuat PX, CTPX, Mat_Hang MH
    Where  PX.SoPX = CTPX.SoPX And CTPX.MaMH = MH.MaMH
    And Convert(char(7), ngayxuat, 21) = @namthang
    Group By         MH.MaMH, TenMH
    Order by         3 DESC
    Select   Top 1 @TenMH=TenMH, @Tongtien = TT
    From    #DoanhThu
```

Sử dụng bảng tạm trong thủ tục (tt)

- ***Gọi thực hiện thủ tục***

```
Declare    @TenMH       char(50), @TongTien        Money
EXEC       sp_TinhDTCaoNhat  '2017-01', @TenMH
OUTPUT, @TongTien   OUTPUT
IF   @TenMH IS NULL
     Print 'không có dữ liệu tính toán'
ELSE
Begin
     Print Rtrim(@TenMH) + 'có doanh thu cao nhất'
     Print 'là ' + CAST(@TongTien AS Varchar(20)) + 'VND'
End
```

Tham số cursor bên trong thủ tục

- **Tham số cursor trả về danh sách các dòng dữ liệu theo điều kiện chọn lọc nào đó.**
- **Cursor được chia làm 2 phần: bên trong thủ tục và bên ngoài thủ tục.**
  - **Các hành động trong thủ tục**:định nghĩa dữ liệu cho biến kiểu cursor và mở cursor.
  - **Các hành động bên ngoài thủ tục:** đọc từng dòng dữ liệu bên trong cursor và sau cùng là đóng cursor lại.

Tham số cursor bên trong thủ tục (tt)

- *Ví dụ:* tạo thủ tục trả về danh sách các mã vật tư đã bán ra nhiều nhất trong năm tháng nào đó.
- *Bước 1: tạo thủ tục có tham số kiểu dữ liệu cursor chứa danh sách các vật tư đã bán ra nhiều nhất.*

Tham số cursor bên trong thủ tục (tt)

```
CREATE PROC    sp_TinhDsoBan
    @NamThang  char(6),
    @cur_Dsmh CURSOR        VARYING      OUTPUT
AS
--Tạo bảng tạm tính ra tổng số lượng bán
    SELECT CTDH.MAMH, SUM(SLDH) AS TongSLBan
    INTO #TongSLBan
    FROM CTDH, MAT_HANG, HOA_DONDH
    WHERE        Convert(char(6), NgayDat, 112) =
                                        @NamThang
    AND          CTDH.MaMH=MAT_HANG.MaMH
    AND          CTDH.MaDH=HOA_DONDH.MaDH
    Group By     CTDH.MaMH
```

Tham số cursor bên trong thủ tục (tt)

```
-Kiểm tra dữ liệu có phát sinh
IF EXISTS(SELECT MaMH FROM #TongSLBan)
Begin
      -- Khởi tạo giá trị biến CURSOR
      SET @cur_Dsmh  = CURSOR Forward_Only
      FOR
                  SELECT MAMH, TongSLBan
                  From      #TongSLBan
                  Where      TongSLBan = (SELECT MAX(TongSLBan)
                                                FROM    #TongSLBan)
      --Mở cursor
      OPEN @cur_Dsmh
      DROP    TABLE #TongSLBan
      Return
End
-- Khi không có dữ liệu phát sinh
DROP          Table #TongSLBan
Return 1
```

Tham số cursor bên trong thủ tục (tt)

- Bước 2: đọc cursor, đón nhận danh sách các mã mặt hàng đã bán ra nhiều nhất trong tháng 01 năm 2002

```
DECLARE     @cur_Dsmh          CURSOR, @Gtmh INT,
            @MaMH   char(4), @TongslBan INT
EXEC        @Gtmh = sp_TinhDsoBan '200702', @cur_Dsmh OUTPUT
--Xử lý tiếp sau đó
IF @Gtmh = 0
Begin
      Print 'danh sách các mặt hàng'
      While(0=0)
      Begin
            Fetch Next From @cur_Dsmh INTO @MaMH, @TongslBan
            IF @@Fetch_status<>0
                  Break;
            Print 'Mã vật tư: ' + @MaMH
            Print 'Tổng số lượng: ' + CAST(@TongslBan AS varchar(10))
            Print Replicate('-', 50)
      End
end
ELSE
            Print 'không có bán hàng trong năm tháng chỉ định'
```