CHƯƠNG 2: LẬP TRÌNH VỚI SQL

Chương 2: Lập trình với SQL

- Nội dung
  - 2.1. Biến và các phép toán trên biến
  - 2.2. Cấu trúc điều khiển
  - 2.3. Sử dụng biến kiểu dữ liệu cursor
  - 2.4. Thủ tục
  - 2.5. Hàm
  - 2.6. Trigger

2.1. Biến và các pháp toán trên biến

- 2.1.1 Khai báo và sử dụng biến
- 2.1.2 Các toán tử

2.1.1 Khai báo và sử dụng biến

- Biến cục bộ
- Biến hệ thống

Slide 42

Biến cục bộ

- Biến tạm thời lưu giữ kết quả trong quá trình xử lý, thoát chương trình thì giá trị của biến không còn nữa.
- Trong Transaction-SQL có 2 loại biến
  - Biến cục bộ
  - Biến hệ thống
- Khai báo biến cục bộ
  - DECLARE @ten_bien Kieu_du_lieu[,…]

Slide 43

Biến cục bộ

- Ví dụ:
  - DECLARE @tong_sl int, @hoten nvarchar(20)
  - DECLARE @ngay_xh Datetime
- Gán giá trị cho biến
  - Dùng lệnh SELECT cùng với phép gán =
  - Dùng lệnh SET để gán các giá trị cụ thể hoặc biểu thức tính toán hoặc giá trị tính toán từ các biến khác
- Ví dụ:
  - DECLARE @tong int
  - SET @tong = 10

Slide 44

Biến cục bộ

- Ví dụ 1:

```
DECLARE @tong_sl  INT
SELECT @tong_sl = SUM(SoLuong)
    From tblctdondh
```

- Ví dụ 2:

```
DECLARE @Min_sl   INT, @Max_sl  INT
SELECT          @Min_sl=min(SoLuong),
                @Max_sl=max(Soluong)
    FROM tblctdondh
```

Slide 45

Biến cục bộ

- Xem giá trị hiện hành của biến

```
PRINT @ten_bien
```

- Ví dụ:

```
DECLARE @Min_sl   INT, @Max_sl  INT
SELECT          @Min_sl=min(SoLuong),
                @Max_sl=max(Soluong)
    FROM tblctdondh
PRINT 'So luong thap nhat la: '
PRINT @Min_sl
PRINT 'So luong cao nhat la: ' +
    CONVERT( varchar(10), @Max_sl)
```

Slide 46

Biến cục bộ

- Phạm vi hoạt động của biến:

  Trong T_SQL phạm vi hoạt động của biến chỉ trong 1 thủ tục nội tại hoặc 1 lô (batch)
- Khái niệm lô: lô được xem như là một nhóm tập hợp của một hoặc nhiều câu lệnh T_sql được biên dịch đồng thời cùng lúc trong Microsoft SQL Server và kết thúc bởi từ khoá go.

Slide 47

Biến cục bộ

- Ví dụ về lô

```
Select * from tblnhacc
    Order by tennhacc
Select * from tblvattu
    Order by tenvtu Desc
Go
```

```
Declare @sNgaydh Datetime
Select  @sNgaydh=Max(ngaydh)
    From tbldondh
Go
Print 'Ngàyđh gần đây nhất:'+
    convert(char(12),@sNgaydh)
Go
```

- Đây là 2 lô và máy sẽ báo lỗi biến @sNgaydh ở lô 2

Slide 48

Biến hệ thống

- Không giống như các ngôn ngữ lập trình khác, Transaction-SQL không có khái niệm biến toàn cục.
- Nó cung cấp danh sách các biến hệ thống, bắt đầu bằng @@, giá trị của các biến hệ thống do MS SQL Server cung cấp
- Người lập trình không can thiệp được vào biến hệ thống của MS SQL Server
- Người sử dụng thường dùng các giá trị của nó để thực hiện các hoạt động theo ý riêng của mình.

Slide 49

Biến hệ thống

- Ví dụ:
  - @@ROWCOUNT: trả về tổng số bản ghi hiện tại

```
SELECT * FROM tblCanBo
    PRINT @@ROWCOUNT
SELECT * FROM tblCanBo WHERE MaDV=2
    PRINT @@ROWCOUNT
```

Slide 50

Biến hệ thống

- Một số biến hệ thống thường dùng
  - @@VERSION
  - @@FETCH_STATUS
    - Đọc dữ liệu trong bảng theo từng dòng cursor
    - Khi đọc mẫu tin thành công thì biến có giá trị = 0
  - @@ROWCOUNT
  - @@SERVERNAME
  - @@ERROR
  - …

Slide 51

2.1.2. Các toán tử

- Toán tử số học
  - + Thực hiện phép cộng 2 số
  - - Thực hiện phép trừ 2 số
  - * Thực hiện phép nhân 2 số
  - / Thực hiện phép chia 2 số
  - % Thực hiện phép chia lấy phần dư
- Ví dụ:
  - SELECT 10 + (2*10)/5
  - SELECT 15%6

Slide 52

2.1.2. Các toán tử

- Toán tử nối xâu (chuỗi): +
- Ví dụ:
  - SELECT 'hello' + ' ' + ' the world'
  - SELECT 'Ten Nha cung cap la: ' + TenNCC
    FROM NCC
    WHERE MNCC='CC001'

Slide 53

2.1.2. Các toán tử

- Toán tử so sánh
  - = Thực hiện phép so sánh bằng
  - > Thực hiện phép so sánh lớn hơn
  - < Thực hiện phép so sánh nhỏ hơn
  - >= Thực hiện phép so sánh lớn hơn hoặc bằng
  - <= Thực hiện phép so sánh nhỏ hơn hoặc bằng
  - <> Thực hiện phép so sánh khác
  - != Thực hiện phép so sánh khác
  - !> Thực hiện phép so sánh không lớn hơn
  - !< Thực hiện phép so sánh không nhỏ hơn

Slide 54

2.1.3. Các hàm thường dùng

- Các hàm chuyển đổi kiểu dữ liệu
  Cast, Convert, Str
- Các hàm ngày giờ
  Getdate, Day, Month, Year
- Các hàm toán học
  Abs, Pi, Round, Sqrt
- Các hàm xử lý chuỗi
  Upper, Lower, Left, Right, Substring, Ltrim, Rtrim, Len, Char

Slide 55

2.1.3. Các hàm thường dùng

- Hàm convert:
- Convert (Kiểu_dữ_liệu, Biểu_thức[, Định dạng])
  - Kiểu dữ liệu: tên của kiểu mà biểu thức sẽ chuyển đổi sang.
  - Biểu thức: tên cột hoặc biểu thức muốn chuyển kiểu.
  - Định dạng: là một con số chỉ định việc định dạng cho việc chuyển đổi dữ liệu từ dạng ngày sang chuỗi.

Slide 56

2.1.3. Các hàm thường dùng

Định dạng năm (yy)	*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
11	111	yy/mm/dd
	21 hoặc 121	yyyy-mm-dd
	20 hoặc 120	yyyy-mm-dd

Slide 57

2.2. Các cấu trúc điều khiển

2.2.1. Các câu lệnh truy vấn dữ liệu
2.2.2. Cấu trúc rẽ nhánh IF
2.2.3. Cấu trúc CASE
2.2.4. Cấu trúc lặp WHILE

Slide 58

2.2.1. Các câu lệnh truy vấn dữ liệu

1. SELECT
2. Truy vấn con
3. Lệnh INSERT INTO
4. Lệnh DELETE FROM
5. Lệnh UPDATE SET

Slide 59

1. Câu lệnh Select

- Cú pháp:

```
Select Danh_sách_các_cột
   From Tên_bảng [Bí_danh]
   [ Inner | Left | Right | Full  Join Tên_bảng_qh
            On điều_kiện_quan_hệ,… ]
   [ Where Điều_kiện ]
   [ Group By Danh_sách_cột_nhóm_dl ]
   [ Having Điều_kiện_lọc_nhóm ]
   [ Order By Tên_cột_sx Desc,…]
```

Slide 60

2. Truy vấn con

- Khái niệm: là câu lệnh truy vấn chọn lựa được lồng vào các câu lệnh truy vấn khác để thực hiện các truy vấn phức tạp.
- Lưu ý:
  - Cần mở và đóng ngoặc cho câu lệnh truy vấn con
  - Kết quả trả về của truy vấn con có thể là một giá trị hoặc một danh sách bảng
  - Cấp độ lồng nhau của truy vấn trong SQL là không giới hạn

Slide 61

2. Truy vấn con

- Truy vấn con trả về giá trị đơn lẻ:

- Hiện các đơn đặt hàng trong ngày gần đây nhất

```
Select  sodh
From tbldondh
where ngaydh = ( Select max(ngaydh)
                 From tbldondh)
```

Slide 62

2. Truy vấn con

- Truy vấn con trả về danh sách các trị:
- Ví dụ 1

- Hiện họ tên nhà cung cấp, Điện thoại của bảng NHACC mà công ty đã có đơn đặt hàng trong tháng 05-2010
- Select Tennhacc, Dienthoai
  From tblnhacc
  Where manhacc IN (Select manhacc
  From tbldondh
  Where convert(char(7),ngaydh,21)='2010-05')

Slide 63

2. Truy vấn con

- Ví dụ 2

Hiện danh sách các nhà cung cấp mà công ty chưa bao giờ đặt hàng.

Select Tennhacc, Dienthoai
From tblnhacc
Where manhacc NOT IN
(Select distinct manhacc
From tbldondh)

Slide 64

3. Lệnh Insert Into

- Thêm mới một dòng dữ liệu

INSERT INTO Tên_bảng [(DS_cột)]
Values (Danh_sách_giá_trị)

- Thêm mới nhiều dòng dữ liệu:

INSERT INTO Tên_bảng ([DS_cột])
Select Danh_sách_cột
From Tên_bảng_dl_nguồn
Where Điều_kiện_lọc

Slide 65

3. Lệnh Insert Into

- Ví dụ:
  - Tạo bảng ảo tính tổng số lượng nhập của các vật tư trong tháng 02/2010
  - Tạo bảng ảo dùng để tính tổng số lượng xuất của các vật tư trong tháng 02/2010
  - Sử dụng INSERT…INTO để thêm dữ liệu vào bảng TONKHO

Slide 66

3. Lệnh Insert Into

- Tạo bảng ảo tính tổng số lượng nhập của các vật tư trong tháng 10/2002:

```
CREATE VIEW vw_tongnhap
AS
    SELECT ct.mavtu, SUM(ct.slnhap) AS Tongnhap
    FROM tblpnhap pn
    INNER JOIN tblctpnhap ct ON pn.sopn = ct.sopn
    WHERE
        (CONVERT(char(7), pn.ngaynhap, 21) = '2010-02')
    GROUP BY ct.mavtu
```

Slide 67

3. Lệnh Insert Into

- Tạo bảng ảo dùng để tính tổng số lượng xuất của các vật tư trong tháng 01/2002:

```
CREATE VIEW vw_tongxuat
AS
    SELECT ct.mavtu, SUM(ct.slxuat) AS Tongxuat
    FROM tblpxuat px
    INNER JOIN tblctpxuat ct ON px.sopx = ct.sopx
    WHERE
        (CONVERT(char(7), px.ngayxuat, 21) = '2010-02')
    GROUP BY ct.mavtu
```

Slide 68

3. Lệnh Insert Into

- Sử dụng INSERT…INTO để thêm dữ liệu vào bảng TONKHO:

```
INSERT tbltonkho
Select '02-2010', TN.Mavtu, TONGNHAP, TONGXUAT,
 TONGNHAP-TONGXUAT
From vw_tongnhap tn left join vw_tongxuat  tx
On tn.mavtu=tx.mavtu
```

Slide 69

4. Lệnh Delete From

- Cú pháp:

```
DELETE [From] Tên_bảng
[ From Tên_bảng
    Inner | Left | Right | Join Tên_bảng_2
    On Biểu_thức_liên_kết ]
[Where Điều_kiện_xóa]
```

- Lưu ý: Trong lệnh Delete nếu quên không sử dụng mệnh đề Where thì tất cả các dòng dữ liệu của bảng sẽ bị xóa.

Slide 70

4. Lệnh Delete From

- Ví dụ: Huỷ bỏ các nhà cung cấp mà công ty chưa bao giờ đặt hàng

- Delete tblnhacc
  from **tblnhac**
  where **Manhacc not in**
  (select Manhacc from tblDonDH)

Slide 71

5. Lệnh Update Set

- Cú pháp:

  Update Tên_bảng
  Set Tên_cột = Biểu_thức [,…]
  [From Tên_bảng1
  Inner | Left | Right Join Tên_bảng2
  On Biểu_thức_liên_kết]
  [Where Điều_kiện_sửa_đổi]

- Lưu ý: Trong lệnh Update nếu không sử dụng mệnh đề Where thì tất cả các bản ghi trong bảng sẽ bị sửa đổi.

Slide 72

2.2.2. Cấu trúc rẽ nhánh IF … ELSE

- Cú pháp
  - IF (bieu_thuc_logic)
    Câu lệnh 1 | Khối lệnh 1
    [ELSE
    Câu lệnh 2 | Khối lệnh 2]
  - Lưu ý:
    - Khối lệnh có từ 2 lệnh trở lên, trong trường hợp này bắt buộc phải đặt các lệnh trong cặp BEGIN … END

Slide 73

2.2.2. Cấu trúc rẽ nhánh IF … ELSE

- Cấu trúc rẽ nhánh IF…ELSE
- Ví dụ
  - Hiển thị xem có vật tư nào đã bán trong 1 phiếu xuất có số lượng nhiều hơn 5 không? Nếu có thì in ra danh sách đó, ngược lại thông báo không có, chúng ta sử dụng cú pháp IF … ELSE như sau:

Slide 74

2.2.2. Cấu trúc rẽ nhánh IF … ELSE

- Cấu trúc rẽ nhánh IF…ELSE

```
if (select count(*) from tblctpxuat where slxuat>5) >0
        begin
        print 'danh sach các hàng hoá ban với so >5'
        select distinct vt.mavtu,vt.tenvtu
            from tblvattu vt
            inner join tblctpxuat ct on vt.mavtu=ct.mavtu
            where ct.slxuat>5
        end
      else
        PRINT 'Chua ban hang hoa nao voi so luong >5'
```

Slide 75

2.2.2. Cấu trúc rẽ nhánh IF … ELSE

- Dùng từ khóa EXISTS kết hợp với IF: với bài trên ta có như sau:

```
if exists (select * from tblctpxuat where slxuat>100)
    begin
    print 'danh sach các hàng hoá ban với so >100'
    select distinct vt.mavtu,vt.tenvtu
    from tblvattu vt
    inner join tblctpxuat ct on vt.mavtu=ct.mavtu
    where ct.slxuat>100
    end
else
print 'Chua ban hang hoa nao voi so luong >100'
```

Slide 76

2.2.3. Biểu thức Case

- Cú pháp:

```
CASE Biểu_thức
    When Giá_trị_1 Then Biểu_thức_1
    [ When Giá_trị_2 Then Biểu_thức_2
    …]
    [ ELSE Biểu_thức_N ]
    END
```

Slide 77

2.2.4. Cấu trúc lặp WHILE

- Cú pháp
  - WHILE (biểu_thức_logic)
    BEGIN
        Các lệnh lặp
    END

Slide 78

2.2.4. Cấu trúc lặp WHILE

- Ví dụ

```
DECLARE @sn INT
SET @sn = 100
WHILE (@sn<110)
BEGIN
    PRINT 'So nguyen: ' + CONVERT(char(3),@sn)
    SET @sn = @sn + 1
END
```

- Bài tập
  - Tính tổng các số chẵn từ 1--> 100

Slide 79

2.3. Sử dụng biến kiểu dữ liệu CURSOR

- Định nghĩa
  - DECLARE Ten_Cursor CURSOR
    [Kiểu đọc, cập nhật dữ liệu]
    FOR Câu_truy_vấn
  - Trong đó
    - Ten_Cursor: tên biến kiểu dữ liệu Cursor
    - Một số từ khóa: (Tự đọc)
- Sử dụng
  - OPEN Ten_Cursor
  - ….
  - CLOSE Ten_Cursor
- Hủy bỏ Cursor
  - DEALLOCATE Ten_Cursor

Slide 80

2.3. Sử dụng biến kiểu dữ liệu CURSOR

- Đọc và xử lý dữ liệu trong cursor
- Cú pháp

  FETCH [NEXT | PRIOR | FIRST | LAST | ABSOLUTE n | RELATIVE n]
  FROM Ten_Cursor
  [INTO Danh_sách_biến]
- Dùng biến hệ thống @@FETCH_STATUS để kiểm tra tình trạng đọc dữ liệu thành công hay thất bại, giá trị của nó = 0 là thành công.

2.3. Sử dụng biến kiểu dữ liệu CURSOR

- Để làm việc với một cursor ta làm theo các bước
  - Dùng câu lệnh DECLARE CURSOR để khai báo một cursor. Khi khai báo ta cũng phải cho biết câu lệnh SELECT sẽ được thực hiện để lấy dữ liệu.
  - Dùng câu lệnh OPEN để đưa data lên memory (populate data). Đây chính là lúc thực hiện câu lệnh SELECT vốn được khai báo ở trên.
  - Dùng câu lệnh FETCH để lấy từng hàng dữ liệu từ cursor.
  - Khi duyệt từng bản ghi ta có thể UPDATE hay DELETE tùy theo nhu cầu.
  - Dùng câu lệnh CLOSE để đóng cursor. Một số tài nguyên (memory resource) sẽ được giải phóng nhưng cursor vẫn còn được khai báo và có thể OPEN trở lại.
  - Dùng câu lệnh DEALLOCATE để giải phóng hoàn toàn các tài nguyên dành cho cursor (kể cả tên của cursor).

2.3. Sử dụng biến kiểu dữ liệu CURSOR

- Ví dụ: Xem các mặt hàng có Tên bắt đầu là B.

```
declare cur_vt cursor
keyset
for
   select * from tblvattu
   where tenvtu like 'B%'
   order by mavtu
open cur_vt
while 0=0
begin
   fetch next from cur_vt INTO ….
   if @@fetch_status<>0 break
   Print…
   end
close cur_vt
deallocate cur_vt
```

2.4: Thủ tục nội tại

- Nội dung
  - 2.4.1. Khái niệm
  - 2.4.2. Quản lý thủ tục nội tại

Slide 84

2.4.1. Khái niệm

- Thủ tục nội tại (store procedure) là một tập hợp chứa các dòng lệnh, các biến, các cấu trúc dùng để thực hiện một hành động nào đó.
- Nội dung của thủ tục nội tại sẽ được lưu trữ tại CSDL của MS SQL SERVER
- Phạm vi hoạt động của thủ tục nội tại do người dùng tạo ra chỉ có tính cục bộ bên trong một CSDL mà nó lưu trữ.
- Các NNLT khác có thể gọi thủ tục nội tại một cách dễ dàng.
- Thực thi nhanh vì nội dung bên trong thủ tục đã được phân tích cú pháp khi chúng được tạo mới.

Slide 85

2.4.1. Khái niệm

1. Tạo mới thủ tục nội tại
2. Gọi thực hiện thủ tục nội tại
3. Hủy bỏ thủ tục nội tại
4. Ví dụ

Slide 86

1. Tạo mới thủ tục nội tại

- Tạo mới thủ tục nội tại bằng lệnh CREATE PROCEDURE
- Cú pháp:
  - CREATE PROCEDURE Ten_Thu_Tuc
    @tên_tham_số kiểu_dữ_liệu *loại*
    AS
    [DECLARE Bien_cuc_bo]
    Các_lệnh

Slide 87

1. Tạo mới thủ tục nội tại

- Trong đó
  - Ten_thu_tuc: phải là duy nhất trong CSDL
  - Bien_cuc_bo: là các biến được sử dụng tính toán tạm thời
  - Các_lệnh: Là các lệnh thực hiện
  - Loại:
    - input (không cần ghi)
    - output

Slide 88

1. Tạo mới thủ tục nội tại

- Ví dụ: Xây dựng thủ tục tính tổng trị giá của một sodh

- Input: số sodh
- Output: Tổng tiền

```
Create proc tt_sodh @sdh char(4), @tt int output
as
Select @tt = sum(vt.dongia*ct.sldat)
    from TBLctdondh ct inner join TBLvattu vt
    on ct.mavtu=vt.mavtu
Where ct.sodh=@sdh
Group by ct.sodh
```

Slide 89

2. Gọi thực hiện thủ tục nội tại

- Chúng ta chỉ có thể thực hiện thủ tục nội tại EXECUTE trong tiện ích Query Analyzer, hay trong NNLT.
- Cú pháp:
  - EXEC[UTE] Ten_thu_tuc giá_trị | @biến [output]
- Ví dụ

```
declare @ttien int
exec tt_sodh '2',@ttien output
print 'Tong tien:' + str(@ttien,7)
```

Slide 90

3. Hủy bỏ thủ tục nội tại

- Khi không cần sử dụng nữa thì chúng ta có thể hủy bỏ nó ra khỏi CSDL.
- Dùng lệnh
  - DROP PROCEDURE Ten_thu_tuc
- Ví dụ:
  - DROP PROCEDURE tt_sohd
- Sửa đổi thủ tục
  - ALTER PROCEDURE Ten_thu_tuc

Slide 91

4. Ví dụ

- Ví dụ 1: Viết thủ tục tính tổng số lượng đặt hàng của 1 vật tư trong tháng năm nào đó

```
Create proc tt2 @tenvattu nvarchar(30),@ngay char(7),@Tx int
                                                    output
as

Select vt.tenvtu, sum(Sldat) as Tongxuat into #bt
    From tblCTDONDH ctdh
            Inner join tbldondh dh on ctdh.sodh=dh.sodh
            inner join tblvattu vt on vt.mavtu=ctdh.mavtu
where       convert(char(7),    ngaydh,21)=@ngay    and
    vt.tenvtu=@tenvattu
    Group by vt.tenvtu
select @tx=Tongxuat from #bt
Drop table #bt
```

Slide 92

4. Ví dụ

- Lời gọi:

```
declare @tong int
exec tt2 N'Bánh mỳ', N'2010-02',@tong output
print @tong
```


4. Ví dụ

- Ví dụ 2: Xây dựng thủ tục trả về danh sách các tên vật tư có số lượng đặt hàng nhiều nhất trong năm tháng nào đó

```
Alter proc Max_vt  @ngay char(7)
AS
-- Tạo bảng tạm lưu Tên vật tư với Tổng số lượng đặt hàng giảm dần của
-- năm tháng  yêu cầu
SELECT      VT.TenVtu, sum(CTDH.SLDat) as Tongsld into #bt
FROM        tblCTDONDH ctdh
      INNER JOIN  tblVATTU vt ON CTDH.MaVTu = VT.MaVTu
      INNER JOIN  tblDONDH dh ON CTDH.SoDH = DH.SoDH
where convert(char(7),ngaydh,21)=@ngay
group by VT.TenVtu
order by sum(CTDH.SLDat) Desc
```


4. Ví dụ

```
-- Hiện các Tên vật tư có số lượng đặt = Tổng số đặt hàng lớn nhất
Select   tenvtu,tongsld from #bt
where tongsld=( select max(tongsld) from #bt)
--Xoá bảng tạm
Drop table #bt
```

- Lời gọi:

```
Exec Max_vt N'2010-02'
```

4. Ví dụ

- Bài tập: Viết thủ tục sinh ra mã vật tư tự động theo quy tắc:
  - Mã vật tư có dạng: VT01
  - 'VT' : quy định (luôn có)
  - 01 : là số
  - VD:
    - Hiện tại Vật tư có mã cao nhất là VT04
    - Thì sinh mã mới là VT05

Slide 96

4. Ví dụ

- Bài tập:

```
Alter procedure sinhkhoa @s char(10) output
as
begin
    declare @max int
    select
        @max = max(cast(substring(Mavtu,3,2) as
    int)) + 1 from tblvattu
    set @s = '0000' + rtrim(cast(@max as char(2)))
    set @s = 'VT' + right(rtrim(@s),2)
end
```

Slide 97

2.4.2. Một số vấn đề khác

1. Sử dụng RETURN trong thủ tục
2. Sử dụng bảng tạm
3. Tham số kiểu CURSOR bên trong thủ tục
4. Thủ tục cập nhật bảng dữ liệu
5. Thủ tục hiện thị bảng dữ liệu

Slide 98

1. Sử dụng RETURN trong thủ tục

- RETURN dùng để thoát ra khỏi thủ tục trong trường hợp dữ liệu không hợp lệ.
- Nó cho phép trả về 1 số nguyên (giống hàm)
- RETURN không có giá trị chỉ định ngầm định 0
- Trong trường hợp thủ tục dùng Return, khi gọi thủ tục ta sử dụng 1 biến cục bộ để đón nhận giá trị trả về của thủ tục đó.

Exec @Biến = Tên_thủ_tục[các_tham_số]

Slide 99

1. Sử dụng RETURN trong thủ tục

- Ví dụ:

```
alter proc tttt @ten nvarchar(30), @tt int output
as

If not exists ( SELECT * FROM tblnhacc WHERE
                Tennhacc =@ten  )
        return 1
else
If not exists (SELECT * FROM tblctdondh ct
        INNER JOIN tbldondh dh ON ct.sodh = dh.sodh
        INNER JOIN tblvattu vt ON ct.mavtu = vt.mavtu
        INNER JOIN tblnhacc nc ON dh.manhacc = nc.manhacc
GROUP BY nc.tennhacc
HAVING     (nc.tennhacc = @ten)     )
```

Slide 100

1. Sử dụng RETURN trong thủ tục

```
return 2
else
SELECT  @tt=SUM(ct.sldat * vt.Dongia)
FROM tblctdondh ct
  INNER JOIN tbldondh dh ON ct.sodh = dh.sodh
  INNER JOIN tblvattu vt ON ct.mavtu = vt.mavtu
  INNER JOIN tblnhacc nc ON dh.manhacc = nc.manhacc
  GROUP BY nc.Tennhacc
  HAVING     (nc.Tennhacc = @ten)
return
```

Slide 101

1. Sử dụng RETURN trong thủ tục

- Lời gọi

```
declare @tien int, @rt int
exec @rt =tttt N'Cơ Phát', @tien output
If @rt =1
print 'khong co nhacc nay'
else
If @rt = 2
print 'nhacc nay chua co gi'
else
print 'Tổng tiền :' + cast(@tien as char(10))
```

Slide 102

2. Sử dụng bảng tạm trong thủ tục

- Cú pháp:
  Select Ds_cột INTO #Tên_btạm
  From tên_bảng_dữliệu
- Nếu sử dụng bảng tạm ta nhớ phải xóa nó bằng DROP TABLE trước khi kết thúc thủ tục

Ví dụ: Hiện vật tư nào có tổng số lượng nhập hàng lớn nhất đến tháng năm nào đó.

Slide 103

2. Sử dụng bảng tạm trong thủ tục

- Create proc **tt4 @stenvtu nvarchar(30) output**
  as
- select **TenVTu, sum(SLDat) as tsl into #bt**
  from **tblVATTU vt inner join tblCTDonDH ct on vt.MaVTu =ct.MaVTu**
  group **by TenVTu**
- select **tenvtu from #bt**
  where **tsl =(select MAX(tsl) from #bt)**
- drop **table #bt**

Slide 104

2. Sử dụng bảng tạm trong thủ tục

- Gọi thực hiện thủ tục:
- set **nocount on**
- exec **tt4**

Slide 105

3. Tham số kiểu Cursor

- Dùng trong trường hợp tham số trả về là các dòng giá trị
- Các hành động liên quan đến các biến dữ liệu kiểu CURSOR được chia thành 2 phần: bên trong thủ tục và bên ngoài thủ tục.
- Bên trong khai báo thủ tục gồm: KHAIBAO, định nghĩa biến kiểu cursor và mở Cursor.
- Bên ngoài thủ tục gồm: đọc từng dòng dữ liệu trong Cursor và đóng nó.

Slide 106

3. Tham số kiểu Cursor

Ví dụ: Hiện tên vật tư, số lượng xuất trong năm tháng nào đó

- ALTER **proc tt5 @snt char(7), @cur_vt CURSOR VARYING output**
- as
- set **@cur_vt =CURSOR**
- KEYSET
- FOR
- select **TenVTu, sum(SLXuat) as tslx**
- from **tblVATTU vt inner join tblCTPXuat ct on ct.MaVTu =vt.MaVTu**
- **inner join tblPXuat px on px.SoPX = ct.SoPX**
- **where convert(char(7),Ngayxuat,21)=@snt**
- group **by TenVTu, convert(char(7),Ngayxuat,21)**
- OPEN **@CUR_VT**

Slide 107

3. Tham số kiểu Cursor

Lời gọi:

- declare **@scur_vt cursor**
- declare **@stenvtu nvarchar(30), @stsl int**
- exec **tt5 '2013-04',@scur_vt output**
- while **0=0**
- begin
- **fetch next from @scur_vt into @stenvtu, @stsl**
- **if @@FETCH_STATUS <>0 break**
- **print @stenvtu+ cast(@stsl as char(10))**
- **end**
- close **@scur_vt**
- deallocate **@scur_vt**

Bài tập:

1. Hiện tổng số lượng bán các vật tư trong năm tháng nào đó.
2. Hiện tổng số lượng đặt hàng các vật tư trong năm tháng nào đó.
3. Hiện tổng số lượng đặt hàng các vật tư của 1 nhà cc nào đó.

Slide 108

3. Tham số kiểu Cursor

Ví dụ: Hiện tên vật tư, số lượng xuất trong năm tháng nào đó, nếu không có hiện dòng thông báo.

- ALTER **proc tt5 @snt char(4), @cur_vt CURSOR VARYING output**
- as
- if **not exists (**select **TenVTu, sum(SLXuat) as tslx**
- from **tblVATTU vt inner join tblCTPXuat ct on ct.MaVTu =vt.MaVTu**
- **inner join tblPXuat px on px.SoPX = ct.SoPX**
- **where convert(char(7),Ngayxuat,21)=@snt**
- group **by TenVTu, convert(char(7),Ngayxuat,21)**
- **)**
- return **1**
- else

Slide 109

3. Tham số kiểu Cursor

- begin
- set **@cur_vt =CURSOR**
- KEYSET
- FOR
- select **TenVTu, sum(SLXuat) as tslx**
- from **tblVATTU vt inner join tblCTPXuat ct on ct.MaVTu =vt.MaVTu**
- **inner join tblPXuat px on px.SoPX = ct.SoPX**
- **where convert(char(7),Ngayxuat,21)=@snt**
- group **by TenVTu, convert(char(7),Ngayxuat,21)**
- OPEN **@CUR_VT**
- end

Slide 110

3. Tham số kiểu Cursor

```
-------lời gọi
declare @scur_vt cursor
declare @stenvtu nvarchar(30), @stsl int,@sb int
exec @sb=tt5 '2014-05',@scur_vt output
if @sb=1
    print 'khong co'
    else
    begin
while 0=0
begin
    fetch next from @scur_vt into @stenvtu, @stsl
    if @@FETCH_STATUS <>0 break
    print @stenvtu+ cast(@stsl as char(10))
    end
close @scur_vt
deallocate @scur_vt
end
```


4. Thủ tục cập nhật bảng dữ liệu

- Ví dụ:

Tblcanbo (MãCB, MãDV, TênCB, Giới tính, Sốnămcôngtác, Địachỉ, HệsốlươngHệsốPC, Lương).

Yêu cầu:

- MãCB : không trùng
- MãDV: phải có trong TblDonvi
- Giới tính: chỉ nhận 0 hoặc 1
- Hệ số lương : between 0 and 8
- Hệ số PC: between 0 and HSLương
- Lương = (HSPC + HSL) * 730

Bài tập:

1. Viết thủ tục nhập dữ liệu vào cho bảng tbldondh, tblctdondh
2. Bảng tblpnhap, tblctpnhap với yêu cầu: tổng slnhap của 1 vật tư , 1 sodh không lớn hơn sldat của vật tư, sodh đó.

4. Thủ tục cập nhật bảng dữ liệu

```
alter proc tt_cn
    @smacb char(4), @sMadv char(4), @sTencb nvarchar(20), @sGioitinh int,
    @sSnct int, @sDiachi nvarchar(20), @sHesoluong float, @sHesoPC float
as
--Kiểm tra mãcb
if exists(select macb from tblcanbo where macb = @smacb)
begin
print 'da co ma nay'
return
End
--Kiểm tra Mã DV
if not exists (select madv from tbldonvi where
    madv =@smadv)
begin
print 'Chưa có madv nay'
return
end
```

4. Thủ tục cập nhật bảng dữ liệu

```
--Kiểm tra gioitinh
if (@sgioitinh < 0) or (@sgioitinh>1)
begin
print 'gioi tinh chi nhan 0 hoac 1'
Return
end

--Kiểm tra Hệ số lương
If @sHesoluong not between 0 and 8
begin
Print 'He so luong nam ngoai'
Return
end
```

Slide 114

4. Thủ tục cập nhật bảng dữ liệu

```
--Kiểm tra Hệ số PC
if not ((@sHesoPC >0) and (@sHesoPC <@sHesoluong))
begin
print ' Heso PC nam ngoai'
return
end
--Tính lương
declare @sLuong int
Set @sluong =(@shesoPC +@sHesoluong)*730
--Thêm vào bảng
Insert into tblcanbo
Values
      (@smacb,      @stencb,@sDiachi,@smadv,@sHesoluong,    @sgioitinh,
      @sSnct,   @sHesopc, @sluong)

--Gọi thủ tục
Exec tt_cn 'CB11','DV01','Le Thi C',0,13, 'Hưng phúc',3.1,5
```

Slide 115

5. Thủ tục hiện thị dữ liệu

- Với các báo cáo ta có thể lấy dữ liệu nguồn từ:
  - Bảng dữ liệu hoặc bảng ảo
  - Câu lệnh Select chọn trực tiếp
  - Đối tượng thủ tục nội tại
- Nếu chúng ta lấy từ thủ tục nó cho phép chứa đựng các biến cục bộ để tính toán dữ liệu phức tạp.

Slide 116

5. Thủ tục hiện thị dữ liệu

- Ví dụ: Xây dựng thủ tục hiện thị dữ liệu cho báo cáo đơn đặt hàng. Tham số vào soDH, nếu gọi thủ tục mà không truyền tham số thì xem như hiện thị toàn bộ các Sodh có trong bảng tblDondh.

```
create proc tt_bc @sSohd char(4) = NULL
as
If @sSohd is Null
    SELECT dh.sodh, tenvtu, sldat, Tennhacc
    FROM tbldondh dh
        INNER JOIN tblctdondh ct ON dh.sodh = ct.sodh
        INNER JOIN tblnhacc nc ON dh.manhacc=nc.manhacc
        INNER JOIN tblvattu vt ON ct.mavtu = vt.mavtu
    ORDER BY dh.sodh, vt.tenvtu
```


5. Thủ tục hiện thị dữ liệu

```
else
    SELECT dh.sodh, tenvtu, sldat, Tennhacc
    FROM tbldondh dh
        INNER JOIN tblctdondh ct ON dh.sodh = ct.sodh
        INNER JOIN tblnhacc nc ON dh.manhacc=nc.manhacc
        INNER JOIN tblvattu vt ON ct.mavtu = vt.mavtu
        Where dh.sodh = @sSohd
    ORDER BY vt.tenvtu
```

- Yêu cầu : thêm Sodh không có trong tbldondh

Bài tập

1.

5. Thủ tục hiện thị dữ liệu

- Bài tập
  - 1. Xây dựng thủ tục hiện thị dữ liệu cho báo cáo xuất hàng: tên vật tư, số lượng xuất, đơn giá, tổng tiền. Tham số vào soPX, nếu gọi thủ tục mà không truyền tham số thì xem như hiện thị toàn bộ các SoPX có trong bảng tblpxuat.
  - 2. Xây dựng thủ tục hiện thị dữ liệu cho báo cáo gồm tên vật tư, số lượng đặt, tổng tiền. Tham số vào tên nhà cung cấp, nếu gọi thủ tục mà không truyền tham số thì xem như hiện thị toàn bộ các nhà cc.

2.5. Hàm do người dùng định nghĩa

- Nội dung
  - 2.5.1. Hàm do người dùng định nghĩa
  - 2.5.2. Hàm trả về giá trị vô hướng
  - 2.5.3. Hàm trả về dạng bảng Inline Table
  - 2.5.4. Hàm trả về dạng bảng Multi Statement

2.5.1. Hàm do người dùng định nghĩa

- Hàm do người dùng tự tạo (User Defined Functions - UDFs) là một đối tượng mới được bổ sung từ phiên bản MS SQL SERVER 2000.
- UDFs là sự kết hợp của hai đối tượng View và Store Procedure, và bổ sung thêm những tính năng mới mà View và Store Procedure chưa có.

2.5.2. Hàm trả về giá trị vô hướng

- Cú pháp

```
Create function Tên_hàm (tham_số_vào)
      returns kiểu_dl_trả_về
As
Begin
    Các_câu_lệnh
Return @Biến_trả_về
End
```

- Ví dụ 1: Viết hàm sinh mã VT tự động.(Hiện tại mã vật tư lớn nhất trong tblvattu là vt07, đưa ra mã vt08)
- Lời gọi hàm:
- Print dbo.Tên_hàm (cac_tham_số_thực_sự)

2.5.2. Hàm trả về giá trị vô hướng

```
Create function sinhma()
Returns char(10)
As
Begin
  declare @s char(10),@max int
  select @max=max(cast(substring(mavtu,3,2) as int))+1 from tblvattu
 set @s='00'+ltrim(rtrim(cast(@max as char(2))))
 Set @s='vt'+substring(@s,len(@s)-1,2)
Return @s
End

--Lời gọi
print dbo.sinhma()
```

Slide 123

2.5.2. Hàm trả về giá trị vô hướng

- Bài tập:

1. Viết hàm tham số vào là mã hàng hoá trả về đơn giá của hàng hoá này.
2. Viết hàm tham số vào là tên hàng hoá trả về đơn giá của hàng hoá này.
3. Viết hàm tham số vào là nhà cung cấp trả về tổng tiền phải trả cho nhà cc này.
4. Viết hàm tham số vào là tên nhà cung cấp và tên hàng hoá trả về tổng tiền phải trả cho nhà cc này với tên hàng hoá trên.
5. Viết hàm tham số vào là số đặt hàng trả về tổng tiền của số đặt hàng này.

Slide 124

2.5.2. Hàm trả về giá trị vô hướng

- Bài tập:

6. Viết hàm tham số vào là số phiếu xuất trả về tổng tiền của nó.
7. Viết hàm tham số vào là tên hàng hoá trả về tổng số lượng nhập hàng của hàng hoá này.
8. Viết hàm tham số vào là tên hàng hoá trả về tổng số lượng xuất hàng của hàng hoá này.
9. Viết hàm tham số vào là tên hàng hoá trả về tổng tiền nhập hàng của hàng hoá này.
10. Viết hàm tham số vào là tên hàng hoá trả về tổng tiền bán hàng của hàng hoá này.

Slide 125

2.5.3. Hàm trả về dạng bảng Inline Table

- Cú pháp:

  Create function tên_hàm(tên_các_tham_số)
  Returns Table
  As
  Return (Câu lệnh Select)

- Lời gọi:

  Select * from dbo.tên_hàm(tên_tham_số)

Slide 126

2.5.3. Hàm trả về dạng bảng Inline Table

- Ví dụ: viết hàm hiển thị các nhà cung cấp đã cc hàng hoá A
- create **function tt_4(@sTenvtu nvarchar(30))**
- returns **Table**
- as
- return
- (select **distinct TenNhacc, DiaChi, DienThoai**
- from **tblNhaCC nc inner join tblDonDH dh on dh.Manhacc=nc.Manhacc**
- **inner join tblPNhap pn on pn.SoDH =dh.SoDH**
- **inner join tblCTPNhap ct on ct.SoPN=pn.SoPN**
- **inner join tblVATTU vt on vt.MaVTu =ct.MaVtu**
- where **TenVTu =@sTenvtu)**

------Lời gọi:

- select *** from dbo.tt_4(N'Xi măng')**
- select *** from dbo.tt_4(N'Gạch')**

Slide 127

2.5.3. Hàm trả về dạng bảng Inline Table

- Bài tập:

QLBH:

1. Tạo hàm tham số vào là nhàcc, ra là tên các hàng hoá, đơn giá, Tổng số lượng đặt hàng, Tiền đã cung cấp
2. Tạo hàm hiển thị các tên hàng hoá, số lượng đặt mua, đơn giá, Tiền của số đặt hàng s.
3. Tạo hàm tham số vào là nhàcc, ra là tên các hàng hoá, đơn giá, Tổng số lượng nhập hàng, Tiền

Slide 128

2.5.4. Hàm trả về dạng bảng Multi Statement

- Nó cho phép thực hiện các câu lệnh select phức tạp,các lệnh update, insert into,...Hàm dạng này luôn trả về một biến table
- Cú pháp:

```
Create Function Tên_hàm (DS_tham_số)
Returns @ Tên_Table TABLE
( Tên_cột_1 kiểu _dl,
  Tên_cột _2 kiểu dl,...)
As
Begin
  Các _câu_ lệnh
  Return
end
```

Slide 129

2.5.4. Hàm trả về dạng bảng Multi Statement

Ví dụ: Tạo hàm đầu vào là sodh, ra là table chứa tên hàng hoá, số lượng đặt, đơn giá, tổng tiền của sodh này

```
Create function h_sodh(@ssodh char(4))
Returns @hh table
    ( tenvtu nvarchar(30),
     dongia int,
     sldat int,
    tongtien  int)
As
begin
```

Slide 130

2.5.4. Hàm trả về dạng bảng Multi Statement

```
Insert into @hh (tenvtu, dongia,sldat)
Select tenvtu, dongia, sldat
From tbldondh dh
 Inner join tblctdondh ct on dh.sodh=ct.sodh
 inner join tblvattu vt on ct.mavtu=vt.mavtu
Where   (dh.sodh=@ssodh)

Update @hh
Set tongtien=dongia*sldat
Return
end
----------
--Lời gọi
Select * from h_sodh(1)
```

Slide 131

2.5.4. Hàm trả về dạng bảng Multi Statement

- Bài tập:

1. VD trên: thêm cột Giảm giá= nếu tổng tiền <= A, để nguyên không giảm, nếu từ A đến B giảm 5%, lớn hơn B giảm =10%, Cột tiền phải trả = tổng tiền-giảm.

2. Tạo hàm tạo table gồm: tên nhà cc, tổng tiền,cột giảm giá: giảm 10% nếu tổng tiền > A, nếu từ A đến B giảm 5%, ít hơn B giảm =0, Cột tiền phải trả = tổng tiền-giảm.

Slide 132

2.6. TRIGGER

- Nội dung
  - 2.6.1. Khái quát về Trigger
  - 2.6.2. Lập trình Trigger
  - 2.6.3. Các hàm và lệnh hệ thống thường dùng trong Trigger
  - 2.6.4. Một số vấn đề khác
  - 2.6.5. Trigger kiểm tra ràng buộc dữ liệu

Slide 133

2.6.1. Khái quát về Trigger

- Trigger được dùng trong các trường hợp sau: Kiểm tra các ràng buộc toàn vẹn dữ liệu phức tạp và các câu thông báo thích hợp theo ý mình khi các ràng buộc dữ liệu sai.
- Các ràng buộc khóa chính, khóa ngoại, kiểm tra miền giá trị,… ta nên tạo thời điểm tạo table. Các ràng buộc toàn vẹn xảy ra khi thực thi các thao tác Insert, Delete, Update ta nên dùng Trigger.

Slide 134

2.6.1. Khái quát về Trigger

- Cơ chế hoạt động của trigger: Khi thêm mới mẫu tin vào table nó sẽ kích hoạt 1 table lưu trữ dữ liệu mẫu tin mới này với tên INSERTED. Khi xóa mẫu tin nó sẽ kích hoạt table với tên DELETED. Khi sửa là phối hợp 2 lệnh delete và insert, do thế bảng DELETED chứa dòng thông tin cũ, bảng INSERTED chứa dòng dữ liệu mới.
- Trigger có các loại sau: INSTEAD OF và AFTER (FOR).

Slide 135

2.6.1. Khái quát về Trigger

- INSTEAD OF: bỏ qua hành động kích hoạt trigger (thao tác insert, delete , update khi gọi sẽ không được thực hiện trên table), thay vào đó nó sẽ thực hiện các câu lệnh bên trong trigger. Nó bỏ qua hành động tác động đến CSDL nhưng việc lưu trữ dữ liệu trên các bảng Inserted và Deleted vẫn được thực hiện. Instead of có thể được định nghĩa trên table hoặc view.
- AFTER(hoặc FOR): có vai trò bổ sung thêm hành động kích hoạt trigger. Các câu lệnh bên trong trigger chỉ được thi hành sau khi các hành động kích hoạt đã được thực hiện rồi. Trigger loại After chỉ được định nghĩa duy nhất trên Table.

Slide 136

2.6.1. Khái quát về Trigger

- Trên 1 Table, nếu ta định nghĩa cả 2 loại trigger và các constraint, thứ tự thi hành sẽ là trigger INSTEAD OF, các constraint được xử lý và sau cùng là trigger AFTER.

  Nếu các constraint bị vi phạm (tính toàn vẹn dữ liệu), các hành động của trigger Instead of sẽ được quay lui, trigger After sẽ không được thi hành.

Slide 137

2.6.2. Lập trình Trigger

- Table giả: Inserted

```
Create trigger them_nhacc
on tblnhacc
after insert
as
    select * from Inserted
```

- Lời gọi:

```
insert into tblnhacc
Values('07','XNK ggh','gfhf','096432')
```

Slide 138

2.6.2. Lập trình Trigger

- Table giả: Deleted

```
create trigger sua_nhacc on tblnhacc
after update
as
select * from inserted
select * from Deleted
```

- Lời gọi:

```
Update tblnhacc
set tennhacc = 'vghvjghk'
 where manhacc ='07'
```

Slide 139

2.6.3. Các hàm và lệnh hệ thống

- Hàm @@Rowcount: trả về số dòng bị ảnh hưởng bởi câu lệnh T-SQL ngay trước đó có trigger.
- Lênh Return: chấm dứt thi hành trigger.
- Lệnh RAISERROR: để hiện dòng thông báo.
- Lệnh ROLLBACK TRANSACTION : quay lui toàn bộ xử lý.

Slide 140

2.6.4. Một số vấn đề khác

- Tạo Trigger bằng Enterprise Manager: nháy phải chuột vào table cần tạo, All Task, Manager Trigger…
- Sửa đổi nội dung trigger: Alter trigger …
- Xóa: Drop trigger Tên_trigger
- Làm cho trigger mất hiệu lực:
  Alter table tên_table DISABLE TRIGGER tên_trigger
- Làm cho trigger có hiệu lực:
  Alter table tên_table ENABLE TRIGGER tên_trigger

Slide 141

2.6.5. Trigger kiểm tra ràng buộc dữ liệu

- Thêm mới mẫu tin
- Ví dụ: Xây dựng trigger trong tblpnhap để kiểm tra khi người dùng thêm mới mẫu tin.
  - Khóa ngoại: sodh phải tồn tại trong tbldondh
  - Miền giá trị: ngày nhập phải sau ngaydh trong bảng tbldondh

Bài tập:

1. Xây dựng trigger trong tblctpnhap: SOPN có trong tblpnhap, Mavtu có trong tblvattu, Slnhap: tổng slnhap của vattu này trong Sodh đó bé hơn hoặc bằng sldat vattu đó của sodh này trong bảng tblctdondh
2. Xây dựng trigger trong tblctpxuat: Sopx có trong tblpxuat, Mavtu có trong tblvattu, Slxuat bé hơn hoặc bằng số lượng tồn kho = tổng số lượng nhập – tổng số lượng xuất

Slide 142

2.6.5. Trigger kiểm tra ràng buộc dữ liệu

- Hủy bỏ dữ liệu
- Ví dụ: Xóa 1 Sodh trong tbldondh. Nếu sodh này đã có trong bảng tblpnhap thì không xóa được. Nếu chưa có thì xóa nó cùng bản ghi có sodh đó trong tblctdondh.

Slide 143

2.6.5. Trigger kiểm tra ràng buộc dữ liệu

- Sửa đổi dữ liệu
- Cú pháp : Update(tên_cột) (bt_đk)
  - Tên_cột: tên cột ta muốn kiểm tra dữ liệu tại đó có bị sửa trong trigger hay không?
  - Bt_đk: True nếu cột bị sửa ngược lại False.
  - Sửa thực chất là 2 hành động đi kèm: xóa dữ liệu cũ hiện có và thêm dữ liệu mới đã được sửa đổi. Do đó trong trigger bảng Inserted sẽ chứa dữ liệu mới sau sửa đổi và bảng Deleted sẽ chứa đựng dữ liệu cũ trước sửa đổi.

Slide 144

2.6.5. Trigger kiểm tra ràng buộc dữ liệu

- Ví dụ: Sửa thông tin bảng tbldondh thõa mãn:
  - Không cho phép sửa dữ liệu cột sodh
  - Sửa manhacc phải tồn tại trong tblnhacc
  - Cột ngaydh phải trước ngày nhập hàng trong tblpnhap.

Slide 145