THỦ TỤC, HÀM

BIẾN CỤC BỘ

- Biến cục bộ: là biến dùng để lưu trữ các giá trị tạm thời trong quá trình tính toán các xử lý. Sau khi thoát khỏi chương trình thì giá trị các biến sẽ không còn trong bộ nhớ.
- Mỗi biến cần có tên rõ ràng và duy nhất trong một phạm vi là một lô (batch) hoặc một thủ tục nội tại (stored procedure)

KHAI BÁO BIẾN

- Khai báo biến phải được thực hiện trước khi sử dụng biến
- Cú pháp:

DECLARE @Tên_biến Kiểu_dữ_liệu [,..]

Tên biến: luôn bắt đầu bằng kí tự @, là duy nhất trong một phạm vi hoạt động

Kiểu dữ liệu: các kiểu dữ liệu cơ bản của SQL Server hoặc các kiểu do người dùng tự định nghĩa

Không thể gán giá trị khởi tạo cho biến lúc khai báo

GÁN GIÁ TRỊ CHO BIẾN

Gán giá trị cho biến

- Sử dụng lệnh SET
- Sử dụng lệnh SELECT cùng với ghép gán =

VD: gán giá trị là ngày 15/1/2019 cho biến Ngayxh

```
DECLARE @Ngayxh DATETIME
SET @Ngayxh='2019-1-15'
```

XEM GIÁ TRỊ HIỆN HÀNH CỦA BIẾN

Cú pháp:

PRINT @Tên_biến| Biểu_thức_ chuỗi

Tên biến: tên của biến đã được khai báo. Nếu kiểu dữ liệu của biến không phải kiểu chuỗi thì phải sử dụng hàm CAST hoặc CONVERT để chuyển về kiểu dữ liệu chuỗi

Biểu thức chuỗi: là một biểu thức chuỗi văn bản cần in ra, độ dài tối đa 8000 kí tự

XEM GIÁ TRỊ HIỆN HÀNH CỦA CHUỖI

Hàm CAST: cho phép chuyển đổi một biểu thức sang kiểu dữ liệu mong muốn

Cú pháp:

CAST(Biểu_thức AS Kiểu_dữ_liệu)

Biểu thức: là tên của một cột, một biến…cần chuyển sang dữ liệu kiểu mới

Kiểu dữ liệu: là kiểu dữ liệu mới muốn chuyển sang.

- Hàm CONVERT: cho phép chuyển đổi một biểu thức nào đó sang một kiểu dữ liệu bất kì theo một định dạng nào đó (đặc biệt đối với kiểu dữ liệu ngày)
- Cú pháp:

CONVERT(Kiểu_dữ_liệu,Biểu_thức [,định dạng])

Hiển thị dữ liệu biến tổng số lượng có của các mặt hàng do nhà cung cấp có mã là S001 cung cấp

```
DECLARE @TongSL INT
SELECT @TongSL= SUM(SoLuongCo)
FROM HANG
WHERE MaNCC LIKE ‘S001’
```

- PRINT ‘TONG SO LUONG CÓ LÀ’ + CAST(@TongSL AS CHAR(5))

Hoặc

- PRINT ‘TONG SO LUONG CÓ LÀ’ + CONVERT (CHAR(5), @TongSL)

BIẾN HỆ THỐNG

- Các biến hệ thống giúp kiểm tra kết quả thực hiện các lệnh trước đó.
- Biến hệ thống không cần khai báo, có giá trị do SQLServer cung cấp vì vậy không thể can thiệp để gán giá trị cho biến hệ thống.
- Luôn bắt đầu bằng @@

VD: Biến @@VERSION trả về phiên bản, ngày của sản phẩm SQL Server và loại CPU

PRINT @@VERSION

Các toán tử

- Toán tử số học: +, -, *,/,%
- Toán tử nối chuỗi: nối các chuỗi rời rạc thành chuỗi liên tục. Ký hiệu: +
- Toán tử so sánh: =,>,<,>=,<=,<>,!=, !>,!<
- Toán tử luận lý: AND, OR, NOT

Cấu trúc điều khiển

- Cấu trúc rẽ nhánh IF…ELSE
- Cấu trúc lặp WHILE

IF..ELSE

- Cú pháp:

IF biểu_thức_logic

Câu_lệnh 1| khối lệnh 1

[ELSE câu_lệnh 2| khối_lệnh 2]

Trong đó:

Biểu thức logic: là biểu thức so sánh

Câu lệnh 1| Khối lệnh 1: là các lệnh được thực thi khi biểu thức so sanh trả về giá trị True

Câu lệnh 2| Khối lệnh 2: là các lệnh được thực thi khi biểu thức so sanh trả về giá trị False

Chú ý: nếu có khối lệnh thì phải có từ khoá BEGIN END

- Tìm xem mặt hàng nào có số lượng có lớn hơn 300 thì in ra ngược lại thì in ra thông báo không có

```
IF (SELECT COUNT(*) FROM HANG WHERE SoLuongCo>300)>0
BEGIN
        PRINT N'DANH SÁCH CÁC MẶT HÀNG CÓ SỐ LƯỢNG CÓ      >300’
        SELECT * FROM HANG WHERE SoLuongCo>300
END
ELSE
        PRINT N'KHÔNG CÓ MẶT HÀNG NÀO CÓ SỐ LƯỢNG CÓ >300'
```

- Cú pháp IF kết hợp EXISTS dùng để kiểm tra sự tồn tại của các dòng dữ liệu bên trong bảng

IF EXISTS (Câu lệnh SELECT)

Câu_lệnh 1| Khối_lệnh 1

[ELSE Câu_lệnh 2| Khối_lệnh 2

- Tìm xem mặt hàng nào có số lượng có lớn hơn 300 thì in ra ngược lại thì in ra thông báo không có

```
IF EXISTS(SELECT MAHANG FROM HANG WHERE SoLuongCo>300)
BEGIN
        PRINT N'DANH SÁCH CÁC MẶT HÀNG CÓ SỐ LƯỢNG CÓ      >300’
        SELECT * FROM HANG WHERE SoLuongCo>300
END
ELSE
        PRINT N'KHÔNG CÓ MẶT HÀNG NÀO CÓ SỐ LƯỢNG CÓ >300'
```

WHILE

- Cấu trúc lặp được phép sử dụng bên trong một lô, các lệnh hoặc bên trong một thủ tục nội tại
- Cú pháp:

```
WHILE Biểu_thức_logic
BEGIN
        Các lệnh lặp
END
```

Sử dụng từ khoá BREAK để thoát khỏi vòng lặp WHILE sớm

Sử dụng từ khoá CONTINUE để khởi động lại vòng lặp WHILE từ đầu

- In ra 10 số nguyên dương bắt đầu từ 100

```
DECLARE @SoNguyen int
SET @SoNguyen=100
WHILE (@SoNguyen<110)
   BEGIN
        PRINT 'SO NGUYEN '+ Cast(@SONGUYEN as char(3))
       SET @SoNguyen=@SoNguyen+1
    END
```

In ra 10 số nguyên dương bắt đầu từ 100 nhưng không in số 108

```
DECLARE @SoNguyen int
SET @SoNguyen=100
WHILE (@SoNguyen<110)
   BEGIN
        SET @SoNguyen=@SoNguyen+1
        IF @SoNguyen=108
        CONTINUE
        PRINT 'SO NGUYEN'+ Cast(@SONGUYEN as char(3))
    END
PRINT 'KET THUC VONG LAP'
```

- In ra 10 số nguyên dương bắt đầu từ 100

```
DECLARE @SoNguyen int
SET @SoNguyen=100
WHILE (@SoNguyen<110)
   BEGIN
        PRINT 'SO NGUYEN '+ Cast(@SONGUYEN as char(3))
       SET @SoNguyen=@SoNguyen+1
    END
```

In ra 10 số nguyên dương bắt đầu từ 100 nhưng không in số 108

```
DECLARE @SoNguyen int
SET @SoNguyen=100
WHILE (@SoNguyen<110)
   BEGIN
        SET @SoNguyen=@SoNguyen+1
        IF @SoNguyen=108
        CONTINUE
        PRINT 'SO NGUYEN'+ Cast(@SONGUYEN as char(3))
    END
PRINT 'KET THUC VONG LAP'
```

HÀM, THỦ TỤC

1. Thủ tục nội tại
2. Hàm do người dùng định nghĩa

1. Thủ tục nội tại

- Thủ tục nội tại(stored procedure - SP) là các lệnh T-SQL được biên dịch từ trước và chứa trong CSDL SQL Server
- Nó có thể nhận các tham số vào, trả lại các kết quả ra, hoặc trả lại các thông báo chỉ trạng thái thành công hay thất bại

Thủ tục nội tại

Cú pháp :

```
CREATE PROC[EDURE] tên_thủ_tục
    [ @tên_tham_số
      kiểu_dữ_liệu [= giá_trị_mặc_định] OUTPUT]
      [,...]
  AS
      Khối_lệnh_SQL
```

Thủ tục nội tại

- Thủ tục nội tai có 4 cách trả lại dữ liệu
  - ✓ Tham số ra, có thể trả lại dữ liệu (số, ký tự . . .) hoặc một biến con trỏ (các con trỏ là các bộ kết quả mà nó có thể truy xuất một dòng tại mỗi thời điểm)
  - ✓ Trả lại các mã, luôn là một giá trị nguyên
  - ✓ Một bộ kết quả cho mỗi câu lệnh SELECT nằm trong thủ tục được lưu hoặc bất kỳ thủ tục được lưu nào được gọi bởi thủ tục được lưu hiện tại
  - ✓ Một con trỏ toàn cục có thể được tham chiếu bên ngoài thủ tục được lưu

Thủ tục nội tại

- **Thực thi thủ tục nội tại:**

```
EXEC[UTE] tên_thủ_tục
```

- **Cập nhật thủ tục:**

```
ALTER PROC[EDURE] tên_thủ_tục
    [ @tên_tham_số   kiểu_dữ_liệu] [= giá_trị_mặc_định]
    [OUTPUT]
    [,...,n]
AS
    Các câu lệnh SQL
```

Thủ tục nội tại

❖ **Xóa thủ tục :**

DROP PROC[EDURE] tên_thủ_tục

Thủ tục nội tại

Lợi ích khi sử dụng thủ tục được lưu

- Giảm truyền thông trên mạng giữa server/client :
  - Chỉ có lời gọi thủ tục được gửi qua mạng
- Bảo mật tốt hơn
  - Khi gọi thủ tục, chỉ thấy được lời gọi thực thi thủ tục. Vì vậy người dùng không thể nhìn thấy tên bảng và các đối tượng CSDL, họ không thể thêm được các câu lệnh T-SQL của mình hoặc truy xuất dữ liệu quan trọng

Thủ tục nội tại

- **Lợi ích khi sử dụng thủ tục được lưu . . .**
  - Tái sử dụng code:
    - Code cho bất kỳ hoạt động cơ sở dữ liệu lặp đi lặp lại nào là ứng cử viên hoàn hảo để đóng gói trong thủ tục (ví dụ, UPDATE dữ liệu trên một bảng)
  - Cải thiện hiệu suất:
    - Khi đã thực hiện lần đầu, thủ tục được lưu trữ trong bộ nhớ cache để nó có thể được sử dụng nhiều lần. SQL Server không phải biên dịch lại thủ tục mỗi khi chạy

Thủ tục nội tại

Câu lệnh SQL

First Time
- *Check syntax*
- *Compile*
- *Execute*
- *Return data*

Second Time
- *Check syntax*
- *Compile*
- *Execute*
- *Return data*

Thủ tục được lưu

Creating
- *Check syntax*
- *Compile*

First Time
- *Execute*
- *Return data*

Second Time
- *Execute*
- *Return data*

Thủ tục nội tại

Nhược điểm khi sử dụng thủ tục được lưu

- Máy chủ cơ sở dữ liệu phải làm việc với cường độ cao (cả bộ nhớ và bộ vi xử lý)
- Khó viết một thủ tục cho logic kinh doanh phức tạp
- Khó gỡ lỗi
- Không dễ viết và bảo trì

NHACC (**MaNCC**, TenNCC, DiaChi, DienThoai)

HANG (**MaHang**, TenHang, DonGia, SoLuongCo, *MaNCC*)

PHIEU_XUAT (**SoPhieu**, NgayXuat, MaCuaHang)

DONG_PHIEU_XUAT ***(SoPhieu, MaHang***, SoLuongXuat)

Đưa ra tên mặt hàng được xuất nhiều nhất

- Viết thủ tục đưa ra tên mặt hàng được xuất nhiều nhất (không có tham số đầu vào)

```
SELECT TenHang, SUM(soluongxuat) AS Tongsl INTO BANGTAM
FROM HANG, DONG_PHIEU_XUAT
WHERE HANG.MaHang=DONG_PHIEU_XUAT.MaHang
GROUP BY TenHang

SELECT TenHang,Tongsl
FROM BANGTAM
WHERE Tongsl =(SELECT MAX(Tongsl) FROM BANGTAM)
```

```
CREATE PROC     sp_TenHangBanNhieuNhat
AS
SELECT TenHang, SUM(soluongxuat) AS Tongsl INTO BANGTAM
FROM HANG, DONG_PHIEU_XUAT
WHERE HANG.MaHang=DONG_PHIEU_XUAT.MaHang
GROUP BY TenHang
SELECT TenHang,Tongsl
FROM BANGTAM
WHERE TONGSL=(SELECT MAX(TONGSL) FROM BANGTAM)
DROP TABLE BANGTAM
GO
-- Thực thi
EXEC sp_TenHangBanNhieuNhat
```

```
CREATE PROC sp_TenHangBanNhieuNhat
AS
        DECLARE @tenHang nvarchar(50), @SLMax int
        SELECT TenHang, SUM(soluongxuat) AS Tongsl INTO BANGTAM1
        FROM HANG, DONG_PHIEU_XUAT
        WHERE HANG.MaHang=DONG_PHIEU_XUAT.MaHang
        GROUP BY TenHang
        SELECT @tenHang =TENHANG
        FROM BANGTAM1
        WHERE Tongsl =(SELECT MAX(Tongsl) FROM BANGTAM1)
        SET @SLMax=(SELECT MAX(Tongsl) FROM BANGTAM1)
        PRINT 'MAT HANG ' @tenHang + ' BAN NHIEU NHAT'
        PRINT ' VOI SO LUONG LA ' + CAST(@SLMax As Char(10))
        DROP TABLE BANGTAM1
GO
```

- Viết thủ tục để tìm xem mỗi phiếu xuất đi bao nhiêu hàng với đầu vào là số phiếu

```
CREATE PROC Sp_SoluongHangTrongPhieu @sSoPhieu int
AS
SELECT COUNT(MaHang) As ‘Số lượng mặt hàng’
FROM DONG_PHIEU_XUAT
WHERE SoPhieu=@sSoPhieu
GO
```

```
CREATE PROC Sp_SoluongHangTrongPhieu @sSoPhieu int
AS
DECLARE @sSoMatHang int
SELECT @sSoMatHang= COUNT(MaHang) FROM
DONG_PHIEU_XUAT WHERE SoPhieu=@sSoPhieu
PRINT N'SỐ LƯỢNG HÀNG TRONG PHIẾU SỐ ' +
CAST(@sSoPhieu as char(4))+N'LÀ:' + CAST(@sSoMatHang as
char(4))
GO
-- Thực thi thủ tục
EXEC Sp_SoluongHangTrongPhieu 2
```

- Viết thủ tục có đầu ra là số lượng mặt hàng trên mỗi phiếu được nhập vào (Thủ tục có đầu vào và đầu ra)

```
CREATE PROC Sp_SLHangTrongPhieu
        @sSoPhieu int,
        @sSoMatHang int OUTPUT
AS
        SELECT @sSoMatHang= COUNT(MaHang) FROM
        DONG_PHIEU_XUAT WHERE SoPhieu=@sSoPhieu
GO
-- thực thi thủ tục
DECLARE @sSoMH int
EXEC Sp_SLHangTrongPhieu 2, @sSoMH OUTPUT
PRINT N'SỐ LƯỢNG MẶT HÀNG TRONG PHIẾU SỐ 2 LÀ:'
+CAST(@sSOMH AS CHAR (4))
```

- Viết thủ tục chèn thêm một nhà cung cấp mới, nếu mã nhà cung cấp đã có trong bảng nhà cung cấpthì đưa ra thông báo không thể chèn thêm, nếu mã nhà cung cấp chưa có thì chèn thêm dòng mới.
- NHACC (**MaNCC**, TenNCC, DiaChi, DienThoai)

```
CREATE PROC SP_INhaCC
@sMaNCC char(4),
@sTenNCC nvarchar(50),
@sDiaChi nvarchar(100),
@sDienThoai nvarchar(12)
AS
IF EXISTS(SELECT * FROM NHACC WHERE MANCC=@sMaNCC)
        PRINT N'NHÀ CUNG CẤP ĐÃ TỒN TẠI, KO CHÈN THÊM'
ELSE
BEGIN
        INSERT INTO NHACC
        VALUES(@sMaNCC,@sTenNCC,@sDiaChi,@sDienThoai)
        SELECT * FROM NHACC
END
GO
```

-- THỰC THI THỦ TỤC

EXEC SP_INhaCC 'S009','SANYO1','HAI PHONG','1234567'

EXEC SP_INhaCC 'S009','SANYO’,HA NOI'1234567'

Bài tập

1. Tạo thủ tục SP_ChiTietPhieuXuat để đưa ra số phiếu, mã hàng, số lượng xuất với mã cửa hàng sẽ được nhập khi gọi thủ tục
2. Sửa thủ tục SP_ChiTietPhieuXuat để đưa ra số phiếu, mã hàng, số lượng xuất với mã cửa hàng sẽ được nhập khi gọi thủ tục. Nếu nhập mã cửa hàng không tồn tại sẽ trả về giá trị 1
3. Tạo thủ tục SP_XoaMH để xoá một mặt hàng trong bảng Hàng với tham số đầu vào là mã hàng. Nếu mã hàng được nhập không có trong bảng Hàng thì đưa ra thông báo.

2. Hàm do người dùng định nghĩa (UDF)

- UDF là một chương trình con, nhận các tham số đầu vào, xử lý và trả về một giá trị kết quả → tương tự như hàm trong ngôn ngữ lập trình
- Giá trị trả về có thể là một giá trị vô hướng hoặc một bộ kết quả
- Hàm không thể thực hiện các thay đổi lâu dài trong SQL như các câu lệnh Insert, Update, Delete trên bảng thực sự

Hàm do người dùng định nghĩa (UDF)

Các loại hàm

- Hàm vô hướng: hàm trả về một giá trị
- Hàm trả lại giá trị là bảng (Table-valued function)
  - Hàm Inline Table-valued
  - Hàm Multi-statement Table-Valued

Hàm do người dùng định nghĩa (UDF)

Hàm vô hướng

```
CREATE FUNCTION [ tên_lược_đồ. ] tên_hàm
   ( [@tên_tham_số   kiểu_dữ_liệu [ ,...n ] ] )
RETURNS kiểu_dữ_liệu_giá_trị_trả_về
[ AS ]
BEGIN
    thân_hàm
    RETURN biểu_thức_vô_hướng
END
--Gọi hàm
SELECT DBO.tên_hàm( đối số 1, đối số 2,…)
```

- Viết hàm trả về số lượng phiếu xuất với đầu vào là ngày xuất
- PHIEU_XUAT (**SoPhieu**, NgayXuat, MaCuaHang)

```
CREATE FUNCTION F_SoLuongPX (@sNgayxuat datetime)
RETURNS INT
AS
BEGIN
        DECLARE @SoLuongPX INT
        SELECT @SoLuongPX=COUNT(SOPHIEU)
        FROM PHIEU_XUAT
        WHERE NgayXuat=@sNgayxuat
        RETURN @SoLuongPX
END
GO
--THỰC THI
SELECT DBO.F_SoLuongPX('2019-1-11') AS 'SLPX'
```

Hàm do người dùng định nghĩa (UDF)

Hàm Inline Table-valued

```
CREATE FUNCTION [tên_lược_đồ.]tên_hàm
( [ @tên_tham_số  kiểu_dữ_liệu [ ,...n ] ] )
RETURNS TABLE
[ AS ]
RETURN [ ( ] câu_lệnh_select [ ) ]
-- Gọi hàm
SELECT * FROM DBO.tên_hàm( đối số 1, đối số 2,…)
```

QLBH

- Tạo hàm đưa ra thông tin của nhà cung cấp với mã nhà cung cấp là tham số đầu vào
- NHACC (**MaNCC**, TenNCC, DiaChi, DienThoai)

```
CREATE FUNCTION F_TTNHACC( @sMaNCC char(4))
RETURNS TABLE
AS
RETURN (SELECT * FROM NHACC WHERE MANCC=@sMaNCC)
GO
-- Thực thi
SELECT * FROM DBO.F_TTNHACC('S003')
```

- Khi sử dụng UDF loại hàm đọc bảng cần quan tâm đến các quy tắc:
  - Mệnh đề RETURNS chứa duy nhất từ khóa TABLE
  - Không có phần thân của hàm với việc bắt đầu bởi khối BEGIN..END
  - Mệnh đề RETURN chứa một câu lệnh SELECT đơn giản nằm trong cặp dấu ngoặc đơn

Hàm do người dùng định nghĩa (UDF)

Hàm Multi-statement Table-Valued

```
CREATE FUNCTION [ tên_lược_đồ. ] tên_hàm
( [  @tên_tham_số   kiểu_dữ_liệu [ ,...n ] ] )
RETURNS @biến_bảng  TABLE <định_nghĩa_bảng>
[ AS ]
    BEGIN
        thân_hàm
        RETURN
    END
--- Gọi hàm
SELECT * FROM DBO.tên_hàm( đối số 1, đối số 2,…)
```

- Tạo hàm thống kê các mặt hàng đã được bán với đầu vào là mã cửa hàng. Bảng thống kê bao gồm các cột Mã Hàng, Tên hàng, số lượng có

NHACC (**MaNCC**, TenNCC, DiaChi, DienThoai)

HANG (**MaHang**, TenHang, DonGia, SoLuongCo, *MaNCC*)

PHIEU_XUAT (**SoPhieu**, NgayXuat, MaCuaHang)

DONG_PHIEU_XUAT ***(SoPhieu, MaHang***, SoLuongXuat)

```
CREATE FUNCTION f_TKMatHangBan (@sMaCH char(4))
RETURNS @TKMHB TABLE (MaHang char(4), TenHang nvarchar(50), SoluongCo int)
AS
BEGIN
        INSERT INTO @TKMHB
        SELECT HANG.MaHang, TenHang,SoLuongCo
        FROM HANG
INNER JOIN DONG_PHIEU_XUAT ON HANG.MaHang=DONG_PHIEU_XUAT.MaHang
INNER JOIN PHIEU_XUAT ON PHIEU_XUAT.SoPhieu=DONG_PHIEU_XUAT.SoPhieu
        WHERE MaCuaHang=@sMaCH
        RETURN
END
GO
-- Thực thi
SELECT * FROM DBO.f_TKMatHangBan ('A001')
```

- Khi sử dụng UDF dạng tạo bảng cần chú ý:
  - Không thể gọi một Stored Procedure từ các câu lệnh bên trong nó
  - Không thể sử dụng các hàm loại không xác định được xây dựng sẵn trong SQL Server (Getdate…)
  - UDF không thể được sử dụng để sửa đổi thông tin trên bảng cơ sở

Bài tập

1. Tạo hàm thống kê số lượng hàng của mỗi nhà cung cấp
2. Tạo hàm thống kê từ ngày X đến ngày Y có bao nhiêu phiếu được tạo. X, Y là ngày được nhập từ bàn phím
3. Tạo bảng thống kê, số phiếu và số lượng hàng bán từ ngày X đến ngày Y. Với X, Y là ngày được nhập từ bàn phím

Lịch học tuần sau (18/11/2022)

- **Sáng thứ 6:** chia 2 ca (nửa lớp học ca 1 (1-31), nửa lớp học ca 2 (32-62) học tại PM9-T9-A1
- **Chiều thứ 6:** Kiểm tra bài thường xuyên 1 (50’)
- Ca 1: 12h40-1h30 (1-15)
- Ca 2: 1h50-2h40 (16-32)
- Ca 3: 3h00-3h50 (33-48)
- Ca 4: 4h10-5h00 (49-62)

Kiểm tra tại (PM3-T7-A1)