BÀI GIẢNG HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU

Bài 4: Stored Procedure

Ths. Nguyễn Xuân Nhựt

Khoa: CNTT

Trường CĐ CÔNG THƯƠNG TP.HCM

- **Dùng từ khóa Declare khai báo biến**

  DECLARE *@ten_bien* *<kieu_du_lieu>* [*,...n*]

- **Gán giá trị cho biến**

  SET *@ten_bien = gia_tri*

Ex:

```
DECLARE @vLastName char(20), @vFirstName varchar(11)

SET @vLastName = 'Mary'

SELECT @vFirstName = FirstName
FROM Northwind.Employees
WHERE LastName = @vLastName

PRINT @vFirstName + ' ' + @vLastName
```

2. Lập trình cấu trúc trong SQL

– **Data Type** (1)

- **Integers**
  - Bigint: 8 bytes
  - Int: 4 bytes
  - Smallint: 2 bytes
  - Tinyint: 1 byte, từ 0 -> 255.
- **bit**
  - Bit: 1 hoặc 0 value.
- **decimal and numeric**
  - Decimal từ -10^38+1->10^38 –1.
  - Numeric: giống **decimal.**
- **money and smallmoney**
  - Money: 8 bytes
  - Smallmoney: 4 bytes
- **Approximate Numerics**
  - Float: từ -1.79E + 308 -> 1.79E + 308.
  - Real: từ -3.40E + 38 -> 3.40E + 38.

2. Lập trình cấu trúc trong SQL

– **Data Type** (2)

- **datetime and smalldatetime**
  - Datetime: từ 1/1/1753-> 31/12/9999.
  - Smalldatetime từ 1/1/1900, -> 6/6/2079.
- **Character Strings**
  - Char: Fixed-length non-Unicode character, <= 8,000 ký tự
  - Varchar: Variable-length non-Unicode , <= 8,000 ký tự
  - Text: Variable-length non-Unicode <=2^31 - 1 (2,147,483,647) ký tự
- **Unicode Character Strings**
  - ncharFixed-length Unicode , <=4,000 characters.
  - nvarcharVariable-length Unicode, <=4,000 characters
  - Ntext Variable-length Unicode <= 2^30 - 1 (1,073,741,823) characters.
- **Other Data Type**
  - Cursor: là một tham chiếu đến một cursor.
- Một biến không thể có kiểu là **text**, **ntext**, hoặc **image**

– **Hiển thị tên CSDL đang làm việc:**

```
SELECT DB_NAME() AS 'database'
Database
Northwind
(1 row(s) affected)
```

– **Định dạng kiểu ngày**

```
SET DATEFORMAT dmy
GO
DECLARE @vdate datetime
SET @vdate = '29/11/00'
SELECT @vdate
2000-11-29 00:00:00.000
```

2. Lập trình cấu trúc trong SQL

Function	Determinism
DATEADD	Deterministic
DATEDIFF	Deterministic
DATENAME	Nondeterministic
DATEPART	Deterministic except when used as DATEPART (dw, date). dw, the weekday datepart, depends on the value set by SET DATEFIRST, which sets the first day of the week.
DAY	Deterministic
GETDATE	Nondeterministic
GETUTCDATE	Nondeterministic
MONTH	Deterministic
YEAR	Deterministic

– **Hàm chuyển kiểu dữ liệu:**

- CAST **(** *expression* AS *data_type* **)**
- CONVERT **(** *data_type* [**(** *length* **)**] **,** *expression* [**,** *style*] **)**

– **BEGIN…END : định nghĩa một khối lệnh**

```
BEGIN
    sql_statement | statement_block
END
```

– **Câu lệnh IF:**

```
IF Boolean_expression
      { sql_statement | statement_block }
[ ELSE
      { sql_statement | statement_block } ]
```

EX:

```
if (select COUNT (*) FROM SINHVIEN WHERE TINH ='TPHCM') >0
BEGIN
        PRINT 'THERE ARE MANY  SV HCM'
        PRINT 'SV HCM'
END
ELSE
        PRINT 'WELLCOME'
```

– **Câu lệnh IF:**

Câu 1: Xem trong kết quả có mã sv có mã là @MASV chưa nếu không có thì xóa nếu có rồi thì in ra câu thông báo

```
DECLARE @MASV NVARCHAR(50)
SET @MASV ='92002'

IF EXISTS(SELECT MASV FROM KETQUA WHERE MASV=@MASV)
        PRINT ' CAN NOT DELETE THIS SV'
ELSE
BEGIN
        DELETE SINHVIEN WHERE MASV=@MASV
        PRINT 'SV DELETE'
END
```

- **While:**

WHILE *Boolean_expression*
{ *sql_statement* | *statement_block* }
[BREAK]
{ *sql_statement* | *statement_block* }
[CONTINUE]

- BREAK: thoát ra khỏi vòng while
- CONTINUE: restart lại vòng lặp, bỏ qua các lệnh sau CONTINUE.

Ví dụ WHILE

```
DECLARE @a  INT
SET @a =1
WHILE @a<100
BEGIN
        PRINT @a
        SET @a = @a+1
END
```

- CASE:

```
CASE input_expression
      WHEN when_expression THEN result_expression
         [ ...n ]
      [ ELSE else_result_expression    ]
   END
```

Ex:

```
(case PHAI
WHEN NU = 1 then 'Nu'
when NU=0 then 'Nam' end)
```

Hay:

```
(case Phai
WHEN  1 then 'Nu'
when  0 then 'Nam' end)
```

Câu 2:

Viết một đoạn mã lệnh để cho biết sinh viên có mã số 92013 có tên là gì và tổng điểm là bao nhiêu.

In ra họ tên sinh viên và tổng điểm. Nếu tổng điểm >=15 là đậu

Ngược lại là rớt

```
DECLARE @TONGDIEM REAL
DECLARE @TENSV NVARCHAR (50)
SELECT @TENSV = TENSV,  @TONGDIEM= SUM (DIEM)
FROM SINHVIEN SV, KETQUA KQ
WHERE SV.MASV = KQ.MASV AND SV.MASV = '92013'
GROUP BY TENSV
IF(@TONGDIEM >=15)
        PRINT 'SINH VIEN: ' +@TENSV + STR(@TONGDIEM,5) + 'DIEM DA DAU'
ELSE
        PRINT 'SINH VIEN: ' +@TENSV + STR(@TONGDIEM,5) + 'DIEM DA ROT'
```

Câu 3:

Hãy thực hiện các lệnh để xoá một môn học có mã môn học là AV. Nếu môn học này có trong Kết quả thì xóa kết quả có mã là AV trước rồi tiến hành xóa Mon học.

Câu 4: Lấy ra danh sách các điểm có tổng điểm MÔN CSDL là lớn nhất

Câu 5: Lấy ra các sinh viên có điểm trung bình <= điểm trung bình của sinh viên 92013

Câu 6: Tính điểm trung bình của 2 sinh viên 91008 và 92013 và in ra kết quả so sánh 2 điểm của sinh viên này

3. Stored Procedure

- **Định nghĩa:**

Stored Procedure là một tập hợp các câu lệnh SQL dùng để thực thi một nhiệm vụ nhất **định**. Nó hoạt động giống như một hàm trong các ngôn ngữ lập trình khác.

- **Ý nghĩa:**
  - Stored procedure là một tập các lệnh Transact SQL được đặt tên và lưu trữ trong database server
  - Có thể nhận tham số vào và tham số trả giá trị về
  - Trả về trạng thái thực thi của procedure là thành công hay không thành công

3. Stored Procedure

- **Lợi ích:**
  - SQL Server chỉ biên dịch các thủ tục lưu trữ một lần và sử dụng lại kết quả biên dịch này trong các lần tiếp theo.
  - Thủ tục lưu trữ được phân tích, tối ưu khi tạo ra nên việc thực thi chúng nhanh hơn nhiều so với việc phải thực hiện một tập rời rạc các câu lệnh SQL tương đương theo cách thông thường.
  - Thủ tục lưu trữ cho phép chúng ta thực hiện cùng một yêu cầu bằng một câu lệnh đơn giản thay vì phải sử dụng nhiều dòng lệnh SQL. Điều này sẽ làm giảm thiểu sự lưu thông trên mạng.

3. Stored Procedure

- **Cú pháp tạo procedure**:

```
CREATE PROCEDURE tên_thủ_tục
[ (danh_sách_tham_số + kiểu_dữ_liệu) ]
[ = default ] [ OUTPUT ]
AS
    Begin
        Câu_lệnh_của_thủ_tục
    End
```

3. Stored Procedure

- **Ví dụ 1**: Viết SP hiển thị danh sách các sinh viên

```
use  baitap01
CREATE PROCEDURE  spSV
AS
        select *
        from SINHVIEN
-------------------------------------------------------------
CREATE PROCEDURE  spYear
@nam int
AS
        select  *
        from Sinhvien
        where  year(ngaysinh) =@nam
```

3. Stored Procedure

- **Lời gọi thủ tục:**

  EXEC ***Tên_thủ_tục*** [(*danh_sách_tham_số*)]

- ***Cần lưu ý****: danh_sách_tham_số truyền vào phải theo đúng thứ tự khai báo các tham số trong thủ tục*
- *Nếu thủ tục được gọi từ một thủ tục khác, thực hiện bên trong một trigger hay phối hợp với câu lệnh SELECT, cấu trúc lời gọi hàm như sau:*

  Exec ***Tên_thủ_tục*** [(*danh_sách_tham_số*)]

- **Xóa một thủ tục:**

  Drop procedure *Tên_thủ_tục*

VD: Thực thi:

Exec spSv

Exec spYear

- **Ví dụ 2**: Viết SP cho biết số lượng SINH Viên

```
CREATE PROCEDURE spSV_count
@thang int, @nam int, @slsv int output
AS
        select @slsv = count(*)
        from Sinhvien
        where month(ngay) =@thang and year(ngay) =@nam
-----------------------------------------------
Thực thi:
declare @sl int
exec    spSV_count 5, 2000, @sl output
select  @sl
```

Ví dụ về việc thêm sinh viên vào trong dữ liệu

Nghiệp vụ của việc thêm sinh viên

1. Xem sinh viên có mã sv chưa nếu có rồi trả về -1
2. Xem khóa ngoại có chưa nếu chưa có không thêm được sinh viên này (return -1)

    Khóa Ngoại với KHOA

3. Thêm vào sinh viên này nếu đúng trả về 0

```
CREATE PROC ThemSinhVien
    @mssv varchar(10),
    @hoTen nvarchar(100),
    @ngaysinh int,
    @tinh nvarchar(20),
    @makhoa varchar(10)
AS
BEGIN
    IF(EXISTS(SELECT * FROM SinhVien s WHERE s.ma = @mssv)) BEGIN
        PRINT N'Mã số sinh viên ' + @mssv + N' đã tồn tại'
    RETURN -1
    END
    IF(NOT EXISTS(SELECT * FROM KHOA KH WHERE MAKHOA = @makhoa)) BEGIN
        PRINT N'Mã khoa c ' + @makhoa + N' chưa tồn tại'
        RETURN -1
    END
    INSERT INTO SinhVien(masv, TENSV, NGAYSINH, TINH, MAKHOA)
    VALUES(@mssv, @hoTen, @ngaysinh, @tinh, @makhoa)
      RETURN 0 /* procedure tự trả về 0 nếu không RETURN */
END
GO
DECLARE @kq INT
EXEC @kq = ThemSinhVien '0212005', N'Nguyễn Văn A', 1987, 'Kinh',
'TH2002/01'
PRINT @kq
```

3. Một số vấn đề khác trong SP

3.1 Mã hóa nội dung thủ tục

3.2 Biên dịch thủ tục

3.3 Thủ tục lồng nhau

3.4 Lệnh return trong SP

3.5 Sử dụng bảng tạm trong thủ tục

3.6 Thủ tục cập nhật dữ liệu

3.7 Thủ tục hiển thị dữ liệu

3.8 Tham số kiểu cursor trong SP

3.1 Mã hóa nội dung thủ tục

- Mục đích: không cho phép người dùng khác xem mã lệnh trong SP.

Cú pháp:

Create Proc[edure] <tên thủ tục>

@<tên tham số> <kdliệu> [Output] [,…]

With Encryption

As

[Declare <biến cục bộ>]

<Các lệnh>

3.2 Biên dịch thủ tục

- Mục đích: mỗi lần có người dùng gọi thủ tục thì bản thân nó được biên dịch lại.

Cú pháp:

Create Proc[edure] <tên thủ tục>

@<tên tham số> <kdliệu> [Output] [,…]

With Recompile [Encryption]

As

[Declare <biến cục bộ>]

<Các lệnh>

Cú pháp: cách khác để biên dịch lại thủ tục

Exec <tên thủ tục> [<các tham số>] With Recompile

Cách này dùng với việc tạo thủ tục không dùng option With Recompile

3.3 Thủ tục lồng nhau

- SQL cho phép các thủ tục lồng vào nhau (gọi lẫn nhau).
- SQL cho phép lồng tối đa 32 cấp.

3.4 Lệnh return trong SP

- Cú pháp:

Return [<giá trị>]

Exec @<biến> = <Tên thủ tục>[<các tham số>][…]

Chú ý: khi gặp câu lệnh return ngay lập tức SP kết thúc

Ví dụ: SP chứa lệnh return

```
--Tao SP dua vao ten mon hoc, va tra ve 1 tuc la co sv hoc mon hoc do, nguoc lai tra ve 0
Create Proc cau29_SP @tenmh nvarchar(30) As
If exists (select       sv.masv,TenSV
           from         SinhVien sv,KetQua kq,MonHoc mh
           where        sv.MaSV=kq.MaSV and
           kq.MaMH=mh.MaMH and TenMH=@tenmh)
Begin
           print 'Danh sach cac sv co diem mon ' +@tenmh+ ' lon nhat:'
           select       sv.masv,TenSV
           from         SinhVien sv,KetQua kq,MonHoc mh
           where        sv.MaSV=kq.MaSV and
                        kq.MaMH=mh.MaMH and
                        TenMH=@tenmh and
                        Diem in (select         max(Diem) as DiemMax
                                 from           KetQua kq,MonHoc mh
                                 where          kq.MaMH=mh.MaMH and TenMH=@tenmh)
           return 1
End
Else       return 0
```

Lời gọi thủ tục

```
declare @kq int, @tenmh varchar(30)
set @tenmh='Triet hoc'
Exec @kq=cau29_SP @tenmh
if @kq=0
        print 'Khong ton tai sinh vien nao hoc mon '+ @tenmh
go
--------------------------------------------------------------------------------
declare @kq int --,@tenmh varchar(30)
Exec @kq=cau29_SP @tenmh='Van phong'
if @kq=0
begin
        print 'Khong ton tai sinh vien nao hoc mon '
        print @tenmh  --loi xay ra
end
```

5.5 Bảng tạm trong SP

- Trong quá trình xây dựng SP, đôi khi ta phải tính toán, thống kê trước khi viết mã cho SP. Có 2 cách giải quyết vấn đề này:
  - Một là xây dựng view tính toán trước.
  - Hai là xây dựng bảng tạm trong chính SP đó.

- Cú pháp xây dựng bảng tạm trong SP:

Select <các thuộc tính> INTO #<tên bảng tạm>

From ….

…..

…..

Drop table #<tên bảng tạm>

3.6 Thủ tục cập nhật dữ liệu

```
Create Proc SP_updatemasv
        @old varchar(5), @new varchar(5)
As
Begin
        If Exists (select *from sys.objects where name='#temp')
                Drop table #temp
        Select * into #temp from ketqua where masv=@old
        Delete from ketqua  where masv=@old
        Update #temp set masv=@new
        Update sinhvien set masv=@new where masv=@old
        Insert into ketqua select *from #temp
        Drop table #temp
End
Exec SP_updateMaSV 91002,1999 --sua masv 91002 thanh 1999
```

3.7 Thủ tục hiển thị dữ liệu

- Đối với các báo cáo để lấy dữ liệu ta có các nguồn khác nhau:
  - Dùng view (bảng ảo)
  - Câu lệnh select trực tiếp
  - Đối tượng thủ tục nội tại

3.8 Tham số kiểu cursor trong SP

- Là loại tham số chứa danh sách các bộ giá trị (bảng).

4. Function - Hàm

- **Function:**

- Do người dùng định nghĩa, chia làm 3 loại:

(1) Scalar (hàm vô hướng): được sử dụng để trả về 1 giá trị duy nhất dựa trên các tham số truyền vào.

(2) Inline table-valued (hàm nội tuyến): trả về 1 bảng dựa trên 1 câu lệnh SQL duy nhất định nghĩa các dòng và các cột trả về.

(3) Multi-statement table-valued: trả về kết quả là một tập hợp nhưng có thể dựa trên nhiều câu lệnh SQL

4. Function - Hàm

- **Hàm vô hướng – Scalar UDF:**

```
CREATE   FUNCTION     tên_hàm [ (danh_sách_tham_số) ]
RETURNS  (kiểu_trả_về_của_hàm)
AS
   begin
        Các_câu_lệnh_của_hàm
        return  (@giá trị trả về)
   end
```

4. Function - Hàm

- **Ví dụ 1:** Câu lệnh dưới đây định nghĩa hàm tính ngày trong tuần (thứ trong tuần) của một giá trị kiểu ngày.

```
    create function f_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)
    end
--Thực thi:
    Select Hoten,dbo.f_thu(ngaysinh) From   nhanvien
```

4. Function - Hàm

- **Hàm nội tuyến – Inline UDF:**

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

- **Ví dụ 2:** Lấy danh sách khách hàng mua hàng trong tháng …, tháng là tham số truyền vào.

```
--Xây dựng hàm:
create function DanhSachKH(@thang int)
returns Table
as
        return(select k.MAKH,TENKH
                from KHACHHANG k, HOADON h
                where k.MAKH = h.MAKH and
                month(NGAY)=@thang)
--Thực thi:
select * from dbo.DanhSachKH(5)
```

- **Multi statement UDF:**

```
CREATE   FUNCTION     tên_hàm [ (danh_sách_tham_số) ]
RETURNS  @biến_bảng TABLE  định_nghĩa_bảng
AS
   begin
        Các_câu_lệnh_của_hàm
        return
   end
```

Lưu ý:

- *Sau từ khoá RETURNS là một biến bảng được định nghĩa với cú pháp lệnh như tạo bảng.*
- *Sau từ khoá RETURN cuối hàm không có tham số đi kèm.*

- **Ví dụ 3:** Lấy danh sách khách hàng mua hàng và ngày đặt hàng, tham số truyền vào là Makh, nếu tham số Makh truyền vào là 0 thì lấy danh sách tất cả khách hàng -

-Xây dựng hàm:

```
create function DanhSachKH_multi(@makh nvarchar(5))
returns @kh Table(      makh nvarchar(5), tenkh nvarchar(30),
                        ngay datetime)
as
begin
        if @makh ='0'
        begin
                insert into @kh select k.makh, tenkh,ngay
                from khachhang k, hoadon h where k.makh =h.makh
        end
```

4. Function - Hàm

- **Ví dụ 3:** Lấy danh sách khách hàng mua hàng và ngày đặt hàng, tham số truyền vào là Makh, nếu tham số Makh truyền vào là 0 thì lấy danh sách tất cả khách hàng -

-Xây dựng hàm:

```
else
        begin
                insert into @kh select k.makh, tenkh,ngay
                from khachhang k, hoadon h
                where k.makh = h.makh and h.makh =@makh
        end
return
end
--Thực thi:
select * from dbo.DanhSachKH_multi('KH01')
```