Chương 6

1

Thủ tục – Hàm Procedure-Function

Nội dung

2

- Khái niệm về thủ tục
- Các thao tác cơ bản với thủ tục
- Tham số bên trong thủ tục
- Một số vấn đề khác trong thủ tục
- Hàm
- Giao tác

Khái niệm về thủ tục

3

- Một thủ tục là một đối tượng trong cơ sở dữ liệu bao gồm một tập nhiều câu lệnh SQL được nhóm lại với nhau thành một nhóm với những khả năng sau:
- Có thể bao gồm các cấu trúc điều khiển (IF, WHILE, FOR).
- Bên trong thủ tục lưu trữ có thể sử dụng các biến nhằm lưu giữ các giá trị tính toán được, các giá trị được truy xuất được từ cơ sở dữ liệu.

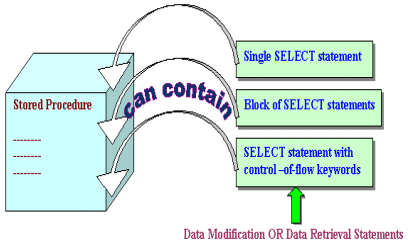
Khái niệm về thủ tục

4

- Một thủ tục có thể nhận các tham số truyền vào cũng như có thể trả về các giá trị thông qua các tham số.
- Khi một thủ tục lưu trữ đã được định nghĩa, nó có thể được gọi thông qua tên thủ tục, nhận các tham số truyền vào, thực thi các câu lệnh SQL bên trong thủ tục và có thể trả về các giá trị sau khi thực hiện xong.

Khái niệm về thủ tục

5



Stored Procedure

Stored Procedure vs. SQL Statement

SQL Statement	*Stored Procedure*
First Time - *Check syntax* - *Compile* - *Execute* - *Return data*	Creating - *Check syntax* - *Compile*
Second Time - *Check syntax* - *Compile* - *Execute* - *Return data*	First Time - *Execute* - *Return data*
	Second Time - *Execute* - *Return data*

Lợi ích của thủ tục

7

- Đơn giản hoá các thao tác trên cơ sở dữ liệu nhờ vào khả năng module hoá các thao tác này.
- 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.
- Cho phép thực hiện cùng một yêu cầu bằng một câu lệnh đơn giản thay vì phải sử dụng nhiều dòng lệnh SQL→ làm giảm thiểu sự lưu thông trên mạng.
- Có thể cấp phát quyền cho người sử dụng thông qua các thủ tục lưu trữ, nhờ đó tăng khả năng bảo mật đối với hệ thống.

Phân loại thủ tục

8

- Các loại Procedures
  - User-defined
  - System
  - Temporary
  - Remote
  - Extended

Phân loại thủ tục

9

- System sp: được lưu trữ trong CSDL master. Các thủ tục có tên bắt đầu là sp. Chúng đóng vai trò khác nhau của các tác vụ được cung cấp trong SQL Server.
- Local sp: được lưu trữ trong các CSDL người dùng, nó thực thi các tác vụ trong CSDL chứa nó. Được người sử tạo hay từ các sp hệ thống.

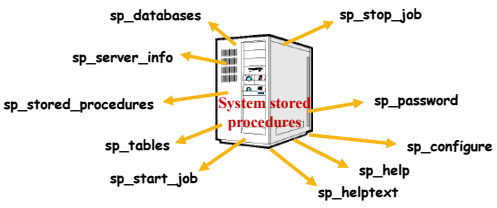
Phân loại thủ tục

10

- Temporary sp: giống local sp nhưng nó chỉ hiện hữu cho đến khi kết nối tạo ra nó bị đóng. Nó được lưu trong CSDL TempDB. Có 3 loại temporary sp: local (private), Global, sp tạo trực tiếp trong TempDB.
- Extended sp: là một thủ tục được tạo từ các ngôn ngữ lập trình khác (không phải SQL Server) và nó được triển khai tính năng của một thủ tục trong SQL Server. Các thủ tục này có tên bắt đầu là xp.
- Remote sp: là một thủ tục được gọi thực thi từ một server từ xa.

Một số thủ tục hệ thống

11



Một số thủ tục hệ thống

12

System Store Procedure	Description
Sp_databases	Lists all the databases available on the server.
Sp_server_info	Lists server information, such as, character set, version, and sort order.
Sp_store_procedure	Lists all the stored procedures avaible in the current environment.
Sp_table	Lists all the objects that can be queried in the current environment.
Sp_start_job	Starts an automated task immediately
Sp_stop_job	Stops an automated task that is running

Một số thủ tục hệ thống

13

System Store Procedure	Description
Sp_password	Change the password for a login account
Sp_configue	Changes the SQL Server global configuration option. When used without options, display the current server settings.
Sp_help	Displays information about any database object.
Sp_helptext	Displays the actual text for a rule, a default, or an un-define function, trigger or view

User-defined Stored Procedures

14

- Được tạo bởi người sử dụng trong CSDL hiện hành.
- Các thủ tục có thể được tạo trước khi các đối tượng mà thủ tục tham chiếu.

Cách 1 : Use Enterprice Manager

Right Click at Database, Select New Store Procedure

Cách 2 : Made Store Procedure by Wizard

Click menu Tools, select Wizard, Click Database, chọn Create Store Prodedure Wizard.

User-defined Stored Procedures

15

Cách 3 : The CREATE PROCEDURE Statement

```
CREATE PROC [ EDURE ] procedure_name [ ;
number ]
  [ { @parameter data_type }
      [ VARYING ] [ = default ] [ OUTPUT ]
  ] [,...n ]
[ WITH { RECOMPILE | ENCRYPTION |
RECOMPILE, ENCRYPTION } ]
[ FOR REPLICATION ]
AS sql_statement [ ...n ]
```

Kiểm tra sự tồn tại của thủ tục

```
sp_helptext 'Procedure_name'
sp_help 'Procedure_name'
sp_depends 'Procedure_name'
Sp_stored_procedures
```

User-defined Stored Procedures

16

- RECOMPILE: Thông thường, thủ tục sẽ được phân tích, tối ưu và dịch sẵn ở lần gọi đầu tiên. Nếu tuỳ chọn WITH RECOMPILE được chỉ định, thủ tục sẽ được dịch lại mỗi khi được gọi.
- ENCRYPTION: Thủ tục sẽ được mã hoá nếu tuỳ chọn WITH ENCRYPTION được chỉ định. Nếu thủ tục đã được mã hoá, ta không thể xem được nội dung của thủ tục.

User-defined Stored Procedures

17

- Example: Không có tham số

```
CREATE PROC Tong
as
        Declare @a int, @b int
        Set @a =7
        Set @b =3
        Print 'Tong ='+convert(varchar(10),@a+@b)
        Print 'Hieu='+convert(varchar(10),@a-@b)
        Print 'Tich ='+convert(varchar(10),@a*@b)
        If @b<>0
                Print 'Thuong ='+convert(varchar(10),@a/@b)
        Else
                Print 'Khong chia duoc'
--Thuc thi
EXEC Tong
```

User-defined Stored Procedures

18

- Example: có tham số

```
CREATE PROC Tong1(@a int, @b int)
as
        Print 'Tong ='+convert(varchar(10),@a+@b)
        Print 'Hieu='+convert(varchar(10),@a-@b)
        Print 'Tich ='+convert(varchar(10),@a*@b)
        If @b<>0
                Print 'Thuong ='+convert(varchar(10),@a/@b)
        Else
                Print 'Khong chia duoc'
--Thực thi

EXEC Tong1 5,7
```

User-defined Stored Procedures

19

Ví dụ 2: hiện ra danh sách khách hang ở tp londor

```
CREATE PROCEDURE London_KH AS
        SELECT * FROM Customers WHERE City=
        'London'
--Thuc thi
Exec LonDon_KH
CREATE PROCEDURE TP_KH (@TP nvarchar(15))
AS
SELECT * FROM Customers WHERE City=@TP
Declare @TP nvarchar(5)
Set @TP="LonDon"
Exec tp_KH @tp
```

Thực thi một Stored Procedure

20

Cú pháp:

```
[ EXEC [ UTE ] ]
    {     [ @return_status = ]
          { procedure_name [ ;number ] | @procedure_name_var
    }
    [ [ @parameter = ] { value | @variable [ OUTPUT ] | [ DEFAULT ] ]
       [ ,...n ]
 [ WITH RECOMPILE ]
```

- Ví dụ 1:

```
EXECUTE TP_KH  'London'
GO
EXECUTE TP_KH 'Paris'
GO
DECLARE @City nvarchar(15), @Return_Value tinyint
SET @City='LonDon'
EXECUTE @Return_Value=TP_KH @City
PRINT @Return_Value
GO
```

Sử dụng tham số

21

- Có 2 loại tham số
  - Input paramater: tham số nhập, đưa giá trị của tham số để thông báo cho thủ tục nên làm gì trong CSDL
  - Output parameter: tham số xuất chứa giá trị trả về của thủ tục.
- Khai báo tham số:

  {@parameter data_type} [= default|NULL][varying] [OUTPUT]

Sử dụng tham số

22

Cú pháp

```
CREATE PROCEDURE procedure_name
@Parameter_name data_type
AS
:
```

Sử dụng tham số

23

- Example

```
CREATE PROC Tong
@a int, @b int
as      Declare @tong int, @hieu int, @tich int, @thuong real
        Set @tong =@a +@b
        Set @hieu = @a -@b
        Set @tich = @a *@b
        Print 'Tong ='+convert(varchar(10),@tong)
        Print 'Hieu='+convert(varchar(10),@hieu)
        Print 'Tich ='+convert(varchar(10),@tich)
        if @b<>0
                Set @thuong = @a/@b
                Print 'Thuong ='+convert(varchar(10),@thuong)
        else
                Print 'Khong chia duoc'
EXEC tong 4,7
```

Tạo thủ tục với tham số

24

Ví dụ 2

```
CREATE PROCEDURE city_KH
@KH_city varchar(15)
AS
    SELECT * FROM Customers
            WHERE City = @KH_city
```

Thực thi thủ tục

Exec city_kh 'London'

Tạo thủ tục với tham số

25

Ví dụ 3:

```
CREATE PROCEDURE CustOrderHist @CustomerID
    nchar(5)
AS
    SELECT ProductName, Total=SUM(Quantity)
    FROM Products P, [Order Details] OD, Orders O,
        Customers C
    WHERE C.CustomerID = @CustomerID
    AND C.CustomerID = O.CustomerID AND O.OrderID =
    OD.OrderID AND OD.ProductID = P.ProductID
    GROUP BY ProductName

    exec CustOrderHist  'NORTS'
```

Thủ tục có trị trả về

26

- Trị trả về là giá trị kiểu integer.
- Mặc định giá trị trả về là 0

Cú pháp

```
DECLARE @return_variable_name data_type
EXECUTE @return_variable_name = procedure_name
```

Ví dụ tạo thủ tục có giá trị trả về

27

- Example

```
CREATE PROC Tinhtoan
@a int, @b int , @tong int output, @hieu int output, @tich int output, @thuong
real output
as
Begin
        Set @tong =@a +@b
        Set @hieu = @a -@b
        Set @tich = @a *@b
        if @b<>0
        begin
                Set @thuong = @a/@b
                Print 'Thuong ='+convert(varchar(10),@thuong)
        end
        else
                Print 'Khong chia duoc'

        Set @b = @b *100
```

Ví dụ tạo thủ tục có giá trị trả về

28

- Thực thi

```
Declare @tong int, @hieu int, @tich int, @thuong real,@a int, @b int
Set @a= 8
Set @b=5
Print 'a = '+convert(varchar(10),@a)
Print 'b= '+convert(varchar(10),@b)
EXEC tinhtoan  @a, @b, @tong OUTPUT,@hieu OUTPUT,
 @tich output, @thuong output
        Print 'a = '+convert(varchar(10),@a)
        Print 'b= '+convert(varchar(10),@b)
        Print 'Tong ='+convert(varchar(10),@tong)
        Print 'Hieu='+convert(varchar(10),@hieu)
        Print 'Tich ='+convert(varchar(10),@tich)
        Print 'Thuong ='+convert(varchar(10),@thuong)
```

Ví dụ tạo thủ tục có giá trị trả về

29

Example 5 :

```
CREATE PROCEDURE prcGetUnitPrice_UnitsInStock  @ProductID int,
        @Unitprice Money OUTPUT, @UnitsInStock smallint OUTPUT
AS
    BEGIN
        IF EXISTS (SELECT * FROM Products
                WHERE ProductID = @ProductID)
            BEGIN
                SELECT @Unitprice=Unitprice,@UnitsInStock=UnitsInStock
                FROM Products
                WHERE ProductID=@ProductID
                RETURN 0
            END
        ELSE
            RETURN 1
        END
```

Ví dụ tạo thủ tục có giá trị trả về

30

Example 5 :

- Declare @Unitprice Money, @UnitsInStock smallint
- EXEC prcGetUnitPrice_UnitsInStock 1, @Unitprice OUTPUT,
  @UnitsInStock OUTPUT
- Select @Unitprice AS Gia, @UnitsInStock AS SoLuongTon

Ví dụ tạo thủ tục có giá trị trả về

31

Ví dụ: viết 1 thủ tục đếm xem có bao nhiêu khách hang ở thành phố bất kỳ bằng 2 cách dung dung tham số có output và input

```
CREATE PROCEDURE KH_city
@KH_city VARCHAR(15) AS
DECLARE @KH_return int
SELECT @KH_return=COUNT(*) FROM
CUSTOMERS WHERE City = @KH_city
RETURN @KH_return+1
--Thuc thi
Declare @SoKH int
EXEC @SoKH=KH_city 'LonDon'
Print 'So KH la '+convert(varchar(4),@SoKH)
```

Ví dụ tạo thủ tục có giá trị trả về

32

```
Example 3:
CREATE PROCEDURE prcDisplayUnitPrice_UnitsInStock @ProductID int
AS
BEGIN
    DECLARE @UnitPrice Money, @UnitsInStock smallint
    DECLARE @ReturnValue Tinyint
    EXEC @ReturnValue = prcGetUnitPrice_UnitSInStock @ProductID,
          @UnitPrice output, @UnitsInStock output
    IF (@ReturnValue = 0)
     BEGIN PRINT 'The Status for product: '+ Convert(char(10), @ProductID)
        PRINT 'Unit price            : '  + CONVERT( char(10), @Unitprice)
        PRINT 'Current Units In Stock:' + CONVERT (char(10), @UnitsInStock)
     END
    ELSE  PRINT 'No records for the given productID ' + Convert(char(10),
          @ProductID)
END
```

Ví dụ tạo thủ tục có giá trị trả về

33

Example:

```
EXECUTE prcDisplayUnitPrice_UnitsInStock 1222
GO
EXECUTE  prcDisplayUnitPrice_UnitsInStock 1
```

Sửa một thủ tục - Stored Procedure

34

```
ALTER PROC[EDURE] procedure_name
[WITH option]
AS
    sql_statement [...n]
```

```
ALTER PROCEDURE KH_city
AS
SELECT * FROM dbo.Customers;

exec KH_city
```

Sửa một thủ tục - Stored Procedure

35

- Example:

```
ALTER PROC prcListCustomer @City char(15)=NULL
AS
   BEGIN
   IF @city is NULL
   BEGIN
     PRINT 'Usage: prcListCustomer <City>'
     RETURN
   END
     PRINT 'List of Customers'
     SELECT CustomerID,CompanyName,Address,Phone
     FROM Customers
     WHERE City = @City
   END
```

Sửa một thủ tục - Stored Procedure

36

- Example 1:

```
ALTER PROC prcListCustomer @City char(15)
AS
BEGIN
  IF EXISTS (SELECT * FROM Customers WHERE City=@city)
      BEGIN
     PRINT 'List of Customers'
     SELECT CustomerID,CompanyName,Address,Phone
     FROM Customers  WHERE City = @City
     RETURN 0
       END
   ELSE
      BEGIN
     PRINT 'No Records Found for given city'
     RETURN 1
       END
END
```

Xóa một Stored Procedure

37

DROP PROCEDURE *proc_name*

Ví dụ:

DROP PROCEDURE City_KH

User-defined Stored Procedures

38

Exercise: **Create a stored procedure called dbo.usp_CustomerTotals displays the total sales from the TotalDue column per year and month for each customer. Test the stored procedure.**

```
IF OBJECT_ID('dbo.usp_CustomerTotals') IS NOT NULL BEGIN
DROP PROCEDURE dbo.usp_CustomerTotals;
END;
GO
CREATE PROCEDURE dbo.usp_CustomerTotals AS
SELECT C.CustomerID, YEAR(OrderDate) AS OrderYear,
MONTH(OrderDate) AS OrderMonth, SUM(TotalDue) AS TotalSales
FROM Sales.Customer AS C
INNER JOIN Sales.SalesOrderHeader AS SOH ON C.CustomerID =
SOH.CustomerID
GROUP BY C.CustomerID, YEAR(OrderDate), MONTH(OrderDate)
GO
EXEC dbo.usp_CustomerTotals;
```

Stored Procedure
Using Parameters-Return value

39

Exercise:Create a stored procedure called dbo.usp_ProductSales that accepts a ProductID for a parameter and has an OUTPUT parameter that returns the number sold for the product. Test the stored procedure.

```
IF OBJECT_ID('dbo.usp_ProductSales') IS NOT NULL BEGIN
DROP PROCEDURE dbo.usp_ProductSales;
END;
GO
CREATE PROCEDURE dbo.usp_ProductSales @ProductID INT,
@TotalSold INT = NULL OUTPUT AS
SELECT @TotalSold = SUM(OrderQty)
FROM Sales.SalesOrderDetail
WHERE ProductID = @ProductID;
GO
DECLARE @TotalSold INT;
EXEC dbo.usp_ProductSales @ProductID = 776, @TotalSold = @TotalSold OUTPU
PRINT @TotalSold;
```

Phần 2

40

HÀM - FUNCTION

NỘI DUNG

41

- Khái niệm về Hàm
- Các loại hàm
- Các loại giá trị trả về của UFDs
- Tạo và quản lý hàm UFDs
- Scalar Function
- Table-valued Function
- Sử dụng hàm UFDs

Khái niệm về Hàm

42

- Hàm tương tự thủ tục bao gồm các phát biểu T-SQL và một số cấu trúc điều khiển được lưu với một tên và được xử lý như một đơn vị độc lập. Hàm được biên dịch trước, không cần kiểm tra và biên dịch lại.
- Điểm khác biệt giữa hàm và thủ tục là hàm trả về một giá trị thông qua tên hàm còn thủ tục thì không.

Khái niệm về Hàm

43

- **Hàm được dùng trong:**
  - Lệnh print hay lệnh Select để hiển thị giá trị trả về của hàm.
  - Danh sách chọn của một câu lệnh Select để cho ra một giá trị.
  - Một điều kiện tìm kiếm của mệnh đề Where trong các câu lệnh T-SQL.

Ưu điểm của Hàm

44

- Người gởi chỉ gởi một câu lệnh đơn và SQL Server chỉ kiểm tra một lần sau đó tạo ra một execute plan và thực thi. Cú pháp của các câu lệnh SQL đã được SQL Sever kiểm tra trước khi save nên nó không cần kiểm lại khi thực thi→giảm nghẽn mạng
- Bảo trì (maintainability) dễ dàng hơn do việc tách rời giữa business rules và database. Nếu có một sự thay đổi nào đó về mặt logic thì ta chỉ việc thay đổi code bên trong hàm
- Security: có thể được encrypt (mã hóa) để tăng cường tính bảo mật.

Các loại Hàm

45

- **Có hai loại:**
  - **Built-in functions:** Hoạt động như là một định nghĩa trong T-SQL và không thể hiệu chỉnh. Chỉ được tham chiếu trong các câu lệnh T-SQL. Trị trả về là một tập các dòng(Rowset), vô hướng(scalar) và argergate(thống kê).
  - **User-define functions** hay còn gọi là UDFs: do người dùng tự định nghĩa để đáp ứng một mục tiêu nào đó. Các tham số truyền vào không được mang thuộc tính OUTPUT, do đó giá trị trả về cho hàm bằng phát biểu RETURN. Giá trị trả về là giá trị vô hướng (Scalar valued) hay bảng (Table – valued).

Các loại giá trị trả về của UFDs

46

- Scalar Function: Một hàm vô hướng trả về một giá trị đơn và có thể được dùng bất cứ nơi nào của biểu thức hay có thể được dùng câu lệnh SELECT, mệnh đề SET của lệnh UPDATE,... Một hàm vô hướng có thể được xem như kết quả của vài phép toán hay hàm chuỗi.
- Table-valued Function : Một hàm có giá trị trả về là một tập kết quả và có thể được dùng bất cứ nơi nào mà bảng hay view được dùng. Hàm giá trị bảng có thể được tham chiếu trong mệnh đề FROM của câu lệnh SELECT.
- Tên và những thông tin về Function khi được tạo ra sẽ chứa trong SysObjects table còn phần text của nó chứa trong SysComments table.

Các lệnh tạo và quản lý UDF

47

- **Tạo Hàm**
  - **CREATE FUNCTION <TenHam>…**
- **Sửa Hàm**
  - **ALTER FUNCTION <TenHam>…**
- **Xóa Hàm**
  - **DROP FUNCTION statement**
- **Thực thi Hàm**
  - Dùng lệnh **Print**
  - Dùng Lệnh **Select**
- **Xem các lệnh của UDFs**
  - **Sp-helptext TenHam**

Scalar Function

48

1. Scalar Function (Không có tham số)

- **Là hàm không nhận giá trị từ bên ngoài truyền vào.**
- Cú pháp:

```
CREATE FUNCTION [Owner_name.]function_name
RETURNS scalar_return_data_type
[WITH { ENCRYPTION | SCHEMABINDING } ]
[ AS ]
    BEGIN
            function_body
            RETURN scalar_expression
    END
```

Scalar Function – Tạo Hàm

49

- Ví dụ 1 : Hàm trả về tổng 2 số 4 và 6

```
Create function tong2so()
Returns int
as
Begin
        Declare @so1 int, @so2 int
        Set @so1 = 4
        set @so2 =6
        Return @so1+@so2
end
--thuc hien
print 'Tong = ' +convert(char(10),dbo.tong2so())
print 'Tong = ' +convert(char(10),tong2so())
select dbo.tong2so() as Tong
```

Scalar Function – Sửa và Xóa Hàm

50

```
Alter function tong2so()
Returns int
With Encryption
as
Begin
        Declare @so1 int, @so2 int
        Set @so1 = 4
        set @so2 =6
        Return @so1+@so2
End
--Xem lệnh
        sp_helptext tong2so
--thuc hien
        print 'Tong = ' +convert(char(10),dbo.tong2so())
        select dbo.tong2so() as Tong
--Xóa hàm        Drop function Tong2so
```

Scalar Function

51

Ví dụ 2 : Hàm trả về tổng tiền của khách hàng có mã là TOMSP

```
Create function Tongtien()
Returns money
AS
Begin
        Declare @tong money
        Select @tong = sum(unitprice*Quantity) from orders o,
                [Order Details] d
        where o.orderid = d.orderid and customerid = 'TOMSP'
        Return @tong
End
print 'Tong = ' +convert(char(10),dbo.tongtien())
select dbo.tongtien() as [Tong Tien Cua Khach Hang TOMPS]
```

Scalar Function

52

2. Scalar Function (Có tham số)

- **Là hàm nhận các giá trị từ bên ngoài truyền vào.**
- Cú pháp:

```
CREATE FUNCTION [owner_name.]function_name
([{@parameter_name [AS] data_type [=default]} [ ,...n ]])
RETURNS scalar_return_data_type
[WITH { ENCRYPTION | SCHEMABINDING } ]
[ AS ]
    BEGIN
            function_body
            RETURN scalar_expression
    END
```

Scalar Function

53

Ví dụ 3 : Hàm trả về tổng của hai số bất kỳ

```
Create function tong(@so1 int, @so2 int)
Returns int
as
Begin
        Return @so1+@so2
end
-- Thuc hien ham
Declare @a int, @b int
Set @a = 4
Set @b =6
Print 'Tong cua '+convert(char(5),@a) +'  '+
        convert(char(5),@b)+'='+convert(char(5),dbo.tong(@a,@b))
Select dbo.tong(@a,@b) as tong
```

Scalar Function

54

Example 4 : Hàm trả về tổng tiền của khách hàng nào đó

```
Create function TongtienTS(@makh nchar(5))
Returns money
AS
Begin
        Declare @tong money
        Select @tong = sum(unitprice*Quantity) from orders o join
        [Order Details] d on o.orderid = d.orderid
        where customerid = @makh
        Return @tong
End
declare @ma nchar(5)
Set @ma = 'TOMSP'
print 'Tong = ' +convert(char(10),dbo.tongtients(@ma))
select dbo.tongtients(@ma) as Tong
```

Scalar Function

55

Bài tập áp dụng : Hàm trả về thứ bằng tiếng việt

```
Create function thu(@ngay datetime)
Returns varChar(10)
As
Begin   Declare @t varchar(10), @d tinyint
        Set @d = datepart(dw,@ngay)
        Set @t = case
                When @d = 1 then 'Chu Nhat'
                When @d = 2 then 'Hai'
                When @d = 3 then 'Ba'
                When @d = 4 then 'Tu'
                When @d = 5 then 'Nam'
                When @d = 6 then 'Sau'
                When @d = 7 then 'Bay'
                end
        Return @t
```

Scalar Function

56

Example 6 : Hàm trả về thứ bằng tiếng việt

```
declare @ngaysinh datetime
Set @ngaysinh = getdate()     --hay '04/12/1982'--
Print 'Ban sinh vao Thu '+dbo.thu(@ngaysinh) +
 ' Ngay '+ convert(char(3),day(@ngaysinh)) + ' thang ' + Convert(char(3),
month(@ngaysinh))+' nam ' +convert(char(5),year(@ngaysinh))
--thuc hien voi cau lenh Select
Select employeeid, LastName +'  '+FirstName as Hoten, thu =
dbo.thu(birthdate) from Employees
Select employeeid, LastName +' '+FirstName as Hoten, [Thu Ngay Thang
Nam Sinh] ='Thu '+dbo.thu(birthdate) +          ' Ngay '+
convert(char(2),day(birthdate)) + ' thang ' + Convert(char(2),
month(birthdate))+ ' nam ' +convert(char(4),year(birthdate)) from Employees
```

Scalar Function

57

Bài tập 2 : Hàm trả về Tổng tiền của các sản phẩm

```
Create function TotalAmount
(@Unitprice money, @quantity Smallint,@Discount real)
Returns Money
As
Begin
        Return (@Unitprice * @Quantity)*(1-@discount)
End
--Su dung
Select Productid, Total =
dbo.TotalAmount(Unitprice,Quantity,Discount)
From [Order Details]
Where Orderid =10250
```

Scalar Function

58

Bài tập 3 :

- Viết hàm trả về chiết khấu của sản phẩm dựa vào số lượng lập hoá đơn và theo quy định sau:
- Nếu số lượng <=5 thì chiết khấu là 0.05
- Nếu số lượng từ 6 đến 10 thì chiết khấu 0.07
- Nếu số lượng từ 11 đến 20 thì chiết khấu là 0.09
- ngược lại thì 0.1

User-Defined Function
Scalar Function Syntax

Bài tập:

1. Viết hàm tên SubTotalOfEmp (dạng scalar function) trả về tổng doanh thu của một nhân viên trong một tháng tùy ý trong một năm tùy ý, với tham số vào @EmpIID, @MonthOrder, @YearOrder *(Thông tin lấy từ bảng [Sales].[SalesOrderHeader])*
2. Viết hàm tên là InventoryProd (dạng scalar function) với tham số vào là @ProductID và @locationID trả về số lượng tồn kho của sản phẩm trong khu vực tương ứng với giá trị của tham số *(Dữ liệu lấy từ bảng[Production].[ProductInventory])*

The table-valued UDFs

60

- The table-valued UDFs: **được chia thành hai loại là inline và multistatement table-valued.**
- *Inline table-valued UDF:*
  - *Được xem như là một View có tham số*. Thực thi một câu lệnh Select như trong một view nhưng có thể bao gồm các tham số giống thủ tục
  - Cú pháp:

```
CREATE FUNCTION [owner_name.]function_name
([{@parameter_name [AS] data_type [=default]} [ ,…n ]])
RETURNS TABLE
[WITH { ENCRYPTION | SCHEMABINDING }]
[AS]
RETURN [(] select-stmt [)]
```

The table-valued UDFs

61

Ví dụ 1: Cho biết tổng số hóa đơn của khách hàng bất kỳ.

```
CREATE FUNCTION  CountOrderCust (@cust  varchar(5))
RETURNS TABLE
AS
RETURN (Select CustomerID, count(orderid)as countOrder
        From orders
        Where customerID like @cust
        Group by customerID )
```

- Thi hành (không cần tên đầy đủ)

```
Select  *  from CountOrderCust('A%' ) --Loi
```

- declare @ma nvarchar(5)
- Set @ma='A%'
- select * from CountOrderCust(@ma)

The table-valued UDFs

62

Ví dụ 2 : trả về tổng số lượng của từng sản phẩm theo loại hàng nào đó.

```
CREATE FUNCTION SalesByCategory(@Categoryid Int)
RETURNS TABLE
AS
RETURN
(SELECT c.CategoryName, P. ProductName,
            SUM(Quantity) AS TotalQty
 FROM Categories c
 INNER JOIN Products p ON c.CategoryID= p. CategoryID
 INNER JOIN [Order Details] od ON p.ProductID = od.ProductID
 WHERE c.CategoryID= @Categoryid
 GROUP BY c. CategoryName,p.ProductName)
```

- Thực thi

```
SELECT * FROM SalesByCategory (1)
```

The table-valued UDFs

63

- Multistatement Table-valued UDF: là dạng phức tạp nhất. Loại hàm này xây dựng tập kết quả từ một hay nhiều câu lệnh Select
- Cú pháp:

```
CREATE FUNCTION [owner_name.]function_name
([{@parameter_name [AS] data_type [=default]} [ ,…n ]])
RETURNS @return_variable
TABLE ({column_definition | table_constraint} [ ,…n ])
[WITH { ENCRYPTION | SCHEMABINDING } ]
[AS]
BEGIN
        function_body
RETURN
END
```

The table-valued UDFs

64

```
Ví dụ 1
CREATE FUNCTION  CountOrderCust()
RETURNS  @fn_CountOrderCust  TABLE
(OrderIdent  tinyint Not null, Cust varchar(5) )
AS
Begin
    Insert @fn_CountOrderCust
    Select Count(orderid),CustomerId From Orders Group by
customerid
    Return
end
--Thi hành
Select * from CountOrderCu()
```

The table-valued UDFs

65

- Ví dụ 2

```
CREATE FUNCTION Contacts(@suppliers bit=0)
RETURNS @Contacts TABLE (ContactName nvarchar(30), Phone nvarchar(24),
ContactType nvarchar(15))
AS BEGIN
    INSERT @Contacts
    SELECT ContactName,  Phone, 'Customer'  FROM Customers
    INSERT @Contacts
    SELECT FirstName + ' ' + LastName,  HomePhone,  'Employee'
    FROM Employees
    IF @Suppliers=1
    INSERT @Contacts
    SELECT ContactName,  Phone, 'Supplier'
    FROM Suppliers
RETURN
END
```

- Thực thi

```
SELECT * FROM CONTACTS(1) ORDER BY ContactName
```

Sử dụng UDFs

66

- A scalar UDF: khi gọi luôn luôn theo cú pháp: *owner.functionname.*

Ví dụ:

```
SELECT ProductID, Total=dbo.TotalAmount(UnitPrice,  Quantity,
Discount)
FROM [Order details]
WHERE OrderID=10250
```

- A scalar UDF có thể được sử dụng trong biểu thức, trong câu lệnh SELECT hay lệnh CREATE TABLE

```
CREATE TABLE [Order Details] (
    OrderID int NOT NULL , ProductID int NOT NULL ,
    UnitPrice money NOT NULL DEFAULT (0),
    Quantity smallint NOT NULL DEFAULT (1),
    Discount real NOT NULL DEFAULT (0),
    Total AS dbo.TotalAmount(UnitPrice, Quantity, Discount))
```

Using UDFs

67

- A table-valued UDF: Có thể được gọi theo cú pháp *owner.functionname hay functionname*

  SELECT * FROM Contacts(1) ORDER BY ContactName
- Nếu table-valued function không có tham số, bạn phải sử dụng dấu()
- Nếu tham số có giá trị mặc định, bạn phải truyền giá trị vào mặc dù bạn có sử dụng từ khóa DEFAULT

  SELECT * FROM Contacts() ORDER BY ContactName

Using UDFs

68

- Viết hàm trả về danh sách các hoá đơn đã lập của một khách hàng nào đó trong một tháng năm nào đó. Thông tin gồm: Makh, TenKh, Diachi, mahd, ngaylapHD, Noichuyen, LoaiHD. Trong đó, LoaiHD được hiển thị rõ là Nhập hoặc Xuất.

User-Defined Function
Scalar Function Syntax

Bài tập:

1. Viết hàm tên SubTotalOfEmp (dạng scalar function) trả về tổng doanh thu của một nhân viên trong một tháng tùy ý trong một năm tùy ý, với tham số vào @EmplID, @MonthOrder, @YearOrder *(Thông tin lấy từ bảng [Sales].[SalesOrderHeader])*
2. Viết hàm tên là InventoryProd (dạng scalar function) với tham số vào là @ProductID và @locationID trả về số lượng tồn kho của sản phẩm trong khu vực tương ứng với giá trị của tham số *(Dữ liệu lấy từ bảng[Production].[ProductInventory]*)

User-Defined Function
Manager Function

1. Viết hàm sumofOrder với hai tham số @thang và @nam trả về danh sách các hóa đơn (SalesOrderID) lặp trong tháng và năm được truyền vào từ 2 tham số @thang và @nam, có tổng tiền >70000, thông tin gồm SalesOrderID, Orderdate, SubTotal, trong đó SubTotal =sum(OrderQty*UnitPrice).
2. Viết hàm tên SumofProduct với tham số đầu vào là @MaNCC (VendorID), hàm dùng để tính tổng số lượng (sumOfQty) và tổng trị giá (SumofSubtotal) của các sản phẩm do nhà cung cấp @MaNCC cung cấp, thông tin gồm ProductID, SumofProduct, SumofSubtotal *(sử dụng các bảng [Purchasing].[Vendor] [Purchasing].[PurchaseOrderHeader] và [Purchasing].[PurchaseOrderDetail])*

User-Defined Function
Manager Function

1. Viết hàm sumofOrder với hai tham số @thang và @nam trả về danh sách các hóa đơn (SalesOrderID) lặp trong tháng và năm được truyền vào từ 2 tham số @thang và @nam, có tổng tiền >70000, thông tin gồm SalesOrderID, Orderdate, SubTotal, trong đó SubTotal =sum(OrderQty*UnitPrice).
2. Viết hàm tên SumofProduct với tham số đầu vào là @MaNCC (VendorID), hàm dùng để tính tổng số lượng (sumOfQty) và tổng trị giá (SumofSubtotal) của các sản phẩm do nhà cung cấp @MaNCC cung cấp, thông tin gồm ProductID, SumofProduct, SumofSubtotal *(sử dụng các bảng [Purchasing].[Vendor] [Purchasing].[PurchaseOrderHeader] và [Purchasing].[PurchaseOrderDetail])*