# Chương 3
# Lập trình T-SQL
## Phần 2 – Stored procedures & User-defined functions

**Giáo trình & Tài liệu tham khảo:**

**1. Microsoft SQL Server 2008 R2 Unleashed**, Ray Rankins, Paul Bertucci, Chris Gallelli, Alex T. Silverstein, 2011, Pearson Education, Inc

2. **MS SQL Server 2012 T-SQL fundamentals**, Tizik Ben-Gan

3. https://docs.microsoft.com/

Gv. Nguyễn Như Hoa

# Nội dung

- Tạo và sử dụng hàm
- Tạo và sử dụng thủ tục

# User-Defined Stored procedures

- Viết tắt **SP**
- SP có thể nhận các tham số , và trả về nhiều giá trị cho chương trình gọi
- SP thường bao gồm **các lệnh thực hiện thao tác làm thay đổi tài nguyên** nằm ngoài phạm vi thủ tục, như : tạo table , thao tác trên dữ liệu , …

# SP
# ưu điểm

- Sử dụng SP có nhiều ưu điểm
  - Giảm lưu lượng network
  - Hỗ trợ bảo mật tốt hơn
  - Tái sử dụng mã
  - Dễ dàng duy trì
  - Cải thiện hiệu năng

  *Tham khảo : Docs.microsoft.com#**Stored Procedures (Database Engine)***

# SP
# Hạn chế

- SP không thể  chứa các lệnh : CREATE DEFAULT, CREATE RULE, CREATE/ALTER VIEW, CREATE/ALTER PROC, CREATE/ALTER FUNCTION, CREATE/ALTER TRIGGER, USE *databaseName*, …
- SP không được biên dịch cho đến khi nó được gọi chạy (lần đầu). Do vậy, mặc dù SP không có lỗi cú pháp, SP vẫn có thể bị lỗi khi chạy. Ví dụ khi chạy, các object trong SP được kiểm tra, và SP sẽ lỗi nếu object tham chiếu không tồn tại
- …

*Tham khảo : Docs.microsoft.com#**Stored Procedures (Database Engine)***

# SP
# Các loại SP

- User-defined procedure
- Temporary procedures
- System procedures
- Extended User-Defined procedures

# SP
# Tạo user-defined procedures

```
CREATE  PROC[EDURE ]  procedure_name
  @parameter  data_type  [= default ] [ OUTPUT ]
    [,...n ]

AS
        sql_statements
```

# SP
# Ví dụ 1

- Tạo SP không có INPUT parameter

```
CREATE PROCEDURE dbo.uspGetAllStore
AS
   SET NOCOUNT ON;
   SELECT BusinessEntityID, Name FROM Sales.Store

GO
```

- Sử dụng thủ tục

```
dbo.uspGetAllStore
GO
Execute dbo.uspGetAllStore
```

# SP
# Ví dụ 2

- Tạo SP có INPUT parameter

```
CREATE PROCEDURE dbo.uspGetAllStore
                                    @name nvarchar(50)
AS
   SET NOCOUNT ON;
   SELECT BusinessEntityID,  Name   FROM  Sales.Store
   WHERE  name  like  @name

GO
```

- Sử dụng thủ tục

exec **dbo.uspGetAllStore** 'New%'

# SP
# Ví dụ 3

- Tạo SP có INPUT parameter với default value

```
CREATE  PROCEDURE  dbo.uspGetAllStore
                    @name  nvarchar(50) = 'new%'
AS
   SET NOCOUNT ON;
    SELECT BusinessEntityID,  Name   FROM  Sales.Store
   WHERE  name  like  @name
GO
```

- Sử dụng thủ tục với parameter default value

```
Exec  dbo.uspGetAllStore
```

# SP
# Ví dụ 4

- Tạo SP có INPUT và OUTPUT parameter

```
CREATE  PROCEDURE dbo.GetStoreAddress  @city nvarchar(15),
                                       @rows int OUTPUT
AS
   SET NOCOUNT ON
     select *
     from sales.Store s join person.BusinessEntityAddress bea
     on s.BusinessEntityID = bea.BusinessEntityID
     join Person.Address a on bea.AddressID=a.AddressID
     where city like @city

   SELECT  @rows= @@rowcount
GO
```

# SP
# Ví dụ 4

- Sử dụng thủ tục

```
declare @rowNo  int
exec dbo.GetStoreAddress   'Dallas',
                                    @rowNo OUTPUT
select @rowNo  as CustNums
```

# SP
# Ví dụ 5

- Tạo SP  nhập dữ liệu

```
Create  proc  Sp_InsertDepartment
        @DeptName nvarchar(50),
        @GroupName nvarchar(50)
AS
Insert into
HumanResources.Department(Name, GroupName)
values (@DeptName,  @GroupName)

GO
```

# SP
# Ví dụ 5

- Sử dụng thủ tục để nhập 10 dòng vào  Department

```
declare @counter int = 1
declare @name nvarchar(50)
while @counter<=10
begin
        set @name = concat(N‘DeptName_', @counter )
        exec  Sp_InsertDepartment  @name , ‘TestGroup’
        set @counter = @counter +1
End
GO
select * from HumanResources.Department
```

# SP
# Ví dụ 6a

- Tạo SP để hiệu chỉnh dữ liệu

```
create proc Sp_updateOrder  @orderID int,
                            @productID int,  @qty smallint
as
update sales.SalesOrderDetail
set OrderQty = @qty
where SalesOrderID = @orderID
        and ProductID = @productID
go
```

- Sử dụng thủ tục

```
Exec  Sp_updateOrder 43659 , 776,  2
```

# SP
# Ví dụ 6b

- Tạo SP có lệnh return – trả về giá trị và exit

```
create proc Sp_updateOrder  @orderID int,
                            @productID int,  @qty smallint
as
Update sales.SalesOrderDetail
set OrderQty = @qty
where SalesOrderID = @orderID
        and ProductID = @productID
If @@rowcount > 0     Return 0
Else   Return 1
GO
```

# SP
# Ví dụ 6b

- Sử  dụng thủ tục

```
DECLARE @return_status  int;
EXEC  @return_status  =    Sp_updateOrder
43659 , 776,  2
If @return_status = 0
        print ‘update thành công’
```

# SP
# Ví dụ 7

- Tạo SP để xóa dữ liệu
  - Nếu table A cần xóa dữ liệu **là table ở phía 1** trong mối quan hệ 1-n với table khác , thì phải xóa cả các dòng dữ liệu ở những table có reference tới dòng cần xóa trong table A

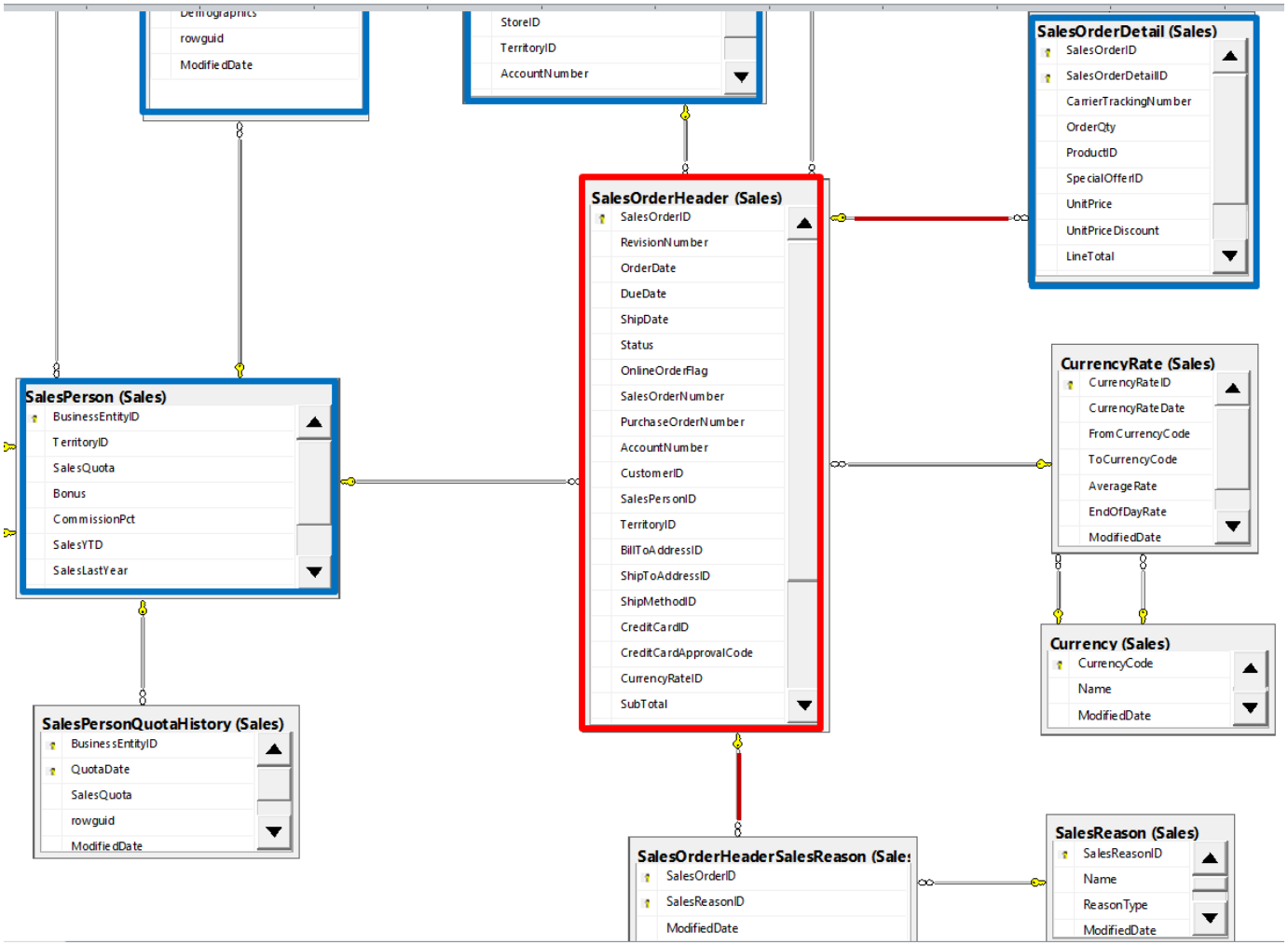
- VD: **Viết thủ tục xóa một Order trong sales.SalesOrderHeader ?**

⇒ cần xác định có table reference tới sales.SalesOrderHeader không ? Foreign constraint trong những table này có thiết lập cascade delete không ?

⇒ Nếu không có Cascade Delete (No Action) => cần xóa dòng dữ liệu trong table mà có reference tới dòng dữ liệu muốn xóa .

Ngược lại , có Cascade Delete => Không cần xóa dòng dữ liệu trong table mà có reference tới dòng dữ liệu muốn xóa .

# SP
# Ví dụ 7


rowguid
ModifiedDate
StoreID
TerritoryID
AccountNumber
SalesOrderDetail (Sales)
SalesOrderID
SalesOrderDetailID
CarrierTrackingNumber
OrderQty
ProductID
SpecialOfferID
UnitPrice
UnitPriceDiscount
LineTotal
SalesOrderHeader (Sales)
SalesOrderID
RevisionNumber
OrderDate
DueDate
ShipDate
Status
OnlineOrderFlag
SalesOrderNumber
PurchaseOrderNumber
AccountNumber
CustomerID
SalesPersonID
TerritoryID
BillToAddressID
ShipToAddressID
ShipMethodID
CreditCardID
CreditCardApprovalCode
CurrencyRateID
SubTotal
SalesPerson (Sales)
BusinessEntityID
TerritoryID
SalesQuota
Bonus
CommissionPct
SalesYTD
SalesLastYear
CurrencyRate (Sales)
CurrencyRateID
CurrencyRateDate
FromCurrencyCode
ToCurrencyCode
AverageRate
EndOfDayRate
ModifiedDate
Currency (Sales)
CurrencyCode
Name
ModifiedDate
SalesPersonQuotaHistory (Sales)
BusinessEntityID
QuotaDate
SalesQuota
rowguid
ModifiedDate
SalesOrderHeaderSalesReason (Sales
SalesOrderID
SalesReasonID
ModifiedDate
SalesReason (Sales)
SalesReasonID
Name
ReasonType
ModifiedDate

# SP
# Ví dụ 7

```
create proc Sp_delOrder   @OrderID int
as
if exists (select * from sales.SalesOrderDetail  where SalesOrderID = @OrderID)
begin
     Delete from sales.SalesOrderDetail  where SalesOrderID = @OrderID
     Delete from sales.SalesOrderHeader  where SalesOrderID = @OrderID
end
else if exists (select * from sales.SalesOrderHeader  where SalesOrderID =
@OrderID)
          Delete from sales.SalesOrderHeader  where SalesOrderID = @OrderID

if  @@ROWCOUNT>0  return 0
else return 1
```

# SP
# Ví dụ 7

- Phục hồi database trở về trang thái trước khi xóa dữ liệu ?

# SP
# Ví dụ 8

- Viết proc nhận 2 tham số  @hoNV và @năm
- Proc thực hiện kiểm tra nếu có nhân viên có @hoNV thì
  - Xuất ra thông tin nhân viên : EmployeeID, Lastname, FirstName, Title , birthdate
  - Trả về tổng trị giá các hóa  đơn do nhân viên này lập trong năm (= @nam)
- Nếu không có  nhân viên thì xuất ra thông báo ‘không có nhân viên có họ là @hoNV

# Lưu ý

Kiểm tra trong thủ tục/hàm:

1. kiểm tra tham số có hợp lệ không ?

    if exists

2. kiểm tra thao tác trên dữ liệu có thành công không ?

    @@rowcount , kết hợp với lệnh return

# Tóm tắt

- Thủ tục :
  - Tham số input (0/nhiều)
  - Tham số output (0/nhiều)
  - Return
  - Xử lý  gì
- Trong thân thủ tục , sử dụng kết hợp
  - set nocount on  => tắt message gửi tới client sau mỗi câu lệnh trong thủ tục
  - @@rowcount

  https://www.sqlshack.com/set-nocount-on-statement-usage-and-performance-benefits-in-sql-server/

# User-Defined Functions

- Viết tắt **UDFs**
- UDFs nhận tham số, thực hiện các thao tác tính toán , và **trả về giá trị**
- UDFs trả về giá trị có dạng
  - Giá trị đơn
  - Tập giá trị

# UDFs
# Ưu điểm

- UDF cho phép lập trình module hóa
- UDF cho phép thực thi nhanh hơn
- UDF được lưu trữ trong database -> cho phép giảm lưu lượng network

*Tham khảo* : docs.microsoft.com#user-defined functions

# UDFs
# Hạn chế

- UDF không được chứa bất kỳ câu lệnh nào mà tạo ra thay đổi của tài nguyên nằm ngoài phạm vi hàm (*side-effecting operator*)
  - UDF không được phép gây ra việc chỉnh sửa data trong table
  - Tuy nhiên được phép thay đổi các object trong phạm vi hàm
- Không sử dụng kiểm soát lỗi trong UDF (không dùng TRY...CATCH, @ERROR or RAISERROR trong hàm)

# UDFs
# Các loại UDF

- **Scalar Function** : trả  về một giá trị đơn
  - Kiểu dữ liệu của giá trị trả về là bất cứ giá trị nào , ngoại trừ **text**, **ntext**, **image**, **cursor**, and **timestamp**
- **Table-Valued Function** : trả về một table (một tập dữ liệu) , gồm 2 dạng
  - **Inline table-valued function**
  - **Multi-statement table-valued function**

# UDFs
# ví dụ

Tạo một UDF có chức năng trả về số lượng Store có trong 1 city tùy ý ?

```
CREATE FUNCTION dbo.custNo(@city  nvarchar(50))
RETURNS  int
AS
BEGIN
   DECLARE @custNo   int
   Select @custNo = count(*)
      from sales.Store s join person.BusinessEntityAddress bea
      on s.BusinessEntityID = bea.BusinessEntityID
      join Person.Address a on bea.AddressID=a.AddressID
      where city like @city

   RETURN @custNo
END
```

# UDFs
# ví dụ

- Sử dụng hàm

```
Select  dbo.custNo('Dallas')
Select  dbo.custNo('Nevada')
...
```

# UDFs

# Cú pháp tạo Scalar Function

```
CREATE FUNCTION  funName(@para1  type, @para2  type, …)
RETURNS   datatype
AS
BEGIN
        các lệnh t-sql
        …
        …
        RETURN  value;
END;
```

# UDFs

## Cú pháp tạo Inline table-valued Function

```
CREATE FUNCTION  funName (@para1 type, …)
RETURNS TABLE
AS
RETURN  (  SELECT  statement  );
```

*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

# UDFs

## Cú pháp tạo multi-statement table-valued Function

```
CREATE FUNCTION funName (@para type, … )
RETURNS @varname   TABLE   (
                                Col1   datatype,
                                Col2   datatype,
                                …. )
 AS
 BEGIN
        các lệnh t-sql
        ….
        ….
        RETURN
END;
```

# UDFs
# Sửa và xóa hàm

- ALTER FUNCTION *functionname*
  - Tương tự lệnh tạo function
- DROP FUNCTION *funtionname*

# UDFs
# Sử dụng hàm

- **Một Scalar Function**
  - khi gọi luôn luôn theo cú pháp: *schema.functionname()*
  - có thể được sử dụng
    - trong biểu thức
    - trong lệnh SELECT
    - trong lệnh CREATE TABLE
- **Một Table-Valued Function**
  - được sử dụng trong mệnh đề FROM của câu SELECT

# UDFs – Ví dụ 1

## Tạo và sử dụng Scalar Function

Viết hàm  tính số năm công tác của nhân viên có mã tùy ý ?

```
CREATE FUNCTION dbo.sonamCT(@empID int)
RETURNS  int
AS
BEGIN
DECLARE @no int
If exists ( select * from HumanResources.Employee where
BusinessEntityID = @empID  and CurrentFlag = 1 )
    SELECT @no = year(getdate()) - year(HireDate)
    FROM HumanResources.Employee
    WHERE BusinessEntityID = @empID

RETURN @no
END
```

# UDFs – Ví dụ 1

## Tạo và sử dụng Scalar Function

- **Sử dụng Scalar Function – cách 1**

```
Select   dbo.sonamCT(1)
Select   dbo.sonamCT(2)
```

- **Sử dụng Scalar Function – cách 2**

```
select BusinessEntityID ,
        dbo.sonamCT(BusinessEntityID ) as sonamCT
from HumanResources.Employee
```

# UDFs – Ví dụ 1

## Tạo và sử dụng Scalar Function

- **Sử dụng Scalar Function – cách 3**

```
CREATE TABLE HumanResources.EmpWorkYear (
    BusinessEntityID   int  NOT NULL,
    WorkingYear AS dbo.sonamCT(BusinessEntityID)
)
```

# UDFs – Ví dụ 2

## Tạo và sử dụng Inline Table-Value funtion

Viết hàm  hiển thị Mã KH và Số lượng hóa đơn của 1 Khách hàng tùy ý ?

```
CREATE FUNCTION  CountOrderCust (@cust varchar(5))
RETURNS TABLE
AS
RETURN (
select CustomerID, count(*) as OrderTotal
from sales.SalesOrderHeader
where CustomerID = @cust
group by CustomerID
)
```

# UDFs – Ví dụ 2

## Tạo và sử dụng Inline Table-Value funtion

- Sử dụng hàm

```
Select  *  from  CountOrderCust(29747)
```

Hoặc

```
Declare @ma int
Set @ma= 29747
Select * from CountOrderCust(@ma)
```

# UDFs – Ví dụ 3

## Tạo và sử dụng Multi statement Table-Valued Function

Viết hàm hiển thị Mã KH và Số lượng hóa đơn của các Khách hàng ?

```
CREATE FUNCTION CountCustOrder()
RETURNS @tablevar TABLE (CustID int ,
                         OrderNum int )
AS
Begin
    Insert @tablevar
    select CustomerID, count(*) as OrderTotal
    from sales.SalesOrderHeader
    group by CustomerID
    Return
End
```

# UDFs – Ví dụ 3

## Tạo và sử dụng Multi statement Table-Valued Function

- Sử dụng hàm

```
Select * from CountCustOrder()
```

# Bài tập

Bài tập 1 :

- Tạo hàm trả về đơn giá nhập của một mặt hàng với tham số truyền vào là productID
- Sử dụng hàm để tính chênh lệch (lợi nhuận) ứng với hóa đơn xuất bán có mã ‘43660’

# Bài tập

Bài tập 2 :

- 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
- Sử dụng hàm trong truy vấn thông tin các mặt hàng xuất bán trong 1 hóa đơn