CHÖÔNG 4. DATA MANIPULATION LANGUAGE (DML)

Ngoân ngöõ thao taùc döõ lieäu coù caùc leänh chính:

- Select
- Insert
- Update
- Delete
- Merge

I. LEÂÏNH SELECT : Choïn ra caùc maãu tin töø 1 hay nhieàu table.

Cuù phaùp:

Select [Distinct] [Top n / Top n Percent]
danh saùch_coät
[INTO # | ## <table aûo>
From danhsaùch_nguoàndöõlieäu
[Where ñieàu kieän]
[Group by coät_nhoùm [Having ñieàu kieän]]
[Order By <coät> [DESC] [, <coät> [DESC]]....]
[Compute AggregateFunction(coät)]

* **Caùc toaùn töû duøng trong ñieàu kieän**:

> >= (!<) < <= (!>) = <> (!=)
Is Null : WHERE Ghichu Is Null
Is Not Null : WHERE Ghichu Is Not Null
Between ... And ... : WHERE Luong Between 8000000 AND 15000000
In (danh saùch caùc trò) : WHERE Loai IN ('N', 'X')
Like : _ ñaïi dieän 1 kyù töï : WHERE Sodt Like '091%5'
% ñaïi dieän 1 string
Not And Or : thöù töï öu tieân : NOT, keá tieáp And, cuoái cuøng laø OR

* Moät coät trong leänh Select coù theå laø: 1 field trong table, 1 haèng, 1 bieåu thöùc, hay 1 haøm aggregate function (Count, Sum, Avg, Max, Min), Select-Stmt
* Moät table trong *danh saùch_table* cuûa leänh Select coù theå laø 1: table heä thoáng, user table, table aûo, view, UDF, Select-Stmt.

Ví duï: Select 'So cac nhan vien trong cong ty ', Count(*)
From Nhanvien
Ví duï: Haõy choïn ra caùc phiếu nhập hàng mà nhân viên có mã là 1 đã lập

```
Select * From Phatsinh
        Where Loai='N' and MANV = 1
```

* **SubQuery:** ñeå thi haønh 1 SubQuery ta duøng caùc toaùn töû so saùnh, vaø moät soá toaùn töû sau: Exists, Not Exists, ALL , ANY, IN

Ví duï 1: Choïn ra caùc nhân viên chöa từng lập phiếu trong công ty

```
Select  *
 From Nhanvien
 Where MANV Not IN ( Select MANV
                     From Phatsinh )
```

*** Bieåu thöùc Case**: SQL cung caáp caáu truùc Case ñeå thay theá 1 trò coù saün trong cô sôû döõ lieäu baèng 1 bieåu thöùc khaùc. Caáu truùc Case coù daïng sau:

```
Case
  When  ñk1 Then Expr1
  When  ñk2 Then Expr2
   ...
  When  ñkn Then Exprn
  Else Expr_x
End
```

Ví duï 2: Haõy hieån thò theâm coät LoaiNV ñeå xeáp loïai nhân viên dựa vào doanh số bán hàng trong tháng 3/2016 cuûa töøng nhân viên theo quy taéc sau:

Doanh số < 20000000 -> LoaiNV =1

Doanh số >=20000000 vaø <50000000 -> LoaiNV =2

Doanh số >=50000000 -> LoaiNV =3

```
SELECT MANV, DOANHSO=SUM (THANHTIEN)
      INTO #TAM
      FROM PHATSINH
           WHERE LOAI ='X'  AND MONTH(NGAY) = 3 AND YEAR (NGAY) = 2016
            GROUP BY MANV

SELECT #TAM.* ,
 LOAINV = (Case
                    When DOANHSO < 20000000  Then 1
                    When DOANHSO < 50000000 Then   2
                    Else 3
              End )
FROM  #TAM_
```

* Löu yù: Trong tröôøng hôïp ta ñaõ taïo caùc Cluster Index hoaëc Noncluster Index, vaø ta muoán lieät keâ caùc records theo thöù töï ñaõ chæ ñònh trong caùc Index thì ta vieát caâu leänh Select nhö sau: Select * from table WITH (INDEX = ten_index)

II. DUØNG DML ÑEÅ HIEÄU CHÆNH DÖÕ LIEÄU:

1. Leänh Insert : theâm 1 record môùi vaøo table

Cuù phaùp:

Insert Into <table> (danh saùch field)
Values (Danh saùch caùc giaù trò)

Ví dụ: Thêm 1 vật tư mới vào table VATTU :

```
INSERT INTO VATTU (MAVT, TENVT, DVT)
  Values ('VT01', 'Máy giặt LG cửa trên', 'Cái')
```

Lệnh Insert còn cho phép lấy dữ liệu từ các table khác chuyển vào qua cú pháp :

INSERT INTO <table> (ds field)
SELECT <ds cột> …

Ví duï 3: Leänh Insert sau seõ copy taát caû caùc record töø 1 version cuõ cuûa table Nhanvien (OldEmp) vaøo version môùi cuûa Nhanvien (coù theâm field NoiSinh vôùi giaù trò laø ' ')

```
Insert Into Nhanvien (MANV, HO, TEN, NOISINH )
 Select MANV, HO, TEN,' '
   From OldEmp
```

Löu yù: Lệnh Select Into ñeå taïo ra 1 table môùi coù caùc maåu tin laáy töø 1 hoặc nhieàu tables.

Ví duï 4: Đưa các mã nhân viên có doanh số bán hàng từ 50000000 trở lên trong tháng 3/2016 vào 1 bảng riêng tên NV_DOANHSO_CAO

```
SELECT *
  INTO  NV_DOANHSO_CAO
FROM  #TAM WHERE DOANHSO >=50000000
```

2. Leänh Update: ñeå thay ñoåi giaù trò cuûa 1 hay nhieàu coät trong table thoûa ñieàu kieän

Ví duï 5: Haõy ñoåi teân cuûa nhân viên maõ 1 sang tên mới Huyønh Vaân Dieäp

```
Update Nhanvien
      Set    Name ='Huyønh Vaân Dieäp'
      Where MANV =1
```

3. Leänh Delete: Xoùa caùc maãu tin thoûa ñieàu kieän

Ví du 14ï: Xoùa nhanvien coù maõ 2

```
Delete
  From Nhanvien Where Manv = 2
```

Ta coù theå xoùa toaøn boä dữ liệu trong table:

Delete From <teân Table>

hay

Truncate Table <teân Table>

Löu yù: Muoán xoùa haún dữ liệu laãn caáu truùc cuûa Table: **Drop Table <teân Table>**

4. Leänh Merge : cho pheùp cheøn/hoaëc caäp nhaät döõ lieäu vaøo table Target töø döõ lieäu trong Source.

Ví duï sau ñaây seõ cheøn vaät tö 'Ñöôøng Bieân Hoøa ',maõ 'VT20', DVT laø 'Kg (Source) vaøo table VATTU (Target) neáu maõ vaät tö 'VT20' chöa coù trong table VATTU; ngöôïc laïi leänh Merge seõ Update SOLUONGTON =20.

```
MERGE  INTO dbo.VATTU  AS Target
USING (SELECT  MAVT='VT20',  TENVT=N'ĐƯỜNG BIÊN HÒA',
DVT=N'KG', SLT=  0  ) AS Source
  ON Target.MAVT= Source.MAVT
  WHEN MATCHED THEN
    UPDATE SET TARGET.SOLUONGTON =20
  WHEN NOT MATCHED THEN
   INSERT (MAVT, TENVT, DVT, SOLUONGTON)
     VALUES (Source.MAVT, Source.TENVT, Source.DVT,
Source.SLT) ;
```

Löu yù: Neáu Source laø 1 table thì caâu leänh seõ laáy töøng maãu tin trong Source ñeå Insert/Update vaøo Target.

*** Lock 1 Table :**

Khi 2 user cuøng truy xuaát 1 table, coù khaû naêng tieán trình update cuûa user naøy seõ xung ñoät vôùi tieán trình Select hay update cuûa user thöù 2. Chaúng haïn nhö khi user 1 ñang thöïc hieän leänh: Select Avg (Discount) From Customer thì user thöù 2 laïi ñang thöïc hieän leänh Update:

Update Customer

Set Discount = 0.01

Where ShipCity = 'Khanh Hoa'

Trong tröôøng hôïp naøy, user 1 coù theå laáy giaù trò cuõ cuûa Discount ñeå tính trung bình. Taäp leänh Transact-SQL cung caáp 1 tuøy choïn ñeå khoùa table, haïn cheá xung ñoät xaûy ra trong quaù trình truy xuaát döõ lieäu.

Leänh sau ñaây seõ baûo veä khoûi söï xung ñoät vôùi update:

Select Avg (Discount)

From Customer **With (TabLock)**

TabLock laø khoùa chung (share lock) treân table – noù ngaên caùc user khaùc update döõ lieäu treân table ñang ñöôïc söû duïng bôûi 1 user naøo ñoù, nhöng vaãn cho pheùp ñoïc döõ lieäu.

Töông töï, phaùt bieåu update coù theå duøng From vôùi **TabLockX** ñeå thieát laäp khoùa rieâng (exclusive lock) – khoâng cho baát cöù loïai truy xuaát naøo treân table ñang duøng.

```
Update Customer
   Set Discount = .10
   From Customer With (TabLockX)
   Where ShipCity = 'Campuchia'
```

III. STORED PROCEDURE:

Một stored procedure là một nhóm các câu lệnh Transact-SQL đã được biên dịch và chứa trong SQL Server dưới một tên nào đó và được xử lý như một đơn vị (chứ không phải nhiều câu lệnh SQL riêng lẻ).

Một stored procedure có thể chứa tất cả các lệnh SQL (ngoại trừ các lệnh CREATE DEFAULT, CREATE PROCEDURE, CREATE RULE, CREATE TRIGGER, CREATE VIEW, USE). Trong stored procedure có các tham số đầu vào, các tham số đầu ra, biến cục bộ, lệnh gán, các thao tác trên cơ sở dữ liệu và cấu trúc điều khiển việc thực thi.

Các stored procedure của Microsoft® SQL Server™ trả về dữ liệu theo 4 dạng:

- Các **output parameter** có thể là:
  - ✓ dữ liệu (ký tự hay số nguyên)
  - ✓ một biến con trỏ (cursor) (các con trỏ là các tập kết quả được rút trích mỗi lần một dòng).
- Các ***mã trả về***, thường là các số nguyên.
- ***Một tập kết quả*** cho mỗi câu lệnh SELECT chứa trong stored procedure hay trong những stored procedure khác được gọi bởi stored procedure.
- ***Một con trỏ toàn cục*** có thể được tham khảo bên ngoài stored procedure.

Ví dụ: stored procedure đơn giản sau minh hoạ 3 phương cách mà stored procedure có thể trả dữ liệu về:

1. Đầu tiên stored procedure cấp một câu lệnh SELECT để trả về một tập kết quả tổng kết hoạt động bán hàng theo từng nhân viên trong bảng **PHATSINH**.

2. Sau đó cấp một câu lệnh SELECT để điền vào tham biến số lượng các phiếu xuất đã tạo.

3. Cuối cùng, stored procedure có một câu lệnh RETURN trả về một số nguyên. Việc trả về các mã thường được dùng để trả về các thông tin kiểm tra lỗi.

```
USE QLVT
    GO
    DROP PROCEDURE sp_ThongKe_XuatHang
    GO

    CREATE PROC sp_ThongKe_XuatHang
      @SoCacPX   INT    OUTPUT
    AS
    -- SELECT để trả về một tập kết quả tổng kết trị giá  của các phiếu xuất theo từng
    -- nhân viên . Kết xuất : MaNV         Tong Tri gia
    SELECT MANV,  SUM(SOLUONG*DONGIA) as TongTriGia
      FROM PhatSinh PS, CT_PHATSINH CT
      Where Loai =’X’ AND PS.PHIEU = CT.PHIEU
      GROUP BY MANV
```

```
 ORDER BY MANV

 SELECT @SoCacPX = Count(Phieu) FROM PHATSINH WHERE LOAI =’X’
-- hoặc có thể viết:
-- SET @SoCacPX = (SELECT Count(Phieu) FROM PHATSINH WHERE LOAI =’X’)

-- Trả về 0 để báo là thành công
RETURN 0
GO

-- Kiểm tra stored procedure vừa viết: ta dùng
-- Khai báo các tham biên giữ mã trả về và tham số xuất.
DECLARE @Maloi INT
DECLARE @ SoCacPX   INT

-- Thực thi thủ tục, trả về tập kết quả từ câu lệnh SELECT đầu ---- tiên.
EXEC @Maloi = sp_ThongKe_XuatHang @SoCacPX  OUTPUT
```

1. Ưu Điểm Của Stored Procedure

Stored procedure có một số ưu điểm chính như sau:

- **Hiệu quả thực thi (performance):** Khi thực thi một câu lệnh SQL thì SQL Server phải kiểm tra permission(quyền thực hiện) xem user gửi câu lệnh đó có được phép thực thi câu lệnh hay không đồng thời kiểm tra cú pháp rồi mới tạo ra một kế hoạch thực thi (excute plan) và thực thi. Nếu có nhiều câu lệnh như vậy gửi qua mạng có thể sẽ làm giảm tốc độ làm việc của server. SQL Server sẽ làm việc hiệu quả hơn nếu dùng stored procedure vì người gửi chỉ gửi một tập các câu lệnh đơn và SQL Server chỉ cần kiểm tra một lần sau đó tạo ra một kế hoạch thực thi và thực thi. Nếu stored procedure được gọi nhiều lần thì kế hoạch thực thi có thể sử dụng lại nên stored procedure đã biên dịch do vậy mà hiệu quả làm việc sẽ nhanh hơn. Ngoài ra cú pháp của các câu lệnh SQL đã được SQL Server kiểm tra trước khi save nên không cần kiểm tra lại mỗi lần thực thi.

- **Tạo khung sườn trong lập trình(Programming Framework):** Một khi stored procedure được tạo ra nó có thể được sử dụng lại. Điều này sẽ làm cho việc bảo trì(maintainability) dễ dàng hơn do việc tách rời giữa các luật business (business rules _ những luật thể hiện bên trong stored procedure) và cơ sở dữ liệu. Ví dụ nếu có một sự thay đổi nào đó về mặt logic thì ta chỉ việc thay đổi code bên trong stored procedure mà thôi. Những ứng dụng dùng stored procedure này có thể sẽ không cần thay đổi mà vẫn tươngthích với các business rule mới. Cũng giống như các ngôn ngữ lập trình khác stored procedure cho phép ta đưa vào các tham số đầu vào và trả về các tham số đầu ra đồngthời nó cũng có khả năng gọi các stored procedure khác.

- **Bảo mật:** Giả sử chúng ta muốn giới hạn việc truy xuất dữ liệu trực tiếp của một user nào đó vào một số bảng, ta có thể viết một stored procedure để truy xuất dữ liệu và chỉ cho phép user đó được sử dụng stored procedure đã viết sẵn mà thôi chứ không thể đụng đến các bảng cách trực tiếp. Ngoài ra stored procedure có thể được mã hoá (encrypt) để tăng thêm tính bảo mật.

2. Các loại Stored Procedure:

Stored procedure có thể được chia thành 5 nhóm sau:

- **System stored procedure :** Là những stored procedure chứa trong cơ sở dữ liệu **master** và thường bắt đầu bằng tiếp đầu ngữ **sp_.** Các stored procedure này thuộc loại built-in và chủ yếu dùng trong việc quản trị cơ sở dữ liệu cũng như quản trị bảo mật. Ví dụ bạn có thể kiểm tra tất cả các tiến trình đang sử dụng bởi user DomainName\Admistrators nhờ vào câu lệnh EXEC sp_who @lginame='DomainName\Administrators'. Có hàng trăm stored procedure hệ thống trong SQL Server. Và đây cũng là phần nghiên cứu trong tâm của đề tài này.
- **Local stored procedure :** Đây là loại thường dùng nhất. Chúng được chứa trong cơ sở dữ liệu do user tạo và thường được viết để thực hiện một công việc nào đó. Thông thường người ta nói đến stored procedure là nói đến loại này. Stored procedure cục bộ thường được viết bởi người quản trị hệ cơ sở dữ liệu hoặc lập trình viên.
- **Temporary stored procedure :** Là những stored procedure tương tự như stored procedure cục bộ nhưng chỉ tồn tại cho đến khi kết nối đã tạo ra chúng bị đóng lại hoặc SQL Server shutdown. Các stored procedure này được tao ra trên cơ sở dữ liệu **temdb** của SQL Server nên chúng sẽ bị xoá khi kết nối tạo ra chúng bị cắt đứt hay khi SQL Server down. **Temporary stored procedure được chia làm 3 loại: local** (bắt đầu bằng dấu #), **global** bắt đầu bằng dấu ##) và stored procedure được **tạo ra trực tiếp trên cơ sở dữ liệu tempdb.** Loại local chỉ được sử dụng bởi kết nối đã tạo đã chúng và bị xoá khi disconnect, loại global có thể được sử dụng bởi bất kỳ kết nối nào. Quyền thực thi cho loại global mặc định không thay đổi cho nhóm **public.** Loãi stored procedure được tạo trực tiếp trên cơ sở dữ liệu tempdb khác với 2 loại trên ở chỗ ta có thể set permission, chúng tồn tại kể cả sau khi kết nối tạo ra chúng bị cắt và chỉ biến mất khi SQL Server shutdown.
- **Extended stored procedure :** Đây là một loại stored procedure sừ dụng chương trình ngoại vi (external program) vốn được biên dịch thành một DLL để mở rộng chức năng hoạt động của SQL Server. Loại này thường bắt đầu bằng tiếp đầu ngư **xp_.** Ví dụ, xp_sendmail dùng để hửi mail cho một người nào đó hay xp_cmdshell dùng để chạy một DOS command … (xp_cmdshell 'dir:\c'). Nhiều loãi stored procedure mở rộng được xem như stored procedure hệ thống và ngược lại.
- **Remote stored procedure :** Những stored procedure gọi stored procedure ở server khác.

3. <u>Taïo 1 Stored Procedure</u>: Môû 1 cöûa soå Query , vaø nhaäp vaøo leänh taïo Stored Procedure theo cuù phaùp sau:

```
CREATE PROCEDURE procedure_name
[ { @parameter data_type [ = default ] [ OUTPUT ] ]

[ WITH { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION}]
[ FOR REPLICATION ]
AS sql_statement
```

Các đối số:

- ✓ *Procedure_name:* là tên của stored procedure mới. Các procedure tạm cục bộ và toàn cục được tạo bằng cách thêm dấu # phía trước tên như *#procedure_name* đối với stored procedure tạm cục bộ, *##procedure_name* đối với các stored procedure tạm toàn cục.

Tên đầy đủ của stored procedure bao gồm cả dấu # không được vượt quá 128 ký tự. Việc chỉ định người tạo stored procedure là tuỳ chọn.

- ✓ *@parameter*: Là tham số trong stored procedure . Có một hay nhiều hơn các tham số được khai báo trong câu lệnh CREATE PROCEDURE. Giá trị của mỗi tham số được khai báo phải được cấp bởi người dùng khi stored procedure được thực thi (nếu tham số không được định nghĩa giá trị mặc định). Một stored procedure có thể lên đến tối đa 2100 tham số.

  Xác định một tên tham số bằng cách thêm vào ký hiệu @ trước ký tự đầu tiên. Tên tham số phải phù hợp với các luật dành cho những định danh. Các tham số cục bộ trong procedure, các tên tham số giống nhau có thể được dùng ở những stored procedure khác nhau.

- ✓ *data_type*: Là kiểu dữ liệu của tham số.

- ✓ *default:* là giá trị mặc định dành cho tham số. Giá trị mặc định phải là một hằng và có thể là NULL. Nó có thể chứa các ký tự như %, _, [], and [^] nếu procedure sử dụng tham số với từ khoá LIKE.

- ✓ OUTPUT: Chỉ định tham số là tham số trả về.

- ✓ {RECOMPILE | ENCRYPTION | RECOMPILE, ENCRYPTION}

  RECOMPILE cho biết SQL Server không lưu lại kế hoạch dành cho stored procedure này và stored procedure này phải được tái biên dịch lại tại mỗi thời điểm chạy.

  ENCRYPTION cho biết SQL Server mã hoá bộ trong bảng **syscomments** chứa văn bản của câu lệnh CREATE PROCEDURE . Sử dụng ENCRYPTION để ngăn procedure publish stored procedure như một phần của việc nhân bản SQL Server.

- ✓ FOR REPLICATION: cho biết stored procedure được tạo ra để nhân bản không thể thực thi trên Subcriber. Tuỳ chọn này không thể sử dụng với tuỳ chọn WITH RECOMPILE .

- ✓ AS: chỉ định các hành động mà procedure sẽ thực hiện.

- ✓ *Sql_statement*: các lệnh trong Sql

Note : Tạo một stored procedure sử dụng công cụ **QUERY ANALYZER**

1) Kết nối với Servercó chứa cơ sở dữ liệu cần tạo stored procedure.
2) Trong cửa sổ Query, đánh câu lệnh Transact-SQL tạo một stored procedure mới.
3) Click nút kiểm tra cú pháp câu lệnh ( ) hay ấn tổ hợp phím Ctrl-F5 để kiểm tra cú pháp.
4) Click nút thực thi ( ) hay ấn phím F5 để tạo một stored procedure mới.

Note: Sau khi taïo 1 Stored Procedure, ñeå thi haønh noù ta vaøo cöûa soå Query Analyze và dùng lệnh Execute:

Thực thi một stored procedure bằng cách sử dụng câu lệnh Transact-SQL EXECUTE. Bạn cũng có thể thực thi một stored procedure mà không cần từ khoá EXECUTE nếu stored procedure là lệnh đầu tiên trong khối.

Cú pháp: EXEC

{[*@return_status* =] { *procedure_name* *@procedure_name_var* }
[[*@parameter* =] { *value* | *@variable* [OUTPUT] | [DEFAULT]]

[,...*n*]
[WITH RECOMPILE]

Các đối số:

- ✓ *@return_status*

  Là một tham biến kiểu int tuỳ chọn lưu trữ trạng thái trả về của một stored procedure. Tham biến này phải được khai báo trong stored procedure trước khi dùng trong một câu lệnh EXCUTE.

- ✓ *procedure_name*

  Là tên stored procedure cần thực thi.

- ✓ *@procedure_name_var*

  Tên tham biến cục bộ đã được định nghĩa thể hiện tên stored procedure.

- ✓ *@parameter*

  Là tham số dành cho stored procedure , như định nghĩa trong phần câu lệnh CREATE PROCEDURE. Khi được dùng dưới dạng *@parameter_name* = *value* , các tên tham số và các hằng không cầntheo thư tự tham số đã khai báo trong câu lệnh CREATE PROCEDURE. Tuy nhiên, nếu dạng *@parameter_name* = *value* được dùng cho bất kỳ một tham số, nó phải sử dụng cho tất cả các tham số tiếp theo.

- ✓ *value*

  Là giá trị tham số cho thủ tục. Nếu tên tham số không được xác định, các giá trị tham số phải được cấp theo đúng thứ tự đã khai báo trong câu lệnh CREATE PROCEDURE .

- ✓ *@variable*

  Là tham biến lưu một tham số hay một tham số trả về.

- ✓ OUTPUT

  Chỉ định stored procedure trả về một tham số. Sử dụng từ khoá này khi các tham biến con trỏ chính là các tham số.

- ✓ DEFAULT

  Cung cấp giá trị mặc định cho tham số đã định nghĩa trong thủ tục. Khi thủ tục yêu cầu một giá trị của tham số không có mặc định đã định nghĩa sẵn và tham số cũng không được chỉ định giá trị, một lỗi sẽ xảy ra.

- ✓ WITH RECOMPILE

  Bắt buộc một kế hoạch mới được biên dịch. Sử dụng tuỳ chọn này nếu tham số bạn đang cấp không đúng kiểu hay dữ liệu vừa thay đổi cách đặc biệt. Kế hoạch được thay đổi được dùng trong các thực thi tiếp sau. Tuỳ chọn này dược dùng cho các stored procedure mở rộng. Khuyến nghị sử dụng tuỳ chọn này cách hạn chế.

Ví duï:

```
CREATE PROCEDURE List_Cust          -- teân stored procedure
   @MinDiscount Dec(5,3)            -- tham soá
 AS                                 -- baét ñaàu thaân cuûa Stored Procedure
   Select *
        From Customer  Where Discount  >= @MinDiscount
```

Ñeå thi haønh Stored Procedure treân, ta nhaäp vaøo: **Execute List_Cust 0.02**

a. Tham soá : 1 Stored Procedure coù theå coù tôùi 2100 tham soá; moãi tham soá coù daïng:

@ten_tham_so kieåu_döõ_lieäu

Ta coù theå ñònh nghóa 1 giaù trò input maëc ñònh (default input value) trong tröôøng hôïp khi goïi 1 Stored Procedure maø laïi khoâng cung caáp giaù trò cho tham soá hình thöùc .

Cuù phaùp : @ten_tham_so kieåu_döõ_lieäu = gia_tri

Ví duï: Trong ví duï treân ta taïo 1 tham soá maëc ñònh nhö sau:

@MinDiscount Dec(5,3) = 0.01

Taát caû tham soá ñöôïc xem nhö laø input; ñeå 1 tham soá ñoùng vai troø laø **output** , ta theâm töø khoùa Output vaøo sau khai baùo cuûa noù.

Ví duï: Taïo 1 Stored Procedure teân GetCustDiscount ñeå traû veà möùc giaûm giaù cuûa 1 khaùch haøng coù maõ do ta gôûi vaøo.

```
CREATE PROCEDURE GetCustDiscount
   @MaKH Int,
   @GiamGia Dec (5,3) Output
 AS
   Set @Giamgia = (Select Discount  From Customer Where CustID = @MaKH)
```

Khi goïi Stored Procedure treân:

Execute GetCustDiscount 246900, @GiamGia Output

vôùi @GiamGia laø 1 bieán

Ví duï aùp duïng: Taïo Stored Procedure teân ListLowHighDiscount cho bieát caùc khaùch haøng coù möùc giaûm giaù <0.01 vaø möùc giaûm giaù >=0.10

c. Phaùt bieåu Return: keát thuùc procedure vaø traû veà 1 trò nguyeân cho ngöôøi goïi.

Ví duï:

```
CREATE PROCEDURE ListCustWithDiscount
   @MinDiscount Dec(5,3) = 0.0001
 AS
   If (@MinDiscount >1) Return (1)
   Select * From Customer Where Discount >=@MinDiscount
   Return (0)
```

Leänh goïi:

Execute @Status = ListCustWithDiscount 0.03

Trong Stored Procedure coù caùc leänh sau:

- Khai baùo bieán :

Declare @dem int

- Leänh gaùn :

Set @dem = 1

Ví duï:

```
CREATE Procedure GetNameAndDiscount
   @CustID int, @Name varchar (30) Output , @Discount Dec(5,3) Output
As
   Select @Name = Name, @Discount = Discount
        From Customer
        Where CustID = @CustID
   Print Cast ('Ten khach hang :'+ @Name as varchar (30) )
   Print   ('Muc giam gia :'+ Str( @discount,5,3) )
```

- Khoái leänh : Trong 1 caáu truùc neáu coù nhieàu leänh thì ñaët caùc leänh ñoù trong

```
Begin
   leänh1
   leänh 2
End
```

4. Caáu truùc trong SP:

- Caáu truùc If :

```
   If <dieukien>
        Leänh thöïc hieän khi ñieàu kieän ñuùng
   Else
        Leänh thöïc hieän khi ñieàu kieän sai
```

Ví duï:

```
If (Select Avg(Diem) From Diem Where Masv='95Q10001') < 5
   PRINT ‘Khong duoc thi tot nghiep’
Else
  PRINT ‘Duoc thi tot nghiep’
```

- Voøng laëp While :

```
   WHILE <ñk>
    BEGIN
        Leänh
        [BREAK]
        [CONTINUE]
   END
```

- Label vaø Goto:

Ví duï:

```
SkipNextStep:
```

....
Goto SkipNextStep

*** Giao tác (Transaction):** Khi thay đổi dữ liệu trên nhiều table hay nhiều records trên 1 table, ta phải đảm bảo tính nhất quán về dữ liệu trên cơ sở dữ liệu. Ví dụ ta đang thực hiện việc tăng mức giảm giá cho các customer thì tiến trình đang thi hành bị ngắt quãng vì 1 nguyên nhân nào đó (mất nguồn). Như vậy, rõ ràng là chỉ có 1 số khách hàng được tăng Discount, còn 1 số khác thì không ; điều này sẽ dẫn đến không nhất quán về dữ liệu.

Để tránh tình trạng này xảy ra, SQL Server cung cấp 1 khả năng cho phép ta phục hồi lại dữ liệu cũ nếu công việc đang thi hành bị lỗi : giao tác .

Một công việc của ta có khả năng là 1 lệnh hay nhiều lệnh SQL tác động lên nhiều Table; một công việc như vậy ta gọi là 1 giao tác. Để bắt đầu 1 giao tác, ta dùng : Begin Transaction; xác nhận 1 giao tác đã hoàn thành : Commit; hủy bỏ giao tác và trả lại dữ liệu cũ : RollBack.

Ví duï: Trong ngaân haøng, 1 giao taùc laø vieäc chuyeån soá tieàn töø taøi khoaûn tieát kieäm cuûa khaùch haøng coù taøi khoaûn `@TKCHUYEN` qua taøi khoaûn `@TKNHAN` với số tiền `@SOTIEN`:

```
CREATE PROC [dbo].[SP_CHUYENTIEN]
 @TKCHUYEN NVARCHAR (10) , @TKNHAN NVARCHAR (10), @SOTIEN BIGINT
 AS
SET XACT_ABORT ON
BEGIN TRANSACTION

BEGIN TRY
   UPDATE TAIKHOAN
     SET SODU = SODU+ @SOTIEN
     WHERE SOTK= @TKNHAN

   UPDATE TAIKHOAN
     SET SODU = SODU -  @SOTIEN
     WHERE SOTK= @TKCHUYEN
  COMMIT
END TRY

BEGIN CATCH
   ROLLBACK
   DECLARE @ErrorMessage VARCHAR(2000)
   SELECT @ErrorMessage = 'Lỗi: ' + ERROR_MESSAGE()
   RAISERROR(@ErrorMessage, 16, 1)
END CATCH
```

* Ghi chú **Về tùy chọn XACT_ABORT**: Đây là tùy chọn ở mức kết nối, chỉ có tác dụng trong phạm vi kết nối của ta. XACT_ABORT nhận hai giá trị ON và OFF (OFF là giá trị mặc định). Khi tùy chọn này được đặt là OFF, SQL Server sẽ chỉ hủy bỏ lệnh gây ra lỗi trong transaction và vẫn cho các lệnh khác thực hiện tiếp, nếu lỗi xảy ra được đánh giá là không nghiêm trọng. Còn khi XACT_ABORT được đặt thành ON, SQL Server mới cư xử đúng như mong đợi – khi gặp bất kỳ lỗi nào nó hủy bỏ toàn bộ transaction và quay lui trở lại như lúc ban đầu.

Baøi taäp:

Cho cô sôû döõ lieäu QLVT vôùi caùc table sau trong SQL Server :

- **Nhanvien (**MANV, HOTEN, DIACHI, NGAYSINH, LUONG, GHICHU)
- **Kho** (MAKHO, TENKHO,)
- **Vattu** (MAVT, TENVT, DVT)
- **Phatsinh** (PHIEU, NGAY, LOAI, HOTEN, THANHTIEN, MANV)
- **CT_Phatsinh** (PHIEU, MAVT, SOLUONG, DONGIA, MAKHO)

1. Vieát 1 SQL Script ñeå taïo caáu truùc cuûa csdl QLVT

2. Taïo 1 SP ñeå huûy phieáu coù soá phieáu do ta gôûi vaøo.

3. *Tính soá löôïng toàn kho cuûa töøng vaät tö*