CHƯƠNG 4. DATA MANIPULATION LANGUAGE (DML)

Ngôn ngữ thao tác dữ liệu có các lệnh chính:

* Select
* Insert, Update, Delete, Merge

**I. LÊNH SELECT** : Chọn ra các mẫu tin từ 1 hay nhiều table.

Cú pháp:

Select [Distinct] [Top n / Top n Percent]

*danh sách\_cột*

[ INTO # | ## <table ảo> ] From danhsách\_nguồndữliệu [Where điều kiện]

[FOR XML PATH(‘’)]

[Group by cột\_nhóm [Having điều kiện]]

[Order By <cột> [DESC] [, <cột> [DESC] ]…. ]

### \* Các toán tử dùng trong điều kiện:

> >= (!<) < <= (!>) = <> (!=)

Is Null : WHERE Ghichu Is Null

Is Not Null : WHERE Ghichu Is Not Null

Between … And … : WHERE Luong Between 8000000 AND 15000000

In (danh sách các trị) : WHERE Loai IN (‘N’, ‘X’) Like : \_ đại diện 1 ký tự : WHERE Sodt Like ‘091%5’

% đại diện 1 string

Not And Or : thứ tự ưu tiên thực hiện: NOT, kế tiếp And, cuối cùng là OR

* Một cột trong lệnh Select có thể là: 1 field trong table, 1 hằng, 1 biểu thức, hay 1 hàm

aggregate function (Count, Sum, Avg, Max, Min), Select-Stmt

* Một table trong danhsách\_nguồndữliệu của lệnh Select có thể là 1: table hệ thống, user table, table ảo, view, UDF, Select-Stmt.

*Ví dụ*: Ěếm số nhân viên trong table Nhanvien

Select 'So cac nhan vien trong cong ty ', Count(\*) From Nhanvien

*Ví dụ*: Hãy chọn ra các phiếu nhập hàng mà nhân viên có mã nhân viên = 1 đã lập

Select \* From PhatSinh

Where MANV = 1 and Loai=’N’

\* **SubQuery:** để thi hành 1 SubQuery ta dùng các toán tử so sánh, và một số toán tử tập hợp

sau: Exists, Not Exists, ALL , ANY, IN

Ví dụ : Chọn ra các nhân viên chưa từng lập phiếu trong công ty Select \*

From Nhanvien

Where MANV **Not IN ( Select MANV**

### From Phatsinh )

Ví dụ: Liệt kê Hoten, Luong của các nhân viên có lập phiếu nhập cho kho ‘TK’

SELECT HOTEN=HO+ ' '+ TEN, LUONG

FROM NhanVien

WHERE 'TK' IN (SELECT MAKHO FROM PhieuNhap WHERE **NhanVien.MANV=PhieuNhap.MANV**)

Hoặc:

SELECT HOTEN=HO+ ' '+ TEN, LUONG

FROM NhanVien

WHERE Exists (SELECT MAPN FROM PhieuNhap WHERE NhanVien.MANV=PhieuNhap.MANV AND MAKHO='TK')

\* **FOR XML PATH(‘’)**: dữ liệu kết xuất theo dạng XML, mỗi field là 1 thẻ. Ví dụ với lệnh: SELECT MAPN, MAVT FROM CTPN FOR XML PATH('')

Thì kết quả :

<MAPN>PN01 </MAPN>

<MAVT>MG01</MAVT>

<MAPN>PN01 </MAPN>

<MAVT>MS01</MAVT>

<MAPN>PN02 </MAPN>

<MAVT>MS01</MAVT>

<MAPN>PN02 </MAPN>

<MAVT>MU01</MAVT>

Ta có thể áp dụng For XML Path để nối dữ liệu nhiều dòng thuộc 1 cột thành 1 ô với cùng cột nhóm id.

Ví dụ: Ta muốn nối các vật tư thuộc cùng 1 phiếu nhập vào 1 ô thì thực hiện lệnh sau:

SELECT MAPN, DS\_VT =

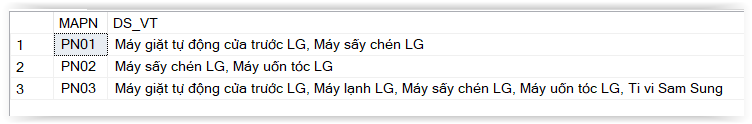
STUFF( (SELECT DISTINCT ', ' + TENVT

FROM CTPN B INNER JOIN VATTU ON B.MAVT=VATTU.MAVT WHERE b.MAPN = a.MAPN

FOR XML PATH('')

) , 1, 2, '') FROM CTPN A GROUP BY MAPN

Kết quả sau khi thực hiện:



**\* Biểu thức Case**: SQL cung cấp cấu trúc Case để thay thế 1 trị có sẵn trong cơ sở dữ liệu bằng 1 biểu thức khác. Cấu trúc Case có dạng sau:

### Case

**When đk1 Then Expr1 When đk2 Then Expr2**

**…**

**When đkn Then Exprn**

**Else Exprx End**

Ví dụ: Hãy hiển thị thêm cột LoaiNV để xếp lọai nhân viên dựa vào doanh số bán hàng trong tháng 3/2016 của từng nhân viên theo quy tắc sau:

Doanh số < 20000000 -> LoaiNV =1

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

Doanh số >=50000000 -> LoaiNV =3 SELECT MANV, DOANHSO=SUM (THANHTIEN)

INTO #TAM

FROM PHIEUXUAT

WHERE MONTH(NGAY) = 3 AND YEAR (NGAY) = 2016 GROUP BY MANV

SELECT #TAM.\* , LOAINV = (Case

End ) FROM #TAM

When DOANHSO < 20000000 Then 1

When DOANHSO < 50000000 Then 2

Else 3

\* Lưu ý: Trong trường hợp ta đã tạo các Noncluster Index, và ta muốn liệt kê các records theo thứ tự đã chỉ định trong các Index thì ta viết câu lệnh Select như sau:

Select \* from table WITH (INDEX = ten\_index)

### DÙNG DML ĚỂ HIỆU CHỈNH DỮ LIỆU:

* 1. **Lệnh Insert** : thêm 1 record mới vào table

Cú pháp:

### Insert Into <table> (danh sách field)

**Values (Danh sách các giá trị)**

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 dụ: Lệnh Insert sau sẽ copy tất cả các record từ 1 version cǜ của table Nhanvien (OldEmp) vào version mới của Nhanvien (có thêm field NoiSinh với giá trị là ‘ ‘)

**Insert Into** Nhanvien (MANV, HO, TEN, NOISINH )

**Select** MANV, HO, TEN,’ ‘

From OldEmp

Lưu ý: Lệnh Select Into để tạo ra 1 table mới có các mẩu tin lấy từ 1 hoặc nhiều tables.

Ví dụ: Ěưa 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

* 1. **Lệnh Update**: để thay đổi giá trị của 1 hay nhiều cột trong table thỏa điều kiện Ví dụ: Hãy đổi tên của nhân viên có mã số = 1 sang tên mới HuǶnh Vân Diệp **Update Nhanvien**

### Set Name = N’HuǶnh Vân Diệp’

**Where MANV =1**

1. **Lệnh Delete:** Xóa các mẫu tin thỏa điều kiện

Ví du : Xóa nhân viên có mã số = 2

Delete From Nhanvien Where Manv = 2

Ta có thể lưu dữ liệu đã xóa vào 1 bảng tạm theo cú pháp **Delete … Output … Into**:

declare @del\_table TABLE (MA\_NV INT, HO NVARCHAR(40), TEN NVARCHAR(10), LUONG FLOAT)

DELETE NhanVien

--**OUTPUT** deleted.MANV, deleted.HO, deleted.TEN, deleted.LUONG **INTO @del\_table**

WHERE LUONG<=5000000

SELECT \* FROM @del\_table

SELECT COUNT(\*) FROM @del\_table

**Ta có thể xóa toàn bộ dữ liệu trong table**: Delete From <tên Table>

hay

Truncate Table <tên Table>

Lưu ý:

* Lệnh Truncate Table <tên Table> thực thi nhanh hơn lệnh Delete From <tên Table>

vì xóa theo trang dữ liệu.

* Muốn xóa hẳn dữ liệu lẫn cấu trúc của Table: **Drop Table <tên Table>**

1. **Lệnh Merge :** Lệnh Merge kết hợp các câu lệnh INSERT, UPDATE và DELETE vào trong 1 lệnh duy nhất, tùy thuộc vào sự tồn tại của một bản ghi.

Ví dụ sau đây sẽ chèn vật tư ‘Ěường Biên Hòa ’,mã ‘VT20’, DVT là ‘Kg’ (Source) vào table VATTU (Target) nếu mã vật tư ‘VT20’ chưa có trong table VATTU; ngược lại lệnh Merge sẽ Update SOLUONGTON =20.

MERGE INTO dbo.VATTU AS Target

USING (SELECT MAVT='VT20', TENVT=N'ĚƯỜNG 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) ;

### Lưu ý: Nếu Source là 1 table thì câu lệnh sẽ lấy từng mẫu tin trong Source để

**Insert/Update vào Target.**

1. **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,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 lên 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)
    - một biến con trỏ (các con trỏ là các tập kết quả được rút trích mỗi lần một dòng).
  + Mã ***trả về của lệnh return (***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 tòan cục*** có thể được tham khảo bên ngoài stored procedure.

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

1. Ěầu tiên stored procedure dùng 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, một câu lệnh SELECT khác để gán vào tham biến số lượng các phiếu xuất đã tạo.
3. Cuối cùng, stored procedure trả về một số nguyên qua phát biểu RETURN. Việc trả về số nguyên 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 khai báo các biến, và gọi như sau:

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 Ěiểm 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 stored procedure đã biên dịch do vậy 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. Ěiều 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.
  + **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, user không thể đụng đến các bảng trực tiếp. Ngoài ra stored procedure có thể được mã hóa (encrypt) để tĕng thêm tính bảo mật.

## 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 cǜng như quản trị bảo mật. Ví dụ bạn có thể kiểm tra tất cả các tiến trình đang sử dụng bởi user DomainName\Administrators nhờ vào câu lệnh EXEC sp\_who

@lginame=’DomainName\Administrators’. Có hàng trĕm stored procedure hệ thống

trong SQL Server .

* + **Local stored procedure :** Ěây 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 tạo ra chúng bị đóng lại.. Các stored procedure này được tạo ra trên cơ sở dữ liệu **temdb** của SQL Server nên chúng sẽ bị xoá khi kết nối tạo ra chúng bị ngắt 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 và bị xóa khi disconnect, loại global có thể được sử dụng bởi bất kǶ kết nối nào. Quyền thực thi cho loại global mặc định không thay đổi cho nhóm **public.** Lỗi 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 ngắt và chỉ biến mất khi SQL Server shutdown.

* + **Extended stored procedure :** Ěây là một loại stored procedure sừ dụng chương

trình ngoại vi (external program) vốn được biên dịch thành một DLL để mở rộng chức nĕng hoạt động của SQL Server. Loại này thường bắt đầu bằng tiếp đầu ngữ **xp\_.** Ví dụ, xp\_sendmail dùng để gửi mail cho một người nào đó hay xp\_cmdshell dùng để chạy một DOS command … (xp\_cmdshell ‘dir:\c’).

* + **Remote stored procedure :** Gọi stored procedure ở server khác, lúc này ta phải tạo

Link Server đến Server chứa Stored Procedure.

1. **Tạo 1 Stored Procedure**: Mở 1 cửa sổ Query , và nhập vào lệnh tạo Stored Procedure

theo cú pháp sau:

**CREATE PROCEDURE *procedure\_name***

**[ { @*parameter data\_type* [ = *default* ] [ OUTPUT ] ]**

**[ WITH { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION}]**

#### 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 tuǶ chọn.
  + *@parameter*: Là tham số trong stored procedure . Giá trị của mỗi tham số được khai báo phải được cung cấp bởi người dùng khi stored procedure được thực thi (nếu tham số không được định nghƿa 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*: 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 thuộc loại tham biến.
  + {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 hóa trong bảng **syscomments** chứa vĕn bản của câu lệnh CREATE PROCEDURE .

* + *Sql\_statement*: các lệnh trong Sql

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

1. Kết nối với Server có 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 .
4. Click nút thực thi () hay ấn phím F5 để tạo một stored procedure mới.

Note: Sau khi tạo 1 Stored Procedure, để thi hành nó ta vào cửa sổ Query Analyze và dùng

lệnh Execute:

Thực thi một stored procedure bằng cách sử dụng câu lệnh Transact-SQL EXECUTE.

**Cú php: EXEC**

**{[ *@return\_status* = ] { *procedure\_name* | *@procedure\_name\_var* } [ [ *@parameter* = ] { *value* | *@variable* [ OUTPUT ] ]**

**[ ,...*n* ]**

### [ WITH RECOMPILE ]

**Các đối số:**

* + *@return\_status:* biến kiểu int nhận trạng thái trả về của một stored procedure bởi

lệnh Return.

* + *procedure\_name:* tên stored procedure cần thực thi.
  + @*procedure\_name\_var:* biến cục bộ lưu tên của stored procedure muốn chạy.
  + *@parameter*

Là tham số trong stored procedure như định nghƿa trong phần câu lệnh CREATE PROCEDURE. Khi được dùng dưới dạng *@parameter\_name* **=** *value* , cc tn tham số và các hằng không cầntheo thư tự tham số đ khai bo trong cu lệnh CREATE PROCEDURE. Tuy nhin, nếu dạng *@parameter\_name* **=** *value* được dùng cho bất kǶ một tham số, nó phải sử dụng cho tất cả các tham số tiếp theo.

* + *Value:* giá trị gởi cho tham số. 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:* gởi giá trị trong biến @variable cho tham số
  + OUTPUT: chỉ định đây là tham số thực biến
  + WITH RECOMPILE

Bắt buộc một kế hoạch mới được biên dịch. Sử dụng tuǶ chọn này nếu tham số bạn đang cấp không đúng kiểu hay dữ liệu vừa thay đổi cách đặc biệt. Kế hoạch được

Ví dụ:

thay đổi được dùng trong các thực thi tiếp sau. TuǶ chọn này dược dùng cho các stored procedure mở rộng. Khuyến nghị hạn chế sử dụng tuǶ chọn này.

CREATE PROCEDURE List\_Cust -- tên stored procedure

@MinDiscount Dec(5,3) = 0.1 -- tham số

AS -- bắt đầu phần thân của Stored Procedure

Select \*

From Customer Where Discount >= @MinDiscount

Ěể thi hành Stored Procedure trên, ta nhập vào: **Execute List\_Cust 0.2**

**a. Tham số** : 1 Stored Procedure có thể có tới 2100 tham số; mỗi tham số có dạng:

@ten\_tham\_so kiểu\_dữ\_liệu

Ta có thể định nghƿa 1 giá trị input mặc định (default input value) trong trường hợp khi gọi 1 Stored Procedure mà không cung cấp giá trị cho tham số hình thức .

Cú pháp : @ten\_tham\_so kiểu\_dữ\_liệu = gia\_tri

Ví dụ: Trong ví dụ trên ta tạo 1 tham số mặc định như sau:

@MinDiscount Dec(5,3) = 0.01

Tất cả tham số được xem như là input; để 1 tham số đóng vai trò là **output** , ta thêm từ khóa Output vào sau khai báo của nó.

Ví dụ: Tạo 1 Stored Procedure tên GetCustDiscount để trả về mức giảm giá của 1 khách hàng có mã do ta gởi vào.

CREATE PROCEDURE GetCustDiscount

@MaKH Int,

### @GiamGia Dec (5,3) Output

AS

**Set @Giamgia =** (Select Discount From Customer Where CustID = @MaKH)

Khi gọi Stored Procedure trên:

Execute GetCustDiscount 246900, @GiamGia Output

với @GiamGia là 1 biến

Ví dụ áp dụng: Tạo Stored Procedure tên ListLowHighDiscount cho biết các khách hàng có mức giảm giá <0.01 và mức giảm giá >=0.1

**c. Phát biểu Return**: kết thúc procedure và trả về 1 trị nguyên cho người gọi. Ví dụ:

CREATE PROCEDURE ListCustWithDiscount

@MinDiscount Dec(5,3) = 0.0001 AS

If (@MinDiscount >1) Return 1

Select \* From Customer Where Discount >=@MinDiscount Return (0)

Lệnh gọi:

Execute @Status = ListCustWithDiscount 0.03

### Trong Stored Procedure có các lệnh sau:

* Khai báo biến :

Declare @dem int

* Lệnh gán :

Set @dem = 1

### Ví dụ:

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

* Khối lệnh : Trong 1 cấu trúc nếu có nhiều lệnh thì đặt các lệnh đó trong

### Begin

lệnh1 lệnh 2

### End

1. **Cấu trúc trong SP:**

* Cấu trúc If :

If <dieukien>

Lệnh thực hiện khi điều kiện đúng

Else

Ví dụ:

Lệnh thực hiện khi điều kiện sai

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

Else

PRINT ‘Duoc thi tot nghiep’

* Vòng lặp While : WHILE <đk> BEGIN

Lệnh [BREAK] [CONTINUE]

END

* Label và Goto: Ví dụ: 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 tĕng 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 tĕng Discount, còn 1 số khách hàng khác thì không ; điều này sẽ dẫn đến không nhất quán về dữ liệu.

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

Một công việc của ta có khả nĕng là 1 lệnh hay nhiều lệnh SQL tác động lên nhiều Table; một công việc như vậy ta gọi là 1 giao tác. Ěể 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 cǜ : RollBack.

Ví dụ: Trong ngân hàng, 1 giao tác là việc chuyển số tiền từ tài khoản tiết kiệm của khách hàng có tài khoản @TKCHUYEN qua tài khoản @TKNHAN với số tiền @SOTIEN:

CREATE PROC [dbo].[SP\_CHUYENTIEN]

@TKCHUYEN NVARCHAR (10) , @TKNHAN NVARCHAR (10), @SOTIEN BIGINT AS

SET XACT\_ABORT ON BEGIN TRAN

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**: Ěây 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 hoặc 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. Khi XACT\_ABORT được đặt thành ON, SQL Server mới xử lý đúng như mong đợi – khi gặp bất kǶ lỗi nào nó hủy bỏ toàn bộ transaction và đưa dữ liệu quay lui về trạng thái như lúc ban đầu.

Nếu ta muốn lưu kết quả trả về bởi lệnh Select trong SP để xử lý tiếp thì tạo 1 table tạm, sau đó gọi lệnh INSERT INTO … EXEC :

* + Tạo table tạm:

CREATE TABLE #AB (

HOCKY INT, HELOP INT, TONG\_GGQC FLOAT

)

* Ěưa kết quả trả về vào Table tạm:

INSERT INTO #AB (HOCKY, HELOP, TONG\_GGQC)

EXEC SP\_THONG\_KE\_GGQC\_HVCS\_THEO\_HKY\_HEDT\_TRONG\_NK '2021 - 2022'

– SP này trả về 3 cột, nên ta tạo table tạm #AB để lưu dữ liệu trả về của SP