NGÔN NGỮ LẬP TRÌNH TRONG SQL SERVER

Phan Hiền

CĂN BẢN NGÔN NGỮ

- Kiểu dữ liệu
  - Char, VarChar, Nchar, NvarChar
  - Bit, TinyInt, SmallInt, Int, BigInt
  - Float, SmallMoney, Money, Real
  - DateTime
  - Text, Image, XML
- Toán tử : + - * / %

CĂN BẢN NGÔN NGỮ

- Cú pháp khai báo

  declare @tên_biến kiểu_dữ_liệu [,… n]

- Gán giá trị

  set @tên_biến =

  giá_trị | biểu_thức | @biến | hàm

VD:

```
declare @x int, @y int
set @y = 5
set @x = @y + 3
```

CĂN BẢN NGÔN NGỮ

- <u>Cấu trúc lệnh IF</u>

```
if  (điều_kiện)
      lệnh …. | khối_lệnh
else
      lệnh …. | khối_lệnh

khối_lệnh :=        begin
                          lệnh … | khối_lệnh
                    end
```

CĂN BẢN NGÔN NGỮ

- Cấu trúc lệnh WHILE

  while (điều_kiện)

  lệnh | khối_lệnh

- Lệnh ngắt vòng lặp

  break

  continue

CĂN BẢN NGÔN NGỮ

- Biến Cursor

  declare tên_biến_cursor cursor

  for câu_truy_vấn

- Sử dụng

  open tên_biến_cursor

  ….

  close tên_biến_cursor

- Hủy cursor

  deallocate tên_biến_cursor

CĂN BẢN NGÔN NGỮ

- <u>Di chuyển Cursor</u>

  fetch *định_vị*

  from tên_biến_cursor

  into @tên_biến [,… n]

  định_vị := next | prior | last | first |

  absolute (giá_trị | biến)

  relative (giá_trị | biến)

CĂN BẢN NGÔN NGỮ

- <u>Trạng thái Cursor</u>

  @@fetch_status

  =0 : Đang trong dòng dữ liệu (lần đi kế tiếp thành công)

  ≠0 : Ngoài dòng dữ liệu (lần đi kế tiếp không thành công)

CĂN BẢN NGÔN NGỮ

- <u>Ví dụ</u> Tính tổng số chẵn từ 1 -> 100

```
Declare @t int, @x int
Set @t = 0 ; Set @x = 1
While (@x <= 100)
begin
    if  ((@x % 2) = 0)
            set @t = @t + @x
    set @x = @x + 1
end
Print @t
```

CĂN BẢN NGÔN NGỮ

- <u>Ví dụ</u> In các sinhvien(masv char(5),tensv char(10))

```
Declare sv cursor for select * from sinhvien
Open sv
Declare @ma char(5),@ten char(10)
Fetch next from sv into @ma,@ten
While (@@fetch_status = 0)
begin
    print @ma + ‘ : ‘ + @ten
     Fetch next from sv into @ma,@ten
end
Close sv; Deallocate sv
```

THỦ TỤC (STORE PROCEDURE)

- <u>Tạo lập thủ tục</u>

  create procedure tên_thủ_tục
  @tên_tham_số kiểu_dữ_liệu *loại*
  [,…n]
  as
  lệnh | khối_lệnh

  loại := input (không cần ghi)
  output

THỦ TỤC (STORE PROCEDURE)

- Thực thi thủ tục

  exec tên_thủ_tục

  giá_trị | @biến [output] [,…n]

- Xóa thủ tục

  Drop procedure tên_thủ_tục

- Thay đổi thủ tục

  Alter procedure tên_thủ_tục

  ……..

THỦ TỤC (STORE PROCEDURE)

- <u>Ví dụ</u> Viết thủ tục xóa các sinh viên theo thành phố

  sinhvien (masv char(5), tp char(5))

```
create procedure xoasinhvien
    @tp char(5)
as
begin
    delete from sinhvien where tp = @tp
end

exec  xoasinhvien  'HCM'
```

THỦ TỤC (STORE PROCEDURE)

- <u>Ví dụ</u> Viết thủ tục đếm xem có bao nhiêu sinh viên theo thành phố.

```
create procedure dem @tp char(5), @t int output  as
begin
      select @t = count(*) from sinhvien
      where tp = @tp
end

declare @tong int
exec  dem  'HCM' , @tong output
print @tong
```

HÀM (FUNCTION)

- <u>Tạo lập hàm</u>

```
create function tên_hàm
( @tên_tham_số  kiểu  [,…n] )
returns  kiểu_trả_về
as
    lệnh | khối_lệnh
```

HÀM (FUNCTION)

- Thực thi hàm

  = tên_hàm (

  giá_trị | @biến [,…n])

- Xóa hàm

  Drop function tên_hàm

- Thay đổi hàm

  Alter function tên_hàm

  ……..

HÀM (FUNCTION)

- <u>Ví dụ</u> Viết hàm đếm xem có bao nhiêu sinh viên theo thành phố.

```
create function dem (@tp char(5))  returns int
as
begin
    declare @t int
    select @t = count(*) from sinhvien where tp = @tp
    return @t
end

declare @tong int
set @tong = dbo.dem('HCM')
```

HÀM (FUNCTION)

- <u>Ví dụ</u>

  Viết hàm sinh ra mã sinh viên tự động theo quy tắc

  – Mã sinh viên có dạng: BA0001

  'BA' : quy định (luôn có)

  0001 : là số

  VD:

  Hiện tại sinh viên có mã cao nhất là BA0024

  Thì sinh mã mới là BA0025

HÀM (FUNCTION)

```
Create function sinhkhoa () returns char(6) As
Begin
    declare @max int
    select
        @max = max(cast(substring(masv,3,4) as int)) + 1
        from sinhvien
    declare @s char(8), @s1 char(6)
    set @s = '0000' + cast(@max as char(4))
    set @s1 = 'BA' + right(rtrim(@s),4)
    return @s1
end
```

HÀM (FUNCTION)

- Ví dụ với Table Function

```
create function laydssv (@malop char(5))
returns TABLE
as
    return (
        select masv,tensv from sinhvien
        where malop = @malop
        )

select * from laydssv('QT1')
```

HÀM (FUNCTION)

- Ví dụ với Table Function

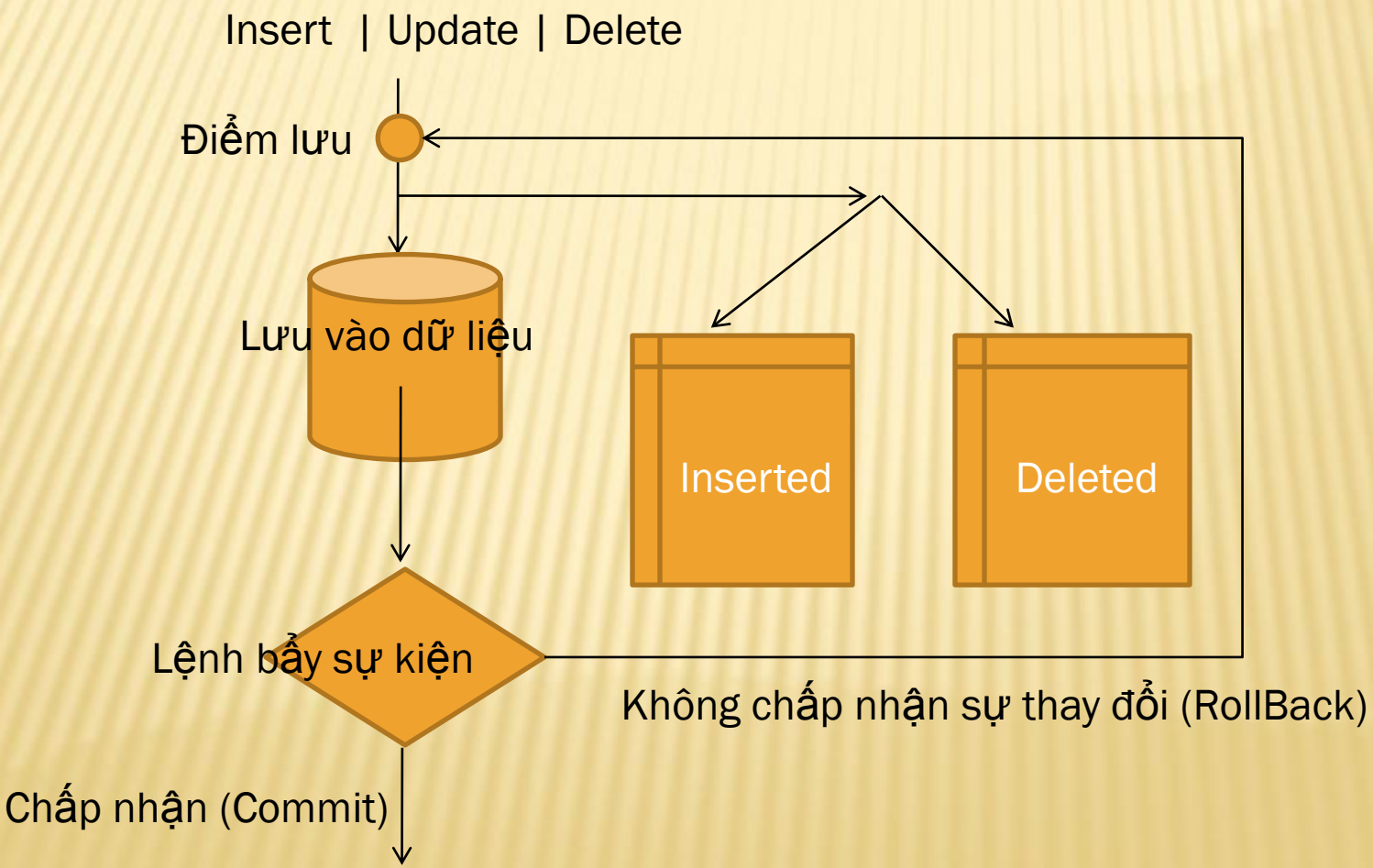
```
create function laydssv1 (@malop char(5))
        returns  @btam table(masv char(5),tensv char(20))
as
begin
        insert into @btam  select masv,tensv from sinhvien
                where malop = @malop
        return
end

select * from laydssv1('QT1')
```

BẪY SỰ KIỆN (TRIGGER)

- Bẩy lệnh được phát sinh sau những hành vi thêm mới hay thay đổi, xóa trên bảng.
  - Có thể hủy các cập nhập trên dữ liệu
- Bẩy lệnh được phát sinh để thay thế những hành vi thêm, đổi, xóa.
- Bẩy sự kiện lưu giử tách rời giá trị mới được đưa vào và giá trị cũ được xóa bỏ.
  - Dùng bảng tạm Inserted và deleted
- Bẩy sự kiện còn áp dụng cho *Login*.

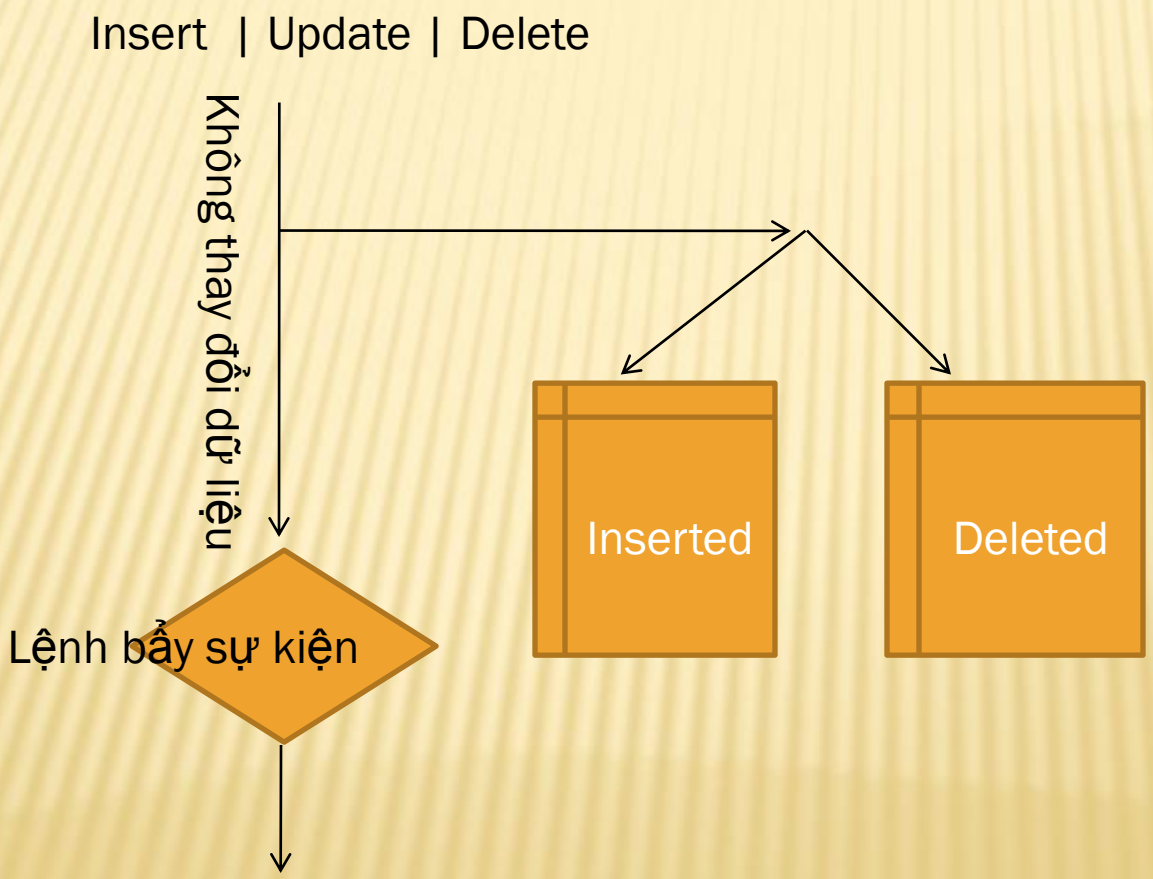
BẪY SỰ KIỆN (TRIGGER)

- Loại trigger FOR



BẪY SỰ KIỆN (TRIGGER)

- Loại trigger INSTEAD OF



BẪY SỰ KIỆN (TRIGGER)

- <u>Bảng tạm cho bẫy sự kiện</u>

inserted : Lưu những thông tin sắp được đưa vào dữ liệu

deleted : Lưu những thông tin đã có và chuẩn bị được thay thế

Bảng tạm có cấu trúc giống bảng đặt sự kiện.

<u>VD:</u>

Cập nhập sinh viên mã ‘BA0002’ với thành phố ‘HCM’ thành ‘HN’.

=> Vào bảng inserted là bộ (‘BA0002’,’aaa’,’HN’)

=> Vào bảng deleted là bộ (‘BA0002’,’aaa’,’HCM’)

BẪY SỰ KIỆN (TRIGGER)

- update(tên_cột) :

  Trả về kết quả True / False nếu cột đó bị cập nhập.

- columns_updated() :

  Trả về một số xác định các cột được cập nhập.

  Ví dụ: Bang (c1,c2,c3,c4)

  Nếu c2 được cập nhật => 0010: kết quả 2

  Nếu c1,c3,c4 được cập nhật =>1101: kết quả 13

BẪY SỰ KIỆN (TRIGGER)

- Thay đổi thông số cho phép đệ quy trực tiếp

  alter database tendatabase set recursive_triggers { on | off }

  (**Update T1**, kích chạy **Trigger 1**. Trigger 1 chạy **update T1** lần nữa, kích chạy **Trigger 1**…)

- Thiết lập giới hạn lồng nhau (đệ quy gián tiếp)

  exec sp_configure 'Nested Triggers' *n*

  (**Update T1**, kích chạy **Trigger 1**. Trigger 1 chạy **update T2**, kích chạy **Trigger 2**. Trigger 2 chạy **update T1**, kích chạy **Trigger 1**. …)

  Lồng nhau được tối đa là 32 cấp độ.

BẪY SỰ KIỆN (TRIGGER)

- <u>Tạo trigger</u>

  create trigger tên_trigger

  on tên_bảng

  {for|instead of} {insert|delete|update}

  as

  lệnh | khối_lệnh

- <u>Xóa và thay đổi</u>

  Alter | Drop trigger tên_trigger

  ……..

BẪY SỰ KIỆN (TRIGGER)

- Tạo trigger cho bảng sinhvien (masv, tensv, malop) thỏa mãn điều kiện một lớp không quá 20 người.

```
Create trigger tssv on sinhvien for insert,update As
Begin
    declare @malop char(5), @ts int
    select @malop = malop from inserted
    select @ts = count(*) from sinhvien
            where malop=@malop
    if (@ts > 20)
            rollback transaction
end
```

BẪY SỰ KIỆN (TRIGGER)

- Tạo trigger cho bảng sinhvien (masv, tensv, trangthai) thỏa mãn điều kiện khi xóa một sinh viên tức thay đổi trạng thái từ 0 thành 1.

```
Create trigger tssv on sinhvien instead of delete As
Begin
    update sinhvien set trangthai = 1
    where          masv in
                   (select masv from deleted)
end
```