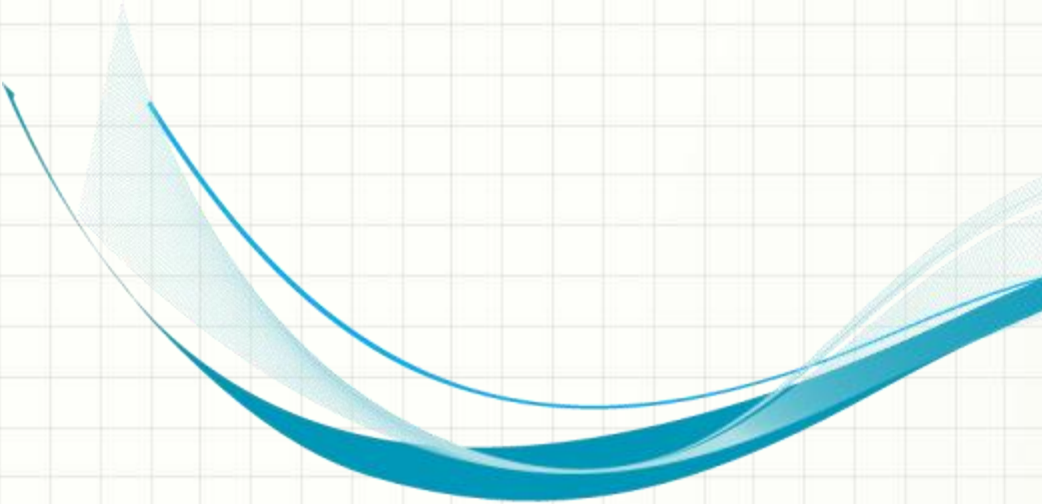


# **PHẦN II: LẬP TRÌNH VỚI T – SQL**

## **CHƯƠNG V : NGÔN NGỮ LẬP TRÌNH T - SQL**

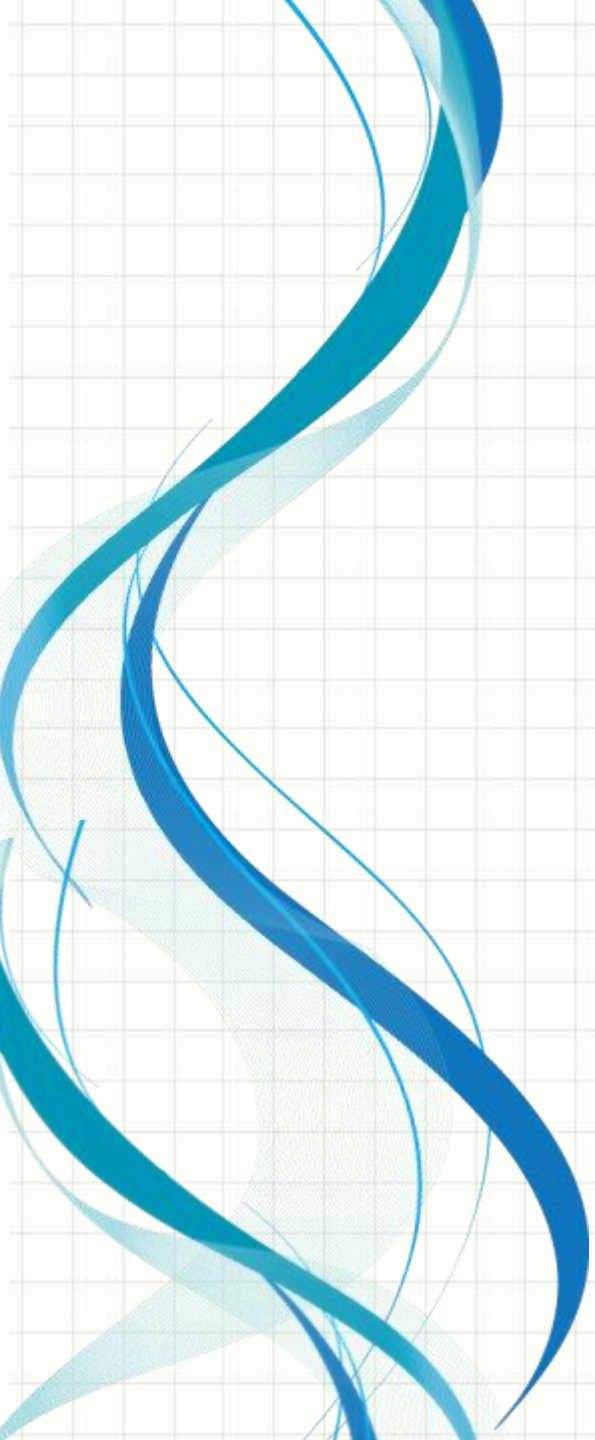
Phù Khắc Anh



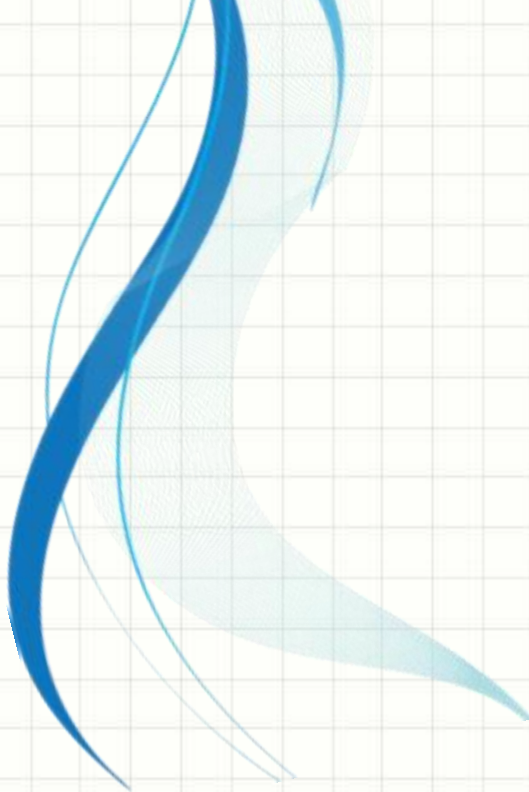
# BIẾN VÀ SỬ DỤNG BIẾN



# CÁC CẤU TRÚC ĐIỀU KHIỂN



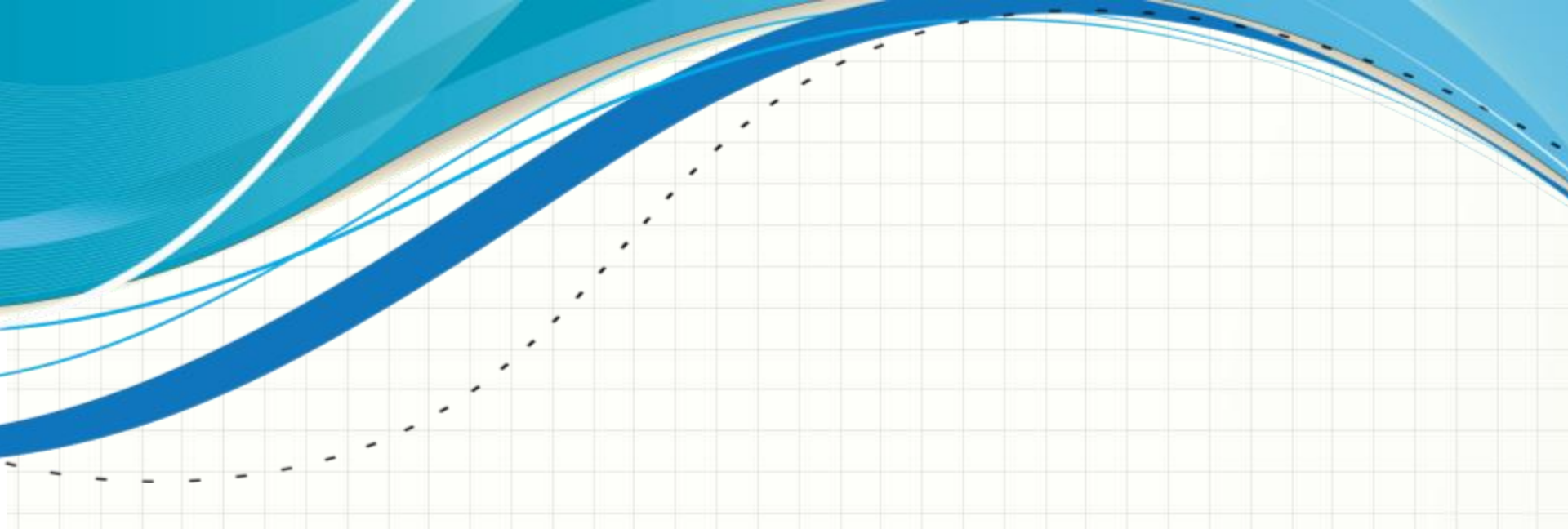
BATCH ( BÓ CÁC  
LỆNH, LÔ CÁC  
LỆNH )



# TRANSACT SQL SCRIPTS.

## BÀI TẬP VỀ NHÀ





# I. BIỂN VÀ SỬ DỤNG BIỂN

# I. BIẾN VÀ SỬ DỤNG BIẾN

## **Khái niệm**

Sao lưu là công việc sao chép thông tin của database vào một thời điểm nhất định vào nơi lưu trữ khác.

## **Vấn đề :**

**Có thật sự có nhu cầu cần sao lưu dữ liệu.**

**Chiến lược sao lưu dữ liệu.**

# I. BIẾN VÀ SỬ DỤNG BIẾN

Biến được dùng để:

- Đếm số lần lặp được thực hiện hoặc dùng để điều khiển vòng lặp
- Dùng lưu giá trị dữ liệu được kiểm tra một số lệnh điều khiển
- Lưu trữ giá trị trả về từ một store Procedure

Các loại biến: biến cục bộ (**local**) và biến toàn cục (**Global**).



# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.1. Biến cục bộ

- Được khai báo trong phần thân của một bó lệnh hoặc một thủ tục.
- Phạm vi hoạt động của biến bắt đầu từ điểm mà nó được khai báo cho đến khi kết thúc một lô (batch) hoặc stored procedure hoặc Function mà nó được khai báo.
- Tên của biến bắt đầu @

# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.1. Biến cục bộ

### ► Khai báo

**DECLARE** @var\_name var\_type

### ► Gán giá trị cho biến

**SET** @var\_name = *expression*

**SELECT** { @var\_name = *expression* } [...n ]

# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.1. Biến cục bộ

Ví dụ 1: tìm hoa đơn theo mahd người dùng nhập vào

```
Declare @mahd INT
```

```
Set @mahd=3
```

```
Select * from Hoadon Where mahd= @mahd
```

# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.1. Biến cục bộ

Ví dụ 2:

-- Khai báo 2 biến.

```
DECLARE @FirstName Nvarchar(20),  
@Region Nvarchar(30)
```

-- Gán giá trị cho 2 biến.

```
SET @FirstName = N'Anne'
```

```
SET @Region= N'WA'
```

-- Dùng chúng trong mệnh đề WHERE của lệnh SELECT.

```
SELECT LastName, FirstName, Title  
FROM Employees  
WHERE FirstName = @FirstName  
      OR Region = @Region
```

# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.1. Biến cục bộ

### Biến cục bộ

Ví dụ 3:


-- Khai báo 1 biến

**DECLARE @EmpID INT**

-- Gán giá trị biến bằng câu lệnh Select

**SELECT @EmpID = MAX(EmployeeID)**

**FROM Employees**



**Gán biểu thức  
cho biến**

# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.1. Biến cục bộ

### Biến cục bộ

Ví dụ 4:

```
Declare @empid int
```

```
Select @empid = employeeid
```

```
From employees
```

```
Order by employeeid desc
```

--Xem kết quả

```
Select @empid
```



# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.1. Biến cục bộ

### Global variable

- Từ SQL Server 7.0 biến Global được định nghĩa như là hàm hệ thống. Mỗi kết nối sẽ tạo ra một session, SQL Server tạo ra sẵn một số biến có sẵn rất tiện ích trong việc lập trình và quản trị hệ thống.
- Các biến này không có kiểu, tên bắt đầu @@.

# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.2. Biến toàn cục

### Danh sách các biến toàn cục

Các biến	Ý nghĩa
<b>@@CONNECTIONS</b>	Số các kết nối đến máy chủ từ lần khởi động cuối.
<b>@@CPU_BUSY</b>	Số milliseconds (một phần nghìn giây) hệ thống đã xử lý từ khi SQL Server được khởi động
<b>@@CURSOR_ROWS</b>	Số bản ghi trong cursor mở gần nhất.
<b>@@DATEFIRST</b>	Giá trị hiện tại của tham số trong lệnh SET DATEFIRST quyết định ngày đầu tiên của tuần.
<b>@@ERROR</b>	Mã lỗi của lỗi xảy ra gần nhất
<b>@@FETCH_STATUS</b>	0 nếu trạng thái lần truy xuất cuối thành công. -1 nếu có lỗi.

# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.2. Biến toàn cục

Các biến	Ý nghĩa
<b>@@IDENTITY</b>	Giá trị identity gần nhất được sinh ra
<b>@@LANGUAGE</b>	Tên của ngôn ngữ đang được sử dụng.
<b>@@MAX_CONNECTIONS</b>	Số kết nối tối đa có thể.
<b>@@ROWCOUNT</b>	Số bản ghi bị tác động bởi câu lệnh SQL gần nhất.
<b>@@SERVERNAME</b>	Tên của máy chủ
<b>@@TIMETICKS</b>	Số milliseconds trong một tick trên máy chủ
<b>@@TRANSCOUNT</b>	Số giao dịch đang hoạt động trên kết nối hiện tại
<b>@@VERSION</b>	Thông tin về phiên bản của SQL Server

# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.3. Biến cursor

### Global variable

- Từ SQL Server 7.0 biến Global được định nghĩa như là hàm hệ thống. Mỗi kết nối sẽ tạo ra một session, SQL Server tạo ra sẵn một số biến có sẵn rất tiện ích trong việc lập trình và quản trị hệ thống.
- Các biến này không có kiểu, tên bắt đầu @@.

# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.3. Biến cursor

### *Khai báo*

```
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
```

# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.3. Biến cursor

*Di chuyển Cursor*

*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*)



# I. BIẾN VÀ SỬ DỤNG BIẾN

## 1.3. Biến cursor

### *Trạng thái Cursor*

@ @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)



## II. CÁC CẤU TRÚC ĐIỀU KHIỂN

## II. Các cấu trúc điều khiển

Control Keyword	Purpose
<u>BEGIN...END</u>	To create a statement block.
<u>GOTO</u> label	To transfer the flow to the specified label.
<u>IF...ELSE</u>	To execute different set of statement/s based on the specified condition.
<u>WHILE</u>	To repeat statement/s while a specified condition holds TRUE.
<u>BREAK</u>	To break the flow of execution and come out of the WHILE loop.
<u>CONTINUE</u>	To restart a WHILE loop.
<u>WAITFOR</u>	To set a delay for statement execution.
<u>RETURN</u>	To exit unconditionally.

## II. Các cấu trúc điều khiển

**BEGIN...END** : Một tập lệnh SQL được thực thi sẽ được đặt trong **BEGIN..END**.

Cú pháp:

**BEGIN**

{

lệnh | đoạn lệnh

}

**END**

## II. Các cấu trúc điều khiển

### Cấu trúc If ... esle

**if** (điều\_kiện)

lệnh .... | khối\_lệnh

**else**

lệnh .... | khối\_lệnh

khối\_lệnh := **begin**

lệnh ... | khối\_lệnh

**end**

## II. Các cấu trúc điều khiển

### Cấu trúc If ... esle

#### Ví dụ:

```
IF ( SELECT COUNT(*) FROM authors
      WHERE contract =0) >0
BEGIN
    PRINT 'These authors do not have contracts on file: '
    SELECT au_lname, au_fname, au_id
    FROM authors
    WHERE contract=0
END
ELSE
BEGIN
    PRINT 'All authors have contracts on file.'
END
```



## II. Các cấu trúc điều khiển

### Cấu trúc If ... else

**Ví dụ:** Cho biết thông tin các thân nhân của nhân viên ở phòng ban 'Nghiên cứu' . Nếu không có xuất ra thông báo trên màn hình ` Nhân viên phòng ban Nghiên cứu không có thân nhân

## II. Các cấu trúc điều khiển

### Cấu trúc If ... else

**Ví dụ:** Kiểm tra xem ứng với mỗi phòng ban lương của trưởng phòng có phải là cao nhất hay không, nếu không tiến hành cập nhật lương trưởng phòng bằng với lương cao nhất của phòng ban đó. Ngược lại thông báo ` kiểm tra dữ liệu hợp lệ `

## II. Các cấu trúc điều khiển

### Cấu trúc While

**while** (*điều\_kiện*)

*lệnh / khối\_lệnh*

*Lệnh ngắt vòng lặp*

- *break*
- *continue*

## II. Các cấu trúc điều khiển

### Cấu trúc While

***Ví dụ 1: 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

## II. Các cấu trúc điều khiển

### Cấu trúc While

- Ví dụ 1: Trong khi đơn giá trung bình vẫn còn nhỏ hơn \$30 thì cập nhật các đơn giá tăng lên gấp đôi đơn giá cũ.

```
WHILE (SELECT AVG(price) FROM titles) < $30
```

```
BEGIN
```

```
    UPDATE titles
```

```
    SET price = price * 2
```

```
    SELECT MAX(price) FROM titles
```

```
    IF (SELECT MAX(price) FROM titles) > $50
```

```
        BREAK
```

```
    ELSE CONTINUE
```

```
END
```

## II. Các cấu trúc điều khiển

### **GOTO:**

Chúng ta có thể thay đổi dòng thực thi của chương trình đến một điểm (còn gọi là nhãn). Các lệnh sau từ khóa **GOTO** sẽ được bỏ qua và tiến trình thực thi tiếp tục ở vị trí nhãn chỉ ra trong mệnh đề **GOTO**.

Cú pháp:

**GOTO** nhãn



## II. Các cấu trúc điều khiển

**RETURN**: Ta có thể dùng **RETURN** bất cứ lúc nào để thoát khỏi một đoạn lệnh hay một thủ tục. Các lệnh sau từ khóa **RETURN** sẽ không được thực thi.

Cú pháp:

**RETURN** [số nguyên]

## II. Các cấu trúc điều khiển

**CASE:** Từ khóa **CASE** cho phép ta trả về một giá trị dựa vào kết quả của một biểu thức. Nó có thể được dùng như một biểu thức.

**Cú pháp:**

```
CASE expression
```

```
WHEN expression1 THEN expression1
```

```
[ [WHEN expression2 THEN expression2] [...]]
```

```
[ELSE expression]
```

```
END
```

## II. Các cấu trúc điều khiển

- **Dạng 1:**

**CASE** **input\_expression**

**WHEN** when\_expression **THEN** result\_expression[ ...n ]

[**ELSE** else\_result\_expression]

**END**

- **Dạng 2:**

**CASE WHEN** *Boolean\_expression* **THEN** result\_expression[ ...n ]

[**ELSE** *else\_result\_expression*]

**END**

## II. Các cấu trúc điều khiển

**Ví dụ:** Hiển thị danh sách authors, nếu state='Or' thì hiển thị Oregon ngược c lại thì hiển thị thuộc tính State

```
SELECT au_fname, au_lname,  
       CASE state  
         WHEN 'OR' THEN 'Oregon' ELSE STATE  
       END AS StateName  
FROM authors
```

## II. Các cấu trúc điều khiển

```
UPDATE publishers
SET state =
CASE
    WHEN country <> "USA"
    THEN "--"
    ELSE state
END
city =
CASE
    WHEN pub_id = "9999"
    THEN "LYON"
    ELSE city
END
WHERE country <> "USA" OR
pub_id = "9999"
```

Hai hay nhiều trường có thể được cập nhật dùng một câu lệnh theo cách này

## II. Các cấu trúc điều khiển

### b. Biểu thức CASE

Ví dụ 4: Ý nghĩa của đoạn lệnh sau?

```
SELECT title, pub_id,  
       CASE WHEN price IS NULL THEN (SELECT  
                                         min(price) FROM titles)  
       ELSE price  
       END  
FROM titles
```



## II. Các cấu trúc điều khiển

### ► Khởi **BEGIN ... END**

Nếu bạn cần nhiều phát biểu được thực thi với nhau thì ta đặt các phát biểu trong cặp Begin ... End

### ► Phát biểu **PRINT**

Phát biểu PRINT: Dùng để in thông tin ra màn hình kết quả của SQL

### ► **PRINT 'any ASCII text' | @local\_variable | @ @FUNCTION | string\_expr**

Ví dụ:

```
PRINT 'Hello!'
```

```
PRINT N'Chào bạn'
```

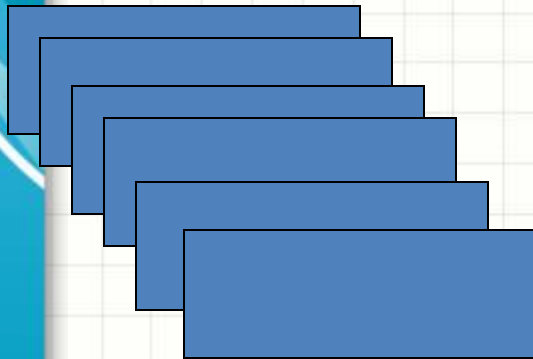
```
PRINT @ @VERSION
```



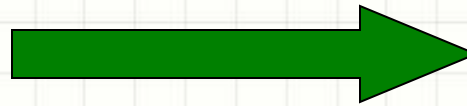
### III. BỐ CÁC LỆNH, LÔ CÁC LỆNH

# Giới thiệu về xử lý theo lô (SQL Batch Processing)

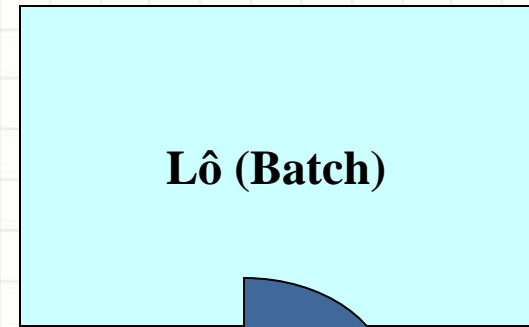
Các lệnh SQL riêng rẽ



Được nhóm lại thành lô (batch)



Lô (Batch)



Được biên dịch thành một kế hoạch thực thi



Kết quả

# Định nghĩa

Quá trình trong đó một tập lệnh được xử lý cùng lúc được gọi là

Xử lý theo lô



# Ví dụ về một lô (batch)

Use Pubs

Select \* from authors

Update authors

set phone= '890 451-7366'

where au\_lname= 'White'

Go

→ **Lệnh báo hiệu kết thúc lô**

# Chú thích trong một lô xử lý

- Các chuỗi ký tự trong mã lệnh chương trình (còn được gọi là chú thích) không được xử lý bởi trình biên dịch.
- Dùng để giải thích cho mã lệnh hay vô hiệu hóa tạm thời các thành phần câu lệnh T-SQL đang xử lý
- Giúp việc bảo trì mã lệnh dễ dàng hơn.
- Chú thích thường được sử dụng để ghi lại tên chương trình, tên tác giả và ngày tháng thực hiện thay đổi mã lệnh.
- Chú thích có thể được dùng để mô tả các phép tính toán phức tạp hay giải thích về phương pháp lập trình.



# Các hình thức chú thích

**SQL Server hỗ trợ hai hình thức chú thích:**

## **1) --(hai gạch ngang)**

**Ví dụ:**

**USE Northwind**

**GO**

**-- Đây là chú thích.**

## **2) /\* ... \*/ (cặp dấu gạch chéo và dấu sao)**

**Ví dụ:**

**SELECT \* FROM Employees /\*Đây là chú thích\*/**

# Chú thích nhiều dòng

- Chú thích nhiều dòng `/* */` không thể vượt quá một lô. Một chú thích hoàn chỉnh phải nằm trong một lô xử lý.
- Ví dụ, trong công cụ Query Analyzer, lệnh GO báo hiệu kết thúc lô. Khi gặp lệnh GO trên dòng lệnh nó sẽ gửi tất cả các mã lệnh sau từ khóa GO cuối cùng lên máy chủ SQL trong một lô xử lý.
- Nếu lệnh GO xuất hiện trên một dòng giữa `/*` và `*/` thì Query Analyzer sẽ gửi đi một đoạn chú thích có các ký tự đánh dấu sai trong mỗi lô và sẽ gây ra lỗi cú pháp.

# Chú thích nhiều dòng

- Nếu một lệnh trong batch bị lỗi, SQL Server sẽ không thi hành tất cả các câu lệnh trong gói.
- Không thể đặt tất cả các câu lệnh sau trong một batch : CREATE PROCEDURE, CREATE TRIGGER, CREATE VIEW, CREATE RULE, CREATE DEFAULT.
- Mỗi lệnh trên yêu cầu đặt trong một batch.



## IV. TRANSACT SCRIPTS - BTVN

## IV. Transact scripts - BTVN

- Một script là tập các câu lệnh T-SQL được lưu trữ trong một file, gồm 1 hoặc nhiều batch
- *Script được sử dụng nhằm:*
  - Lưu giữ bản sao các bước tạo database trên server.
  - Di chuyển các câu lệnh từ máy này sang máy khác.

# IV. Transact scripts - BTVN

- Bài tập về nhà