

TRƯỜNG ĐÀO TẠO LẬP TRÌNH VIÊN VÀ QUẢN TRỊ MẠNG QUỐC TẾ BACHKHOA-APTECH

---

# Bài 13

# Lập trình với Transact-SQL

# Tóm tắt

- Tổng quan lập trình Transact-SQL
- Thành phần của Transact-SQL
- Lệnh điều khiển luồng
- Hàm Transact-SQL



# Tổng quan

**Lập trình Transact-SQL** là:

- ✓ Một **ngôn ngữ thủ tục mở rộng** cho SQL.
- ✓ Được mở rộng bằng cách thêm vào các **chương trình con** (subroutines) và **lập trình cấu trúc** tương tự như các ngôn ngữ bậc cao.
- Lập trình Transact-SQL cũng có các **nguyên tắc** (rules) và **cú pháp điều khiển** và cho phép các câu lệnh lập trình làm việc cùng nhau.
- Người dùng có thể **điều khiển luồng** chương trình bằng các câu lệnh điều kiện như **IF** và vòng lặp như **WHILE**.

# Thành phần

- Các thành phần ngôn ngữ lập trình Transact-SQL cho phép cho phép thực hiện **nhều thao tác** không thể thực hiện bằng một câu **lệnh đơn lẻ**.
- Người dùng có thể nhóm nhiều câu lệnh Transact-SQL với nhau bằng cách dùng một trong các cách sau:
  1. Các lô (**Batches**)
  2. Thủ tục lưu (**Stored Procedures**)
  3. **Triggers**
  4. Kịch bản (**Scripts**)

# Thành phần

## Các lô (Batches)

- Là một tập hợp gồm một hay nhiều câu lệnh Transact-SQL được gửi đi như một khối từ ứng dụng tới server.

## Thủ tục lưu (Stored Procedures)

- Là một tập hợp các Transact-SQL đã được định nghĩa và biên dịch từ trước trên server.

## Triggers

- Là một dạng thủ tục lưu đặc biệt được thực thi khi người dùng thực hiện một sự kiện như các thao tác INSERT, DELETE, hoặc UPDATE trên bảng.

## Kịch bản (Scripts)

- Là một dãy các câu lệnh Transact-SQL được lưu trữ trong một file được sử dụng là đầu vào cho trình soạn thảo code SSMS và tiện ích sqlcmd.

# Thành phần

Các tính năng sau đây cho phép người dùng làm việc với các câu lệnh Transact-SQL:

## Biến (Variables)

- Cho phép người dùng lưu trữ dữ liệu có thể được dùng làm đầu vào cho câu lệnh Transact-SQL.

## Điều khiển luồng (Control-of-flow)

- Được sử dụng cho các cấu trúc điều kiện trong Transact-SQL.

## Quản lý lỗi (Error Handling)

- Là kỹ thuật được sử dụng cho quản lý lỗi và cung cấp thông tin cho người dùng về lỗi đã xảy ra.

# Thành phần

## Lô(batch):

- Là một nhóm của một hoặc **nhều câu lệnh** Transact-SQL được gửi tới server như là một khối(unit) từ ứng dụng để thực thi.
- SQL Server biên dịch các câu lệnh SQL trong lô (batch) thành **một khối** có thể thực thi duy nhất, hay còn được gọi là một execution plan.
- Trong execution plan, các câu lệnh SQL được thực hiện từng lệnh một.

# Thành phần

- Câu lệnh lô Transact-SQL nên được kết thúc bằng **dấu chấm phẩy** (semicolon).
- **Lỗi biên dịch** như lỗi cú pháp làm hạn chế việc biên dịch execution plan.
- Các **câu lệnh SQL** thực hiện **trước khi có lỗi** thực thi xảy ra sẽ **không bị ảnh hưởng**.



# Thành phần

**Lỗi thực thi** (run-time error) như vi phạm ràng buộc (constraint) hoặc tràn số sẽ gây một trong các ảnh hưởng sau:

- Phần lớn các lỗi thực thi (run-time errors) sẽ làm **dừng câu lệnh hiện tại** và những câu lệnh tiếp theo trong lô (batch).
- Lỗi thực thi cụ thể như **vi phạm ràng buộc** không chỉ **dừng thực hiện các câu lệnh** hiện có mà còn cả các câu lệnh còn lại trong lô được thực hiện.

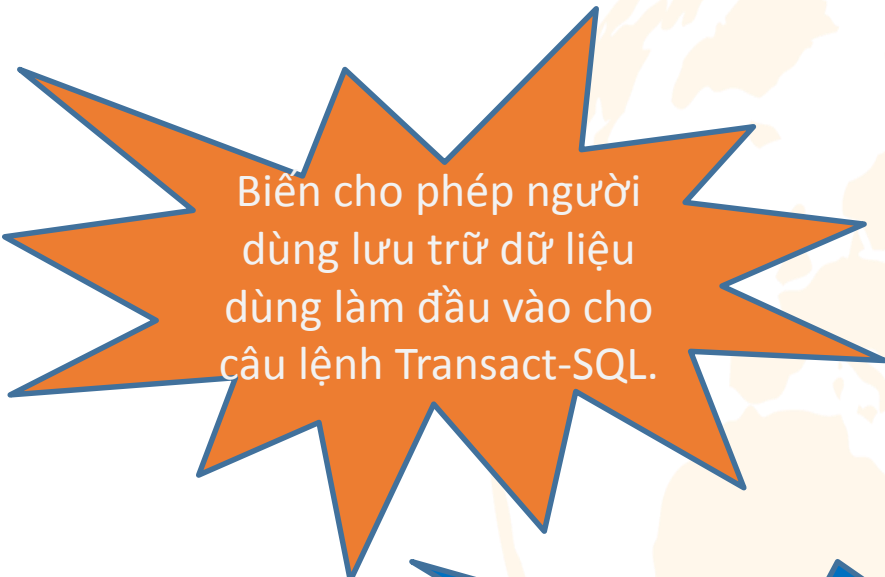
# Thành phần

Ví dụ:

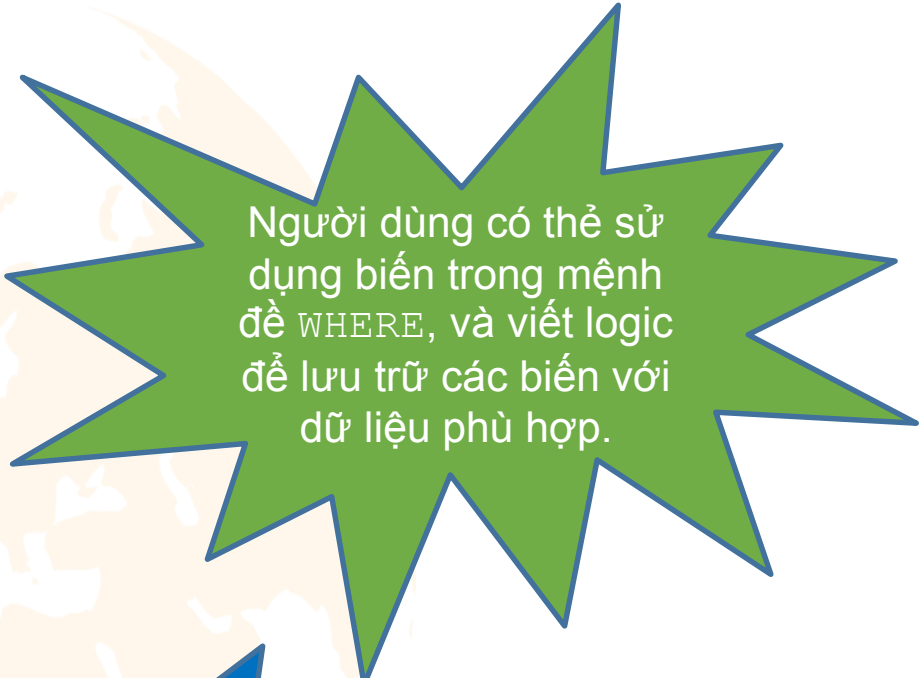
```
USE BKInside;
GO
CREATE TABLE LopHoc
(Id int IDENTITY(100, 5),
TenLop nvarchar(100)
)
GO
INSERT LopHoc (TenLop) VALUES (N'Quản trị mạng')
INSERT LopHoc (TenLop) VALUES (N'Lập trình Web PHP')
INSERT LopHoc (TenLop) VALUES (N'Lập trình di động Android')
INSERT LopHoc (TenLop) VALUES (N'Hệ thống')
INSERT LopHoc (TenLop) VALUES (N'Kiểm thử phần mềm')
GO
SELECT Id, TenLop FROM LopHoc ORDER BY TenLop ASC;
GO
```

# Thành phần

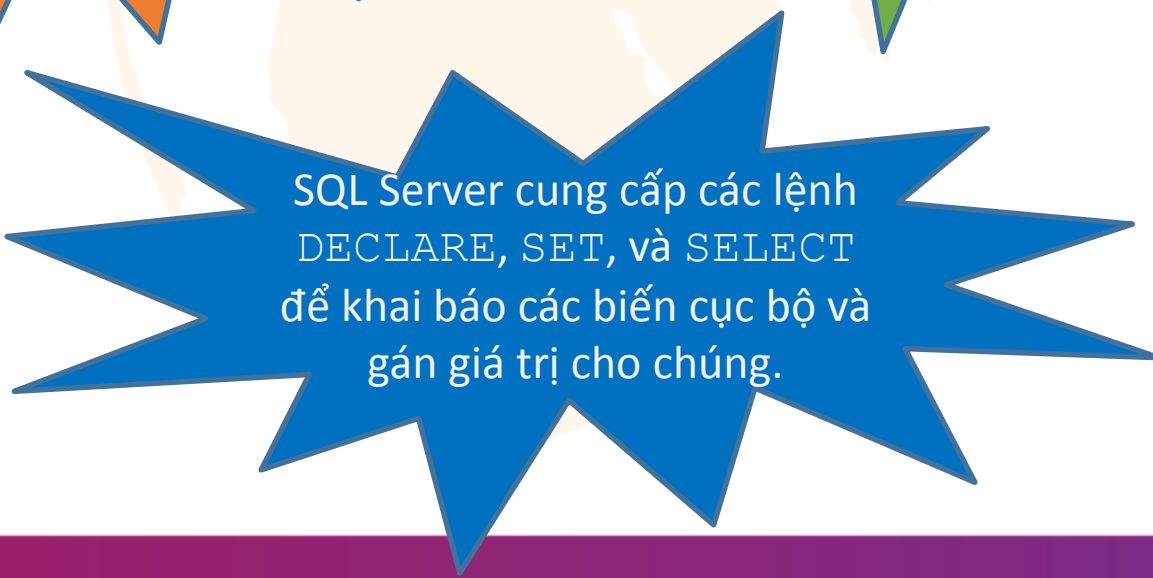
## Biến (variable):



Biến cho phép người dùng lưu trữ dữ liệu dùng làm đầu vào cho câu lệnh Transact-SQL.



Người dùng có thể sử dụng biến trong mệnh đề `WHERE`, và viết logic để lưu trữ các biến với dữ liệu phù hợp.



SQL Server cung cấp các lệnh `DECLARE`, `SET`, và `SELECT` để khai báo các biến cục bộ và gán giá trị cho chúng.

# Thành phần

## DECLARE:

- Từ khóa được sử dụng **để khai báo và khởi tạo giá trị** ban đầu cho biến là NULL nếu không cung cấp giá trị cho nó.
- Các biến **được gán giá trị** bằng câu lệnh **SELECT** hoặc **SET**.

### Cú pháp:

```
DECLARE { { @local_variable [AS] data_type } | [ = value ] }
```

# Thành phần

Trong đó:

- **@local\_variable**: chỉ ra tên của biến cục bộ được bắt đầu bằng kí hiệu @.
- **data\_type**: chỉ ra kiểu dữ liệu cho biến. Không sử dụng kiểu dữ liệu image, text, hoặc ntext.
- **= value**: gán giá trị value cho biến . Value có thể là một biểu thức hoặc hằng giá trị. Giá trị lên khớp với kiểu khai báo cho biến hoặc nên chuyển kiểu tường minh.

# Thành phần

Ví dụ:

```
DECLARE
```

```
@tong_so_nv int,
```

```
@tong_so_pb int
```

Gán giá trị cho biến bằng SELECT hoặc SET



```
SELECT @tong_so_nv = COUNT(*) FROM NhanVien
```

```
SET @tong_so_pb = (SELECT COUNT(*) FROM PhongBan)
```

```
PRINT N'Tổng số nhân viên BKShop là: ' +
```

```
CONVERT(varchar, @tong_so_nv)
```

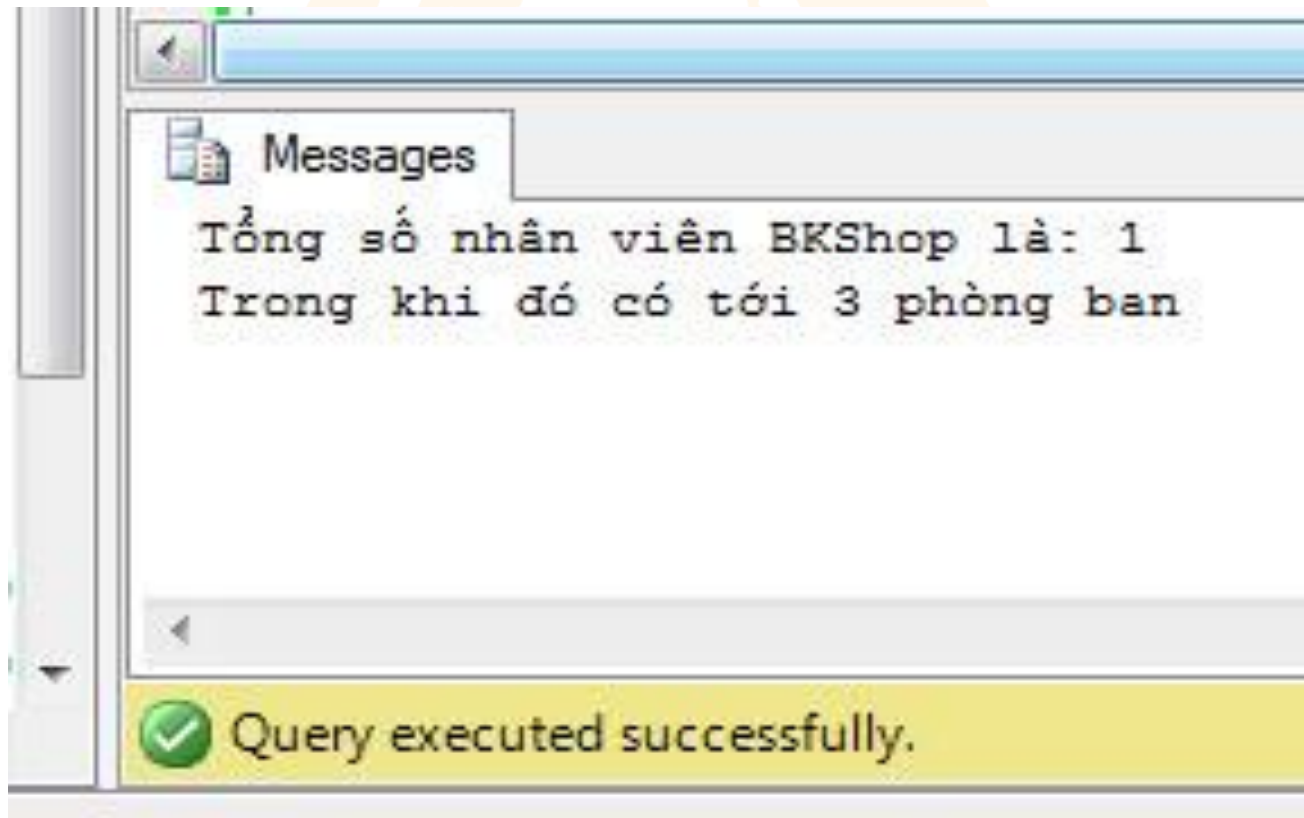
```
PRINT N'Trong khi đó có tới ' +
```

```
CONVERT(varchar, @tong_so_pb) + N' phòng ban'
```

```
GO
```

# Thành phần

Kết quả:



# Thành phần

Chỉ gán được giá trị cho một biến tại một thời điểm

Khi sử dụng truy vấn con để gán giá trị, nó sẽ phát sinh lỗi và không làm việc nếu truy vấn này trả về nhiều giá trị hay nhiều dòng.

Có thể gán các giá trị cho nhiều biến trong cùng một câu lệnh

Gán một trong các giá trị trả về cho biến, thậm chí người dùng cũng không biết rằng đã có nhiều giá trị trả về.

SET

SELECT

Để gán giá trị cho các biến, khuyên nên (recommended) sử dụng `SET @local_variable` thay cho `SELECT @local_variable`



# Điều khiển

Bảng sau đây liệt kê một số từ khóa của ngôn ngữ điều khiển luồng Transact-SQL:

Control-Of-Flow Language Keywords
RETURN
THROW
TRY....CATCH
WAITFOR
WHILE
BEGIN....END
BREAK
CONTINUE
GOTO label
IF...ELSE

# Điều khiển

Câu lệnh **BEGIN...END** bao quanh một dãy các câu lệnh Transact-SQL với mục đích tạo một nhóm các lệnh Transact-SQL để thực thi:

## Cú pháp:

```
BEGIN
{
  sql_statement | statement_block
}
END
```

Trong đó:

- **{sql\_statement| statement\_block}**: là một câu lệnh Transact-SQL hợp lệ bất kỳ được định nghĩa bằng một khối câu lệnh.

# Điều khiển

- Câu lệnh **IF...ELSE** áp đặt(enforces) một điều kiện về việc thực thi một câu lệnh Transact-SQL.
- Câu lệnh Transact-SQL theo sau từ khóa IF và **điều kiện** chỉ được **thực thi nếu** điều kiện được thỏa mãn và **trả về TRUE**.
- Từ khóa **ELSE** là câu lệnh Transact-SQL tùy chọn, chỉ được thực thi khi điều kiện của IF **không thỏa mãn** và **trả về FALSE**.

# Điều khiển

## Cú pháp:

```
IF Boolean_expression  
{ sql_statement | statement_block }  
[ ELSE  
{ sql_statement | statement_block } ]
```

Trong đó:

- **Boolean\_expression**: chỉ ra một biểu thức trả về giá trị TRUE hoặc FALSE
- **{sql\_statement | statement\_block}**: là câu lệnh Transact-SQL hợp lệ bất kỳ được định bằng việc sử dụng một khối lệnh.

# Điều khiển

- Câu lệnh **WHILE** – chỉ ra một điều kiện cho việc thực hiện lặp đi lặp lại một khối lệnh.
- Các câu lệnh được **thực thi lặp đi lặp lại** cho tới điều kiện chỉ ra là đúng.
- Việc thực thi các câu lệnh trong vòng lặp WHILE có thể được điều khiển bằng việc dùng các từ khóa **BREAK** và **CONTINUE**.

# Điều khiển

## Cú pháp:

```
WHILE Boolean_expression  
{ sql_statement | statement_block | BREAK | CONTINUE }
```

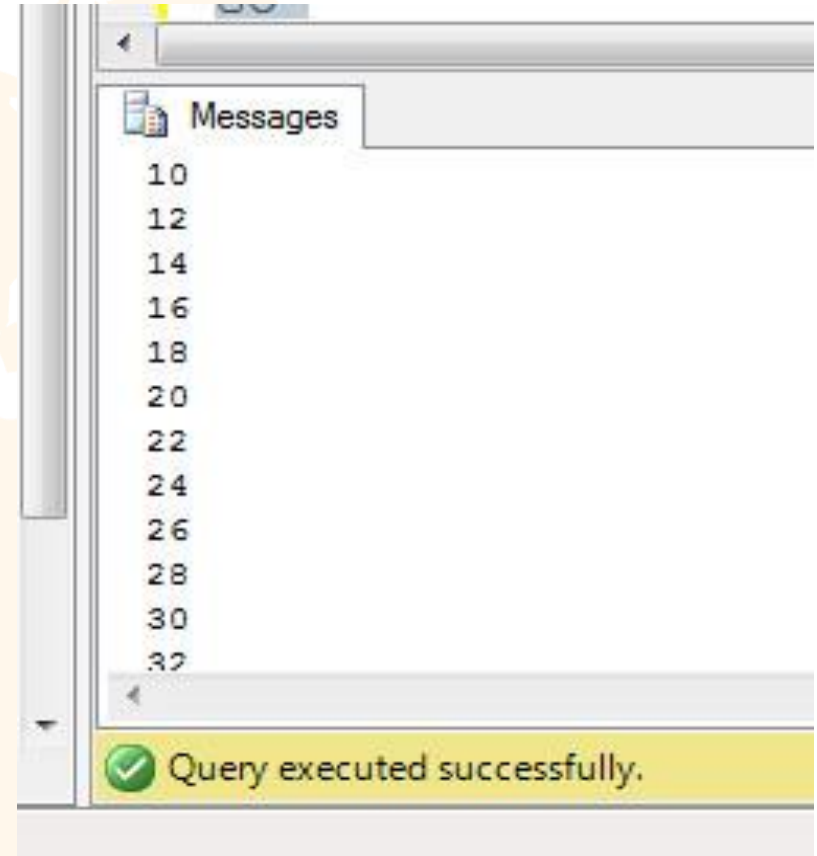
Trong đó:

- **Boolean\_expression**: chỉ ra biểu thức trả về giá trị TRUE hoặc FALSE.
- **{sql\_statement| statement\_block}**: là câu lệnh Transact-SQL hợp lệ bất kỳ được định bằng việc dùng một khối lệnh.
- **BREAK**: thoát khỏi vòng lặp.
- **CONTINUE**: chuyển sang lần lặp kế tiếp.

# Điều khiển

Ví dụ: in ra tất cả số chẵn từ 10 tới 95

```
DECLARE @flag int
SET @flag = 10
WHILE (@flag <=95)
BEGIN
    IF @flag%2 =0
    PRINT @flag
    SET @flag = @flag + 1
    CONTINUE;
END
GO
```



# Hàm

Bảng dưới đây liệt kê một số hàm xác định, không xác định có sẵn:

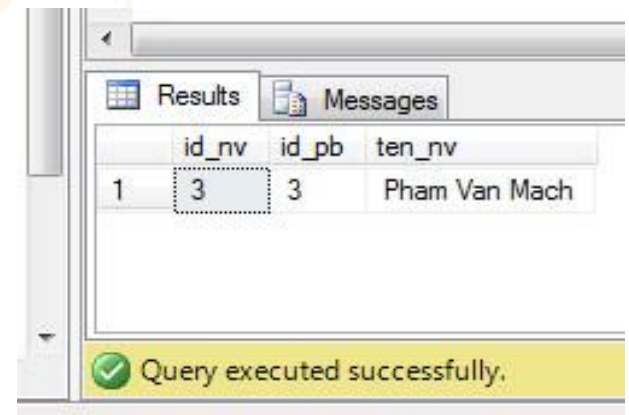
Deterministic Built-in Functions	Non-Deterministic Built-in Functions
POWER	@@TOTAL_WRITE
ROUND	CURRENT_TIMESTAMP
RADIANS	GETDATE
EXP	GETUTCDATE
FLOOR	GET_TRANSMISSION_STATUS
SQUARE	NEWID
SQRT	NEWSEQUENTIALID
LOG	@@CONNECTIONS
YEAR	@@CPU_BUSY
ABS	@@DBTS
ASIN	@@IDLE
ACOS	@@IOBUSY
SIGN	@@PACK_RECEIVED



# Hàm

Ví dụ: Tạo hàm truy vấn lấy tất cả nhân viên phòng kỹ thuật trong csdl BKShop

```
CREATE FUNCTION getNhanVienKyThuat() RETURNS TABLE AS  
RETURN(  
SELECT * FROM NhanVien WHERE id_pb = 3  
)  
GO  
  
SELECT * FROM getNhanVienKyThuat()  
GO
```



# Tóm tắt bài học

- Transact-SQL cung cấp **các thành phần lập trình cơ bản** như biến, các thành phần điều khiển luồng, cấu trúc lặp và điều kiện.
- Một **lô(batch)** là một tập hợp của một hay nhiều câu lệnh Transact-SQL được gửi như là **khối(unit)** từ ứng dụng tới server.
- **Biến** cho phép người dùng **lưu trữ dữ liệu** và được dùng làm đầu vào của các câu lệnh Transact-SQL khác.

## TRƯỜNG ĐÀO TẠO LẬP TRÌNH VIÊN VÀ QUẢN TRỊ MẠNG QUỐC TẾ BACHKHOA-APTECH

---

# Thank for watching!

