

# **FPT POLYTECHNIC**



## **Bài 3:** **LÀM VIỆC VỚI KIỂU DỮ LIỆU VÀ MÃ KỊCH BẢN**

---

- Các nội dung đã học trong bài trước
  - Kiểu dữ liệu
  - Tạo CSDL quan hệ
  - Làm quen với T-SQL

1. Làm việc với các kiểu dữ liệu

2. Mã kịch bản

# LÀM VIỆC VỚI CÁC KIỂU DỮ LIỆU

- Khi làm việc với các biểu thức chứa nhiều kiểu dữ liệu khác nhau, phải thực hiện chuyển đổi giữa các kiểu dữ liệu.
- Hai loại chuyển đổi dữ liệu
  - Chuyển đổi ngầm (do SQL Server tự thực hiện)
  - Chuyển đổi tường minh (sử dụng các hàm thư viện)

- Các trường hợp xảy ra chuyển đổi ngầm
  - Gán giá trị cho một cột có kiểu dữ liệu khác với giá trị được gán.
  - Biểu thức tính toán có sự tham gia của nhiều loại dữ liệu khác nhau. Trong trường hợp này:
    - SQL Server chuyển đổi giá trị kiểu dữ liệu có độ ưu tiên thấp hơn sang kiểu dữ liệu có độ ưu tiên cao hơn.
- Lập trình viên cần hiểu về chuyển đổi ngầm dữ liệu để tránh thu được kết quả không mong muốn khi viết câu lệnh SQL.



### ■ Biểu thức tính toán có sự tham gia của nhiều kiểu dữ liệu khác nhau

InvoiceTotal \* .0775

-- InvoiceTotal (kiểu money) bị chuyển đổi ngầm sang kiểu decimal

PaymentTotal – 100

-- Giá trị số 100 bị chuyển đổi ngầm sang kiểu money

### ■ Gán giá trị có kiểu dữ liệu khác với kiểu dữ liệu của cột

PaymentDate = '2008-08-05'

-- Giá trị ngày có định dạng chữ '2008-08-05' bị chuyển đổi ngầm sang giá trị smalldatetime

## Thứ tự ưu tiên của các kiểu dữ liệu phổ biến trong SQL Server

Độ ưu tiên	Nhóm	Kiểu dữ liệu
Cao nhất	Ngày/Giờ	datetime smalldatetime
	Số	float real decimal money smallmoney int smallint tinyint bit
Thấp nhất	Chuỗi	nvarchar nchar varchar char



- Không phải tất cả kiểu dữ liệu đều có thể thực hiện chuyển đổi ngầm sang toàn bộ kiểu dữ liệu khác.
  - Sinh viên tham khảo phần **Chuyển đổi không thể thực hiện ngầm** trong sách giáo khoa

- Khi thực hiện các chuyển đổi:
    - Không thể hoàn thành với chuyển đổi ngầm định
    - Chuyển dữ liệu có kiểu dữ liệu với độ ưu tiên cao hơn về kiểu dữ liệu có độ ưu tiên thấp hơn
- ta cần sử dụng hàm **CAST** hoặc **CONVERT** để thực hiện *phép chuyển đổi tương minh*.

### ■ Cú pháp hàm CAST

**CAST(<Biểu thức> AS <Kiểu dữ liệu>[(length)])**

### ■ Trường hợp sử dụng:

- Hàm **CAST** dùng để chuyển đổi tường minh, hay *ép kiểu*, *một biểu thức* từ kiểu dữ liệu này sang kiểu dữ liệu khác.

### ■ Câu lệnh SELECT sử dụng hàm CAST

```
SELECT InvoiceDate, InvoiceTotal,
       CAST(InvoiceDate AS varchar) AS varcharDate,
       CAST(InvoiceTotal AS int) AS integerTotal,
       CAST(InvoiceTotal AS varchar) AS varcharTotal
FROM Invoices
```

	InvoiceDate	InvoiceTotal	varcharDate	integerTotal	varcharTotal
1	2008-04-08 00:00:00	3813.33	Apr 8 2008 12:00AM	3813	3813.33
2	2008-04-10 00:00:00	40.20	Apr 10 2008 12:00AM	40	40.20
3	2008-04-13 00:00:00	138.75	Apr 13 2008 12:00AM	139	138.75
4	2008-04-16 00:00:00	144.70	Apr 16 2008 12:00AM	145	144.70

### ■ Chuyển đổi dữ liệu khi thực hiện phép chia số nguyên

Thao tác	Kết quả
50/100	0 (Kiểu int)
50/CAST(100 AS decimal(3))	.500000 (Kiểu decimal)

### ■ Cú pháp hàm **CONVERT**

**CONVERT(<Kiểu dữ liệu>[(length)], <Biểu thức>[, <Tham số định dạng>])**

### ■ Trường hợp sử dụng

- Sử dụng hàm **CONVERT** để chuyển đổi tương minh một biểu thức từ kiểu dữ liệu này sang kiểu dữ liệu khác
  - Khi chuyển đổi sang kiểu dữ liệu ký tự và cần định dạng hiển thị cho dữ liệu
- **Tham số định dạng:** định dạng hiển thị cho các giá trị ngày/giờ, số thực, tiền tệ khi chuyển đổi sang kiểu ký tự.
- Chỉ có SQL Server hỗ trợ hàm **CONVERT**. Hàm này ít được sử dụng hơn hàm **CAST** (Hàm chuẩn ANSI).

### ■ Câu lệnh SELECT sử dụng hàm CONVERT

```

SELECT  CONVERT(varchar, InvoiceDate)           AS varcharDate,
        CONVERT(varchar, InvoiceDate, 1)       AS varcharDate_1,
        CONVERT(varchar, InvoiceDate, 107)     AS varcharDate_107,
        CONVERT(varchar, InvoiceTotal)         AS varcharTotal,
        CONVERT(varchar, InvoiceTotal, 1)      AS varcharTotal_1
FROM    Invoices

```

	varcharDate	varcharDate_1	varcharDate_107	varcharTotal	varcharTotal_1
1	Apr 8 2008 12:00AM	04/08/08	Apr 08, 2008	3813.33	3,813.33
2	Apr 10 2008 12:00AM	04/10/08	Apr 10, 2008	40.20	40.20
3	Apr 13 2008 12:00AM	04/13/08	Apr 13, 2008	138.75	138.75
4	Apr 16 2008 12:00AM	04/16/08	Apr 16, 2008	144.70	144.70

- Sinh viên tham khảo các mã tham số định dạng cho câu lệnh CONVERT trong Book Online hoặc trên Website sau <http://msdn.microsoft.com/en-us/library/ms187928.aspx>

# Làm việc với kiểu dữ liệu chuỗi

Hàm	Mô tả
<code>LEN (string)</code>	Trả về số lượng ký tự trong chuỗi, tính cả ký tự trắng đầu chuỗi nhưng không bao gồm ký tự trắng cuối chuỗi
<code>LTRIM (string)</code>	Trả về chuỗi với các ký tự trắng đầu chuỗi bị loại bỏ
<code>RTRIM (string)</code>	Trả về chuỗi với các ký tự trắng cuối chuỗi bị loại bỏ
<code>LEFT (string, length)</code>	Trả về chuỗi con có chiều dài length tính từ đầu chuỗi
<code>RIGHT (string, length)</code>	Trả về chuỗi con có chiều dài length tính từ cuối chuỗi
<code>SUBSTRING ( (string, start, length)</code>	Trả về chuỗi con có chiều dài length tính từ vị trí start
<code>CHARINDEX (find, search[, start])</code>	Trả về vị trí xuất hiện đầu tiên của chuỗi find trong chuỗi search bắt đầu từ vị trí start. Trả về 0 nếu không tìm thấy
<code>PATINDEX (find, search)</code>	Tương tự hàm CHARINDEX nhưng chuỗi mẫu find có thể bao gồm các ký tự thay thế
<code>REPLACE (search, find, replace)</code>	Trả về chuỗi search với tất cả các chuỗi find được thay thế bởi chuỗi replace
<code>REVERSE (string)</code>	Trả về chuỗi với các ký tự đảo ngược
<code>LOWER (string)</code>	Trả về chuỗi được chuyển đổi thành các chữ cái thường
<code>UPPER (string)</code>	Trả về chuỗi được chuyển đổi thành các chữ cái hoa
<code>SPACE (integer)</code>	Trả về chuỗi với số lượng ký tự trắng được chỉ định qua tham số integer



Hàm	Kết quả
<code>LEN('SQL Server')</code>	10
<code>LEN(' SQL Server ')</code>	12
<code>LEFT('SQL Server', 3)</code>	'SQL'
<code>LTRIM(' SQL Server ')</code>	'SQL Server '
<code>RTRIM(' SQL Server ')</code>	' SQL Server'
<code>LTRIM(RTRIM(' SQL Server '))</code>	'SQL Server'
<code>LOWER('SQL Server')</code>	'sql server'
<code>UPPER('ca')</code>	CA
<code>PATINDEX('%v_r%', 'SQL Server')</code>	8
<code>CHARINDEX('SQL', ' SQL Server')</code>	3
<code>CHARINDEX('-', '(559) 555-1212')</code>	10
<code>SUBSTRING('(559) 555-1212', 7, 8)</code>	555-1212
<code>REPLACE(RIGHT('(559) 555-1212', 13), ') ', '-')</code>	559-555-1212

### ■ Câu lệnh SELECT sử dụng các hàm xử lý chuỗi LEFT, RIGHT và SUBSTRING

```
SELECT VendorName, VendorContactLName + ', ' +
    LEFT(VendorContactFName, 1) + '!' AS ContactName,
    RIGHT(VendorPhone, 8) AS Phone
FROM Vendors
WHERE SUBSTRING(VendorPhone, 2, 3) = 559
ORDER BY VendorName
```

	VendorName	ContactName	Phone
1	Abbey Office Furnishings	Francis, K.	555-8300
2	BFI Industries	Kaleigh, E.	555-1551
3	Bill Marvin Electric Inc	Hostlery, K.	555-5106
4	Cal State Termite	Hunter, D.	555-1534
5	California Business Machines	Rohansen, A.	555-5570

### ■ 1. Sắp thứ tự

- Khi sắp xếp một cột kiểu chuỗi chứa số, bạn có thể nhận được kết quả không mong đợi.
- Để tránh điều đó, bạn chuyển đổi cột kiểu chuỗi thành giá trị số trong mệnh đề ORDER BY.

### ■ Hướng dẫn sử dụng hàm CAST để sắp xếp cột kiểu chuỗi chứa các số

- **Bảng StringSample được sắp xếp**

```
SELECT * FROM StringSample
ORDER BY ID
```

	ID	Name	AltID
1	1	Lizbeth Darien	01
2	17	Lance Pinos-Potter	17
3	2	Damell O'Sullivan	02
4	20	Jean Paul Renard	20
5	3	Alisha von Strump	03

- **Bảng StringSample được sắp xếp bởi cột ID được ép kiểu thành số nguyên**

```
SELECT * FROM StringSample
ORDER BY CAST(ID AS int)
```

	ID	Name	AltID
1	1	Lizbeth Darien	01
2	2	Damell O'Sullivan	02
3	3	Alisha von Strump	03
4	17	Lance Pinos-Potter	17
5	20	Jean Paul Renard	20



## ■ 2. Phân tách ký tự

- Nếu chuỗi gồm hai hay nhiều thành phần, bạn có thể phân tách chuỗi thành những thành phần độc lập.
- Sử dụng hàm CHARINDEX để định vị những ký tự phân tách. Sau đó, dùng hàm LEFT, RIGHT, SUBSTRING và LEN để trích ra những thành phần độc lập.

## ■ Hướng dẫn sử dụng hàm chuỗi ký tự để tách chuỗi

```
SELECT Name,  
       LEFT(Name, CHARINDEX(' ', Name) - 1)           AS First,  
       RIGHT(Name, LEN(Name) - CHARINDEX(' ', Name) ) AS Last  
FROM StringSample
```

	Name	First	Last
1	Lizabeth Darien	Lizabeth	Darien
2	Damell O'Sullivan	Damell	O'Sullivan
3	Lance Pinos-Potter	Lance	Pinos-Potter
4	Jean Paul Renard	Jean	Paul Renard
5	Alisha von Strump	Alisha	von Strump

# Các hàm xử lý dữ liệu ngày/giờ

Hàm	Mô tả
GETDATE ( )	Trả về giá trị datetime cho ngày và giờ hiện tại.
GETUTCDATE ( )	Trả về giá trị datetime cho ngày và giờ hiện tại theo UTC dựa trên đồng hồ hệ thống và múi giờ được cài đặt.
SYSDATETIME ( )	Trả về giá trị datetime2(7) cho ngày và giờ hiện tại.
SYSUTCDATETIME ( )	Trả về giá trị datetime2(7) cho ngày và giờ hiện tại theo UTC dựa trên đồng hồ hệ thống và múi giờ được cài đặt.
SYSDATETIMEOFFSET ( )	Trả về giá trị datetimeoffset(7) cho ngày và giờ hiện tại theo UTC dựa trên đồng hồ hệ thống và múi giờ được cài đặt, với múi giờ <i>không</i> được điều chỉnh theo ngày thay đổi múi giờ.
DAY (date)	Trả về ngày trong tháng theo kiểu số nguyên.
MONTH (date)	Trả về tháng theo kiểu số nguyên.
YEAR (date)	Trả về năm với 4 chữ số, theo kiểu số nguyên.
DATETIME (datepart, date)	Trả về chuỗi tương ứng với phần datepart (year, month...) của date.
DATEPART (datepart, date)	Trả về số nguyên tương ứng với phần datepart (year, month...) của date
DATEADD (datepart, number, date)	Trả về ngày là kết quả phép cộng số lượng đơn vị datepart được chỉ định vào date.
DATEDIFF (datepart, startdate, enddate)	Trả về số đơn vị datepart giữa ngày bắt đầu và kết thúc được chỉ định.
ISDATE (expression)	Trả về giá trị 1 (true) nếu biểu thức là giá trị ngày/giờ; ngược lại, trả về giá trị 0 (false).

# Các hàm xử lý dữ liệu ngày/giờ

- Các hàm **DATETIME**, **DATEPART**, **DATEADD**, **DATEDIFF** phải được truyền vào tham số **datepart**
- Các giá trị tham số **datepart** và dạng viết tắt tương ứng

Tham số	Dạng viết tắt
year	yy, yyyy
quarter	qq, q
month	mm, m
dayofyear	dy, y
day	dd, d
week	wk, ww
weekday	Dw
hour	Hh
minute	mi, n
second	ss, s
millisecond	Ms
microsecond	Mcs
Tzoffset	tz

Hàm	Kết quả
GETDATE ()	2008-08-06 14:10:13.813
GETUTCDATE ()	2008-08-06 21:10:13.813
SYSDATETIME ()	2008-08-06 14:10:13.8160822
SYSUTCDATETIME ()	2008-08-06 21:10:13.8160822
SYSDATETIMEOFFSET ()	2008-08-06 14:10:13.8160822-07.00
MONTH ('2008-09-30')	9
DAY ('2008-09-30')	30
YEAR ('2008-09-30')	2008
ISDATE ('2008-09-30')	1
ISDATE ('2008-09-31')	0
ISDATE ('23:59:59')	1
ISDATE ('23:99:99')	0

# sử dụng hàm DATEPART

Hàm	Kết quả
<code>DATEPART(day, '2008-09-30 11:35:00')</code>	30
<code>DATEPART(month, '2008-09-30 11:35:00')</code>	9
<code>DATEPART(year, '2008-09-30 11:35:00')</code>	2008
<code>DATEPART(hour, '2008-09-30 11:35:00')</code>	11
<code>DATEPART(minute, '2008-09-30 11:35:00')</code>	35
<code>DATEPART(second, '2008-09-30 11:35:00')</code>	0
<code>DATEPART(quarter, '2008-09-30 11:35:00')</code>	3
<code>DATEPART(dayofyear, '2008-09-30 11:35:00')</code>	273
<code>DATEPART(week, '2008-09-30 11:35:00')</code>	40
<code>DATEPART(weekday, '2008-09-30 11:35:00')</code>	7
<code>DATEPART(milliseconds, '11:35:00.1234567')</code>	123
<code>DATEPART(microsecond, '11:35:00.1234567')</code>	123456
<code>DATEPART(nanosecond, '11:35:00.1234567')</code>	123456700
<code>DATEPART(tzoffset, '11:35:00.1234567 -07:00')</code>	-420



Hàm	Kết quả
DATENAME (day, '2008-09-30 11:35:00')	30
DATENAME (month, '2008-09-30 11:35:00')	September
DATENAME (quarter, '2008-09-30 11:35:00')	3
DATENAME (weekday, '2008-09-30 11:35:00')	Saturday
DATEADD (day, 1, '2008-09-30 11:35:00')	2008-10-01 11:35:00.000
DATEADD (week, 1, '2008-09-30 11:35:00')	2008-10-07 11:35:00.000
DATEDIFF (day, '2007-12-01', '2008-09-30')	303
DATEDIFF (month, '2007-12-01', '2008-09-30')	9
DATEDIFF (minute, '06:46:45', '11:35:00')	289
DATEDIFF (second, '06:46:45', '11:35:00')	17295
DATEDIFF (quarter, '2007-12-01', '2008-09-30')	3
DATEDIFF (week, '2007-12-01', '2008-09-30')	44
DATEDIFF (day, '2008-09-30', '2007-12-01')	-303



- Giá trị thời gian thường chứa cả thành phần ngày và giờ. Do đó, việc tìm kiếm theo riêng thành phần ngày và giờ có thể gặp khó khăn.
- **Với dữ liệu trong bảng DateSample như bên dưới.**

	ID	StartDate
1	1	1979-03-01 00:00:00.000
2	2	1999-02-28 00:00:00.000
3	3	2003-10-31 00:00:00.000
4	4	2005-02-28 10:00:00.000
5	5	2008-02-28 13:58:32.823
6	6	2008-03-01 09:02:25.000

→ **Cột StartDate có kiểu dữ liệu datetime**

- **Câu lệnh SELECT tìm kiếm theo thành phần ngày sau sẽ không trả về kết quả.**

```
SELECT * FROM DateSample WHERE StartDate = '2008-02-28'
```

- **Câu lệnh SELECT sử dụng hàm CONVERT loại bỏ thành phần giờ (SQL Server 2008 hoặc phiên bản mới hơn)**

```
SELECT * FROM DateSample  
WHERE CONVERT(date, StartDate) = '2008-02-28'
```

- **Câu lệnh SELECT tìm kiếm theo miền của ngày**

```
SELECT * FROM DateSample  
WHERE StartDate >= '2008-02-28' AND StartDate < '2008-02-29'
```

- **Câu lệnh SELECT sử dụng các hàm MONTH, DAY, and YEAR để tìm kiếm theo từng thành phần ngày, tháng, năm**

```
SELECT * FROM DateSample  
WHERE MONTH(StartDate) = 2 AND  
      DAY(StartDate) = 28 AND  
      YEAR(StartDate) = 2008
```

- **Câu lệnh SELECT sử dụng hàm CAST để loại bỏ thành phần giờ**

```
SELECT * FROM DateSample  
WHERE CAST(CAST(StartDate AS char(11)) AS datetime) = '2008-02-28'
```

- **Câu lệnh SELECT sử dụng hàm CONVERT để loại bỏ thành phần giờ**

```
SELECT * FROM DateSample  
WHERE CONVERT(datetime, CONVERT(char(10), StartDate, 110)) = '2008-02-28'
```

- Kỹ thuật thứ hai (tìm kiếm theo miền của ngày) là kỹ thuật duy nhất không sử dụng bất cứ hàm nào trong mệnh đề WHERE. Do đó, đây là kỹ thuật hiệu quả nhất trong việc tìm kiếm ngày.

- Với dữ liệu trong bảng DateSample như trước
  - Hai điều kiện tìm kiếm sau không trả về kết quả

```
SELECT * FROM DateSample  
WHERE StartDate = CAST('10:00:00' AS datetime)
```

```
SELECT * FROM DateSample  
WHERE StartDate >= '09:00:00' AND StartDate < '12:59:59:999'
```

- Nguyên nhân:
  - Giá trị giờ sẽ được chuyển đổi ngầm định sang kiểu date/time với thành phần ngày mặc định là 1900-01-01.

- Câu lệnh SELECT loại bỏ thành phần ngày. Sử dụng trong phiên bản SQL Server 2008.**

```
SELECT * FROM DateSample
WHERE CONVERT(time, StartDate) >= '09:00:00' AND
      CONVERT(time, StartDate) < '12:59:59:999'
```

	ID	StartDate
1	4	2005-02-28 10:00:00.000
2	6	2008-03-01 09:02:25.000

- Câu lệnh SELECT loại bỏ thành phần ngày. Sử dụng trong các phiên bản trước SQL Server 2008**

```
SELECT * FROM DateSample
WHERE CONVERT(datetime, CONVERT(char(12), StartDate, 8)) >= '09:00:00'
      AND CONVERT(datetime, CONVERT(char(12), StartDate, 8)) < '12:59:59:999'
```

	ID	StartDate
1	4	2005-02-28 10:00:00.000
2	6	2008-03-01 09:02:25.000

- Trong SQL Server dữ liệu ngày/giờ được xử lý dưới định dạng tháng/ngày/năm
- Để sử dụng định dạng ngày/giờ dưới dạng ngày/tháng/năm: Cần chú ý
  - Khi sử dụng câu lệnh INSERT phải truyền dữ liệu ngày/giờ theo định dạng tháng/ngày/năm
  - Khi truy vấn dữ liệu, để lấy về giá trị có định dạng ngày/tháng/năm có thể sử dụng hàm CONVERT với mã định dạng 3 hoặc 103

# Làm việc với kiểu dữ liệu số

---

- Sinh viên tự tìm hiểu cách làm việc với dữ liệu kiểu số trong SGK phần **Hướng dẫn làm việc với dữ liệu số**



# MÃ KỊCH BẢN

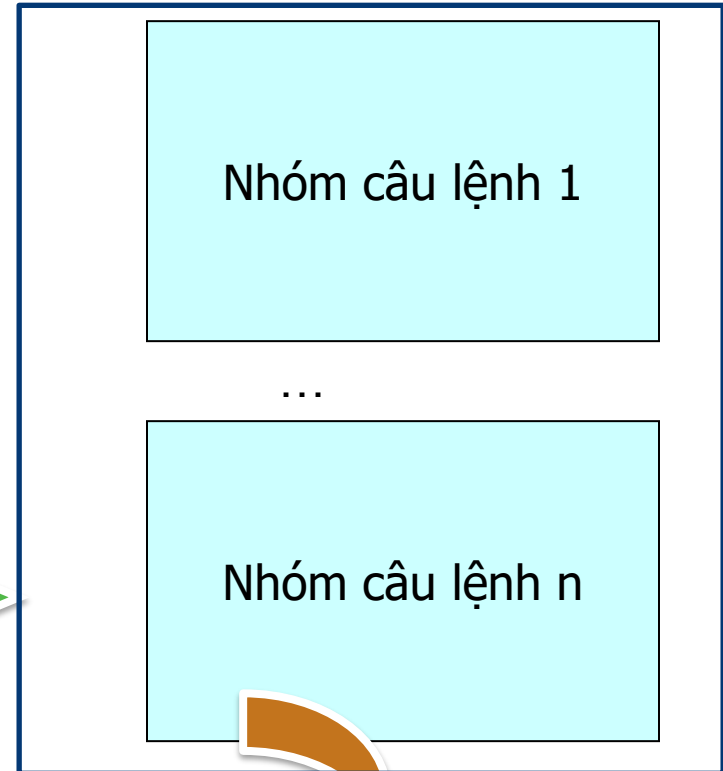
Các câu lệnh SQL  
riêng lẻ



Gom nhóm thành các  
nhóm câu lệnh  
(Batch )



Mã kịch bản



## KẾT QUẢ



Mỗi nhóm câu lệnh  
được biên dịch như  
một kế hoạch thực  
thi đơn

- Lưu ý:
  - Tất cả các ví dụ ở bài hai đều là các mã kịch bản, gồm một nhóm câu lệnh

# Ví dụ về mã kịch bản gồm nhiều nhóm câu lệnh

```
/*
Tạo ba bảng trong cơ sở dữ liệu có tên ClubRoster.
Tác giả: Bryan Syverson
Ngày tạo: 2006-08-12
Ngày sửa: 2008-09-26
*/
```

→ Chú thích cho  
mã kịch bản.

```
CREATE DATABASE ClubRoster
```

→ Nhóm câu lệnh  
(Batch) 1

```
GO
```

```
USE ClubRoster
```

```
CREATE TABLE Members
(MemberID int NOT NULL IDENTITY PRIMARY KEY,
LastName varchar(75) NOT NULL,
FirstName varchar(50) NOT NULL,
MiddleName varchar(50) NULL)
```

Nhóm câu lệnh  
(Batch) 2

```
CREATE TABLE Committees
(CommitteeID int NOT NULL IDENTITY PRIMARY KEY,
CommitteeName varchar(50) NOT NULL)
```

```
CREATE TABLE CommitteeAssignments
(MemberID int NOT NULL REFERENCES Members(MemberID),
CommitteeID int NOT NULL REFERENCES Committees(CommitteeID))
```

Lệnh Go đánh dấu  
điểm kết thúc của  
một nhóm câu  
lệnh.

**Mã kịch bản tạo ba bảng “Members”, “Committees”, “CommitteeAssignment”  
trong CSDL “ClubRoater”**

## Các câu lệnh T-SQL xử lý mã kịch bản

Đã học

Từ khóa	Mô tả
<b>USE</b>	Thay đổi cơ sở dữ liệu hiện thời thành cơ sở dữ liệu được chỉ định trong câu lệnh.
<b>PRINT</b>	Trả về thông báo tới client.
<b>DECLARE</b>	Định nghĩa biến cục bộ.
<b>SET</b>	Gán giá trị cho biến cục bộ hoặc biến theo phiên làm việc.
<b>EXEC</b>	Thực thi câu lệnh SQL hoặc Stored Procedure động.

## Các câu lệnh T-SQL điều khiển luồng thực thi

Đã học

Từ khóa	Mô tả
<b>IF...ELSE</b>	Điều khiển luồng thực thi dựa vào điều kiện.
<b>BEGIN...END</b>	Định nghĩa khối câu lệnh.
<b>WHILE</b>	Lặp lại các câu lệnh chừng nào điều kiện được chỉ định là đúng.
<b>BREAK</b>	Thoát khỏi vòng lặp <b>WHILE</b> trong cùng.
<b>CONTINUE</b>	Quay về điểm bắt đầu vòng lặp <b>WHILE</b> .
<b>TRY...CATCH</b>	Điều khiển luồng thực thi khi có lỗi xảy ra.
<b>GOTO</b>	Thay đổi luồng thực thi vô điều kiện.
<b>RETURN</b>	Thoát vô điều kiện.



- **Cú pháp của câu lệnh USE**  
**USE** <Tên CSDL>
- **Cú pháp của câu lệnh PRINT**  
**PRINT** <Biểu thức chuỗi>

## ■ Ví dụ về mã kịch bản sử dụng các câu lệnh T-SQL trên

USE AP

-- Khai báo biến @TotalDue

DECLARE @TotalDue money

-- Gán giá trị biến @TotalDue bằng tổng dư nợ chưa thanh toán của tất cả hóa đơn trong bảng

-- Invoices

```
SET      @TotalDue = (SELECT SUM(InvoiceTotal -  
                                PaymentTotal - CreditTotal)  
                                FROM Invoices)
```

-- Kiểm tra tổng dư nợ chưa thanh toán

-- Nếu lớn hơn 0 in ra dòng chữ "Total invoices due = \$" cùng với giá trị tổng dư nợ chưa thanh toán

-- được lưu trong biến @TotalDue

-- Nếu nhỏ hơn 0 in ra dòng chữ Invoices Paid in full

IF @TotalDue > 0

PRINT 'Total invoices due = \$' + CONVERT(varchar,@TotalDue,1)

ELSE

PRINT 'Invoices paid in full'



# Lưu trữ giá trị trong mã kịch bản

Đã học

Biến vô hướng

Lưu trữ

Giá trị đơn  
(Giá trị dữ liệu chuẩn)

Đã học

Biến bảng

Lưu trữ

Tập kết quả  
(Dữ liệu bảng)

Bảng tạm



- Dùng để lưu trữ một tập kết quả trả về (dữ liệu dưới dạng bảng) từ một câu lệnh SELECT
- Hai loại bảng tạm:
  - Bảng tạm cục bộ
    - Tên bắt đầu bằng dấu #
    - Tồn tại trong phiên làm việc hiện tại (Trong phạm vi cửa sổ soạn thảo truy vấn tạo ra bảng đó)
  - Bảng tạm toàn cục
    - Tên bắt đầu bằng dấu ##
    - Tồn tại trong tất cả các phiên làm việc
- Bảng tạm được lưu trong CSDL tempdb
- Để xóa một bảng tạm, sử dụng lệnh DROP TABLE

-- Lưu tập kết quả tra về của câu lệnh SELECT vào bảng tạm cục bộ #TopVendors

```
SELECT TOP 1 VendorID, AVG(InvoiceTotal) AS AvgInvoice
    INTO #TopVendors
    FROM Invoices
    GROUP BY VendorID
    ORDER BY AvgInvoice DESC
```

-- Truy xuất dữ liệu được lưu trong bảng tạm

```
SELECT Invoices.VendorID, MAX(InvoiceDate) AS LatestInv
    FROM Invoices JOIN #TopVendors
        ON Invoices.VendorID = #TopVendors.VendorID
    GROUP BY Invoices.VendorID
```

Bảng tạm có thể dùng thay thế vị trí của bảng thông thường trong câu lệnh SELECT, UPDATE, DELETE, INSERT

Results		Messages
	VendorID	LatestInv
1	110	2008-07-31 00:00:00

## ■ Cú pháp lệnh **TRY ... CATCH**

**BEGIN TRY**

**{<câu lệnh SQL> | <Khối câu lệnh>}**

**END TRY**

**BEGIN CATCH**

**{<Câu lệnh SQL> | <Khối câu lệnh>}**

**END CATCH**

- Hoạt động tương tự các hàm xử lý lỗi trong các ngôn ngữ lập trình
- **Các hàm có thể dùng trong khối CATCH**

Hàm	Mô tả
<b>ERROR_NUMBER()</b>	Trả về mã lỗi.
<b>ERROR_MESSAGE()</b>	Trả về thông điệp lỗi.
<b>ERROR_SEVERITY()</b>	Trả về mức nghiêm trọng của lỗi.
<b>ERROR_STATE()</b>	Trả về trạng thái của lỗi.

- Câu lệnh **TRY ... CATCH** chỉ bắt và xử lý được các lỗi có mức nghiêm trọng từ 10-20

# Xử dụng câu lệnh TRY ... CATCH

```
USE AP
GO
```

-- Bắt và xử lý lỗi chèn dữ liệu vào bảng Invoices

```
BEGIN TRY
```

```
    INSERT Invoices
```

```
    VALUES (799, 'Z XK-799', '2008-07-01', 299.95, 0, 0,
            1, '2008-08-01', NULL, NULL, NULL)
```

-- Nếu lệnh chèn thực thi thành công in ra dòng bên dưới

```
    PRINT 'SUCCESS: Record was inserted.'
```

```
END TRY
```


-- Nếu có lỗi xảy ra khi chèn dữ liệu in ra dòng thông báo lỗi cùng với thông tin mã lỗi và thông báo lỗi

```
BEGIN CATCH
```

```
    PRINT 'FAILURE: Record was not inserted.'
```

```
    PRINT 'Error ' + CONVERT(varchar, ERROR_NUMBER(), 1)
          + ': ' + ERROR_MESSAGE()
```

```
END CATCH
```

 Messages

```
FAILURE: Record was not inserted.
```

```
Error 245: Conversion failed when converting the varchar value 'Z XK-799' to data type int.
```

Tên hàm	Mô tả
<b>@@IDENTITY</b>	Trả về giá trị cuối cùng được tạo cho một cột định danh trên server. Trả về NULL nếu không có giá trị định danh nào được tạo.
<b>IDENT_CURRENT('tablename')</b>	Tương tự @@IDENTITY, nhưng trả về giá trị định danh cuối cùng được tạo cho một bảng xác định.
<b>@@ROWCOUNT</b>	Trả về số hàng chịu ảnh hưởng bởi câu lệnh SQL gần nhất.
<b>@@ERROR</b>	Trả về mã số lỗi được tạo bởi việc thực thi của câu lệnh SQL gần nhất. Trả về 0 nếu không xảy ra lỗi.
<b>@@SERVERNAME</b>	Trả về tên server cục bộ.
<b>HOST_NAME()</b>	Trả về tên trạm làm việc hiện thời.
<b>SYSTEM_USER</b>	Trả về tên người dùng hiện thời.

USE AP

DECLARE @MyIdentity int, @MyRowCount int

INSERT Vendors (VendorName, VendorAddress1, VendorCity, VendorState,  
VendorZipCode, VendorPhone, DefaultTermsID,  
DefaultAccountNo)

VALUES ('Peerless Binding', '1112 S Windsor St', 'Hallowell', 'ME',  
'04347', '(207) 555-1555', 4, 400)

/\* Gán giá trị cho biến @MyIdentity là giá trị định danh được tự động sinh cho cột VendorID của bảng Vendors và @MyRowCount số hàng bị ảnh hưởng bởi câu lệnh INSERT \*/

SET @MyIdentity = @@IDENTITY

SET @MyRowCount = @@ROWCOUNT

-- Kiểm tra xem câu lệnh INSERT ở trên có thực hiện thành công

IF @MyRowCount = 1

INSERT Invoices

VALUES (200, @MyIdentity, 'BA-0199', '2008-08-01', 4598.23,  
0, 0, 4, '2008-09-06', NULL, NULL)

## Các nội dung đã học trong bài:

### ■ Làm việc với các kiểu dữ liệu

- Khi làm việc với các biểu thức chứa nhiều kiểu dữ liệu khác nhau, phải thực hiện chuyển đổi giữa các kiểu dữ liệu.
- Hai loại chuyển đổi dữ liệu
  - Chuyển đổi ngầm (do SQL server tự thực hiện)
  - Chuyển đổi tường minh (sử dụng các hàm thư viện)
    - Sử dụng hàm CAST hoặc CONVERT
- Hai trường hợp chuyển đổi ngầm
  - Gán giá trị cho một cột có kiểu dữ liệu khác với giá trị được gán.
  - Biểu thức tính toán có sự tham gia của nhiều loại dữ liệu khác nhau
    - Chuyển đổi ngầm từ kiểu dữ liệu có độ ưu tiên thấp hơn sang kiểu dữ liệu có độ ưu tiên cao hơn



- SQL Server cung cấp các hàm làm việc với kiểu dữ liệu chuỗi, số, ngày giờ
- Các vấn đề xảy ra khi làm việc với kiểu dữ liệu chuỗi
  - Sắp thứ tự cho kiểu dữ liệu chuỗi chứa dữ liệu số
    - Sử dụng hàm CAST để chuyển đổi sang kiểu số trong mệnh đề ORDER BY
  - Phân tách dữ liệu chuỗi lưu trong một cột thành nhiều thành phần
    - Sử dụng các hàm CHARINDEX, LEFT, RIGHT, SUBSTRING, LEN
- Các vấn đề xảy ra với kiểu dữ liệu ngày giờ
  - Do kiểu dữ liệu ngày giờ thường chứa cả ngày và giờ -> Khi tìm kiếm theo thành phần ngày hoặc giờ, dễ không trả về kết quả
    - Sử dụng các hàm chuyển đổi dữ liệu, hàm trích xuất thành phần ngày, giờ hoặc tìm kiếm theo phạm vi

## ■ Mã kịch bản

- Tập hợp các câu lệnh riêng lẻ được gom nhóm thành nhóm câu lệnh.
- Mỗi nhóm câu lệnh được đánh dấu kết thúc bởi lệnh GO
- Một mã kịch bản chứa từ một đến nhiều nhóm câu lệnh
- Các câu lệnh T-SQL sử dụng trong mã kịch bản
  - Các câu lệnh xử lý mã kịch bản: USE, PRINT, DECLARE...
  - Các câu lệnh điều khiển luồng thực thi: IF...ELSE, TRY...CATCH, WHILE...
- Lưu trữ giá trị trong mã kịch bản
  - Biến vô hướng
  - Biến bảng
  - Bảng tạm

**XIN CẢM ƠN!**