# Bài 2: Lập trình với T-SQL

Ngày học @September 18, 2023

(T-SQL : Transact - SQL)

## 1. Một số quy tắc chung khi lập trình với T-SQL

- Một lệnh có thể viết trên nhiều dòng.
- Kết thúc lệnh có thể có dấu chấm phẩy hoặc không đều được.
- Một **khối lệnh** được viết trong cặp từ khóa:

> **begin**
>
> **end**

- **Kết thúc tập lệnh sử dụng từ khóa GO.**
- Không phân biệt chữ hoa & chữ thường.

## 2. Khai báo biến

- Biến phải khai báo trước khi sử dụng.
- Cú pháp

> **DECLARE @Tên_biến Kiểu_dữ_liệu [= Giá_trị_khởi tạo]**

- Lưu ý:

> - Tên biến phải bắt đầu bởi 1 dấu @
> - Có thể khai báo nhiều biến bởi cùng 1 lệnh **DECLARE**, phân cách bởi dấu phẩy.
> - Nếu ngắt lệnh bởi GO, thì muốn dùng lại biến đó, thì phải khai báo lại

Ví dụ:

```
DECLARE @firstName nvarchar(50),
        @lastname nvarchar(50),
        @age int = 40;

select @firstName, @lastName, @age;
```

## 3. Phép gán

Trong T-SQL, có 2 cách để viết phép gán:

### Cách 1: Dùng lệnh SET theo cú pháp

💡 **SET @Tên_biến = Biểu thức**

> **Lưu ý: Trong SQL, 1 câu lệnh SELECT trả về 1 dòng và tối đa 1 cột được xem là 1 biểu thức.**

Ví dụ:

```
DECLARE @name nvarchar(50),
        @id int;
SET @id = 1;
SET @name = (SELECT CustomerName from Customers WHERE CustomerID = @id);

PRINT @name;
GO;
```

> Mỗi lệnh **SET** chỉ thực hiện phép gán cho 1 biến (không viết gộp).

### Cách 2: Sử dụng lệnh SELECT

Cú pháp:

💡 **SELECT @Tên_biến_1 = Biểu_thức,**
**@Tên_biến_2 = Biểu_thức_2,**
**…….**
**@Tên_biến_n = Biểu_thức_N**
**FROM …..**
**WHERE …..**
**……..**

Phép gán này thường dùng để truy vấn dữ liệu và lưu kết quả vào biến.

Ví dụ:

```
DECLARE @firstName nvarchar(50),
        @lastname nvarchar(50),
        @id int = 10;

SELECT @firstName = Firstname, @lastName = LastName
FROM Employees
WHERE EmployeeID = @id;

PRINT @firstName;
PRINT @lastName;

GO
```

## 3. Cấu trúc điều khiển

Có 2 loại: rẽ nhánh & lặp

### a. Rẽ nhánh

💡 **IF điều_kiện**
**Khối_lệnh_của_IF**
**ELSE**
**Khối_lệnh_của_ELSE**

### b. Lặp

💡 **WHILE điều_kiện**
**Khối_lệnh_của_WHILE**

> Chú ý: Có thể sử dụng lệnh **BREAK** và **CONTINUE** trong vòng lặp WHILE.
> **NHỚ: viết điều kiện dừng, lặp**

Ví dụ: Viết vòng lặp chạy in ra các số từ 1 đến 10 và cho biết đó là số lẻ hay chẵn

```
DECLARE @i int = 1;
WHILE @i <= 10
    BEGIN
       IF (@i % 2 = 0)
          PRINT STR(@i) + N' là số chẵn'
       ELSE
          PRINT STR(@i) + N' là số lẻ'

       SET @i += 1;
   END
GO
```

## 4. Biến kiểu bảng

- Trong T-SQL không có các kiểu dữ liệu trừu tượng như trong các NNLT (struct, array, list, object,…)
- Biến kiểu bảng trong T-SQL có thể dùng để biểu diễn dữ liệu “phức tạp”.

Để khai báo biến kiểu bảng, sử dụng cú pháp:

💡 **DECLARE @tên_biến TABLE**
**(**
**Khai_báo_các_cột_của_bảng_(như_lệnh_CREATE_TABLE)**
**)**

```
DECLARE @tblThuNgay TABLE
(
  Ngay Date primary key,
  Thu nvarchar(50)
)
```

- Một biến kiểu bảng sử dụng để lưu trữ dữ liệu dưới dạng tập các dòng và các cột.
- Không thể sử dụng các lệnh GÁN thông thường cho các biến kiểu bảng.
- Trên biến kiểu bảng, chỉ dùng các lệnh SELECT, INSERT, UPDATE và DELETE.
- Biến bảng được khởi tạo với DECLARE độc lập.

💡 Hàm **DATEFROMPARTS()**

được sử dụng để trả về một ngày từ các phần được chỉ định. Thời gian được trả về sẽ theo thứ tự giá trị **năm-tháng-ngày.** Cú pháp:

**DATEFROMPARTS(year, month, day)**

Tham số:

- ***year*** là giá trị năm gồm 4 số
- ***month*** là giá trị tháng gồm 2 số
- ***day*** là giá trị ngày gồm 2 số

💡 **Hàm WEEKDAY()**

trả về số tương ứng với thứ của ngày

**Note:** 2 = Monday, 3 = Tuesday, 4 = Wednesday, 5 = Thursday, 6 = Friday, 7 = Saturday

💡 **Hàm DATEPART(dangthoigian, thoigian)**

trong SQL Server trả về một giá trị thời gian của đối số truyền vào, có thể là ngày, tháng, năm, quý, giờ, phút, giây, mili giây… Giá trị trả về là kiểu số nguyên (int).

VD: DATEPART(WEEKDAY, @ngay): trả về thứ trong @ngay

💡 **Hàm DATEADD** `(dangthoigian, number, thoigian)`
trả về giá trị thời gian mới khi nó được cộng thêm một khoảng thời gian được chỉ định.

- *dangthoigian:* dạng thời gian sử dụng để tính thêm vào *thoigian* những giá trị sau:

| Giá trị | Giải thích |
|---|---|
| year, yyyy, yy | Năm |
| quarter, qq, q | Quý |
| month, mm, m | Tháng |
| dayofyear | Ngày trong năm |
| day, dy, y | Ngày |
| week, ww, wk | Tuần |
| weekday, dw, w | Ngày trong tuần |
| hour, hh | Giờ |
| minute, mi, n | Phút |
| second, ss, s | Giây |
| millisecond, ms | Milli giây |

- *number:* số lượng khoảng thời gian mà bạn muốn thêm.
- *thoigian:* khoảng thời gian bạn muốn thêm *number* vào.

```
-- Ví dụ: Viết một đoạn chương trình để tạo ra 1 biến bảng có 2 cột:
-- Ngay date
-- Thu nvarchar(50)
-- Đưa vào bảng này danh sách các ngày và thứ (tiếng Việt) của một tháng @m năm @y nào đó
DECLARE @m int = 2,
    @y int = 2023;
DECLARE @tblThuNgay TABLE
(
  Ngay Date primary key,
```

```
  Thu nvarchar(50)
)

DECLARE @ngay date,
    @ngayCuoiThang date,
    @thu nvarchar(50);

SET @ngay = DATEFROMPARTS(@y, @m, 1)
SET @ngayCuoiThang = DATEADD(DAY, -1, DATEADD(MONTH, 1, @ngay));

-- SELECT @ngay, @ngayCuoiThang;
WHILE @ngay <= @ngayCuoiThang
  BEGIN
  -- Lấy thứ của @ngay
    SET @thu = CASE DATEPART(WEEKDAY, @ngay)
            WHEN 2 THEN N'Thứ hai'
            WHEN 3 THEN N'Thứ ba'
            WHEN 4 THEN N'Thứ tư'
            WHEN 5 THEN N'Thứ năm'
            WHEN 6 THEN N'Thứ sáu'
            WHEN 7 THEN N'Thứ bảy'
            ELSE N'Chủ nhật'
          END

    INSERT INTO @tblThuNgay(Ngay, Thu)
    VALUES(@ngay, @thu);
    SET @ngay = DATEADD(DAY, 1, @ngay);
  END

SELECT * FROM @tblThuNgay
```

💡 **ISNULL(bieuthuc, giatri_thaythe)**
trả về một giá trị được chỉ định nếu biểu thức là NULL. Nếu biểu thức không phải là NULL thì hàm trả về chính biểu thức.

```
--Cho 1 đoạn code sau:
DECLARE @month int = 2,
    @year int = 2018;

SELECT o.OrderDate, SUM(od.Quantity * od.SalePrice)
FROM Orders AS o JOIN OrderDetails as od
ON o.OrderId = od.OrderId
WHERE MONTH(o.OrderDate) = @month
      AND YEAR(o.OrderDate) = @year
GROUP BY o.OrderDate

go

-- Yêu cầu: Sửa lại đoạn code trên sao cho kết quả thống kê phải thể hiện đủ dữ liệu của tất cả các ngày trong tháng, những ngày không có
-- thì thể hiện số số tiền doanh thu là 0
-- Gợi ý:
--      + Sử dụng một biến bảng để lưu trữ đủ các ngày trong tháng
--      + Dùng phép nối ngoài (LEFT JOIN) giữa biến bảng và câu truy vấn trên

DECLARE @month int = 2,
    @year int = 2018;

DECLARE @tblNgay Table
  (
    Ngay date
  )
DECLARE @ngay date = DATEFROMPARTS(@year, @month, 1)
WHILE (month(@ngay) = @month)
  BEGIN
    insert into @tblNgay values (@ngay);
    set @ngay = DATEADD(day, 1, @ngay);
  END

select t1.Ngay, ISNULL (t2.revenue, 0) as revenue
from @tblNgay AS t1
LEFT JOIN
(
  select o.OrderDate, SUM(od.Quantity * od.SalePrice) as Revenue
```

```
  from Orders as o
  join OrderDetails as od ON o.OrderId = od.OrderId
  where month(o.OrderDate) = @month and YEAR(o.OrderDate) = @year
  group by o.OrderDate
) as t2
ON t1.Ngay = t2.OrderDate
GO
```

```
-- Lập trình để hiển thị số liệu thống kê doanh thu bán hàng của từng tháng trong năm @year
-- Yêu cầu: Số liệu phải đủ tất cả các tháng, những tháng không có doanh thu thì hiển thị với doanh thu là 0
DECLARE @year int = 2018,
    @month int = 1;
DECLARE @tblThang Table
  (
    Thang int
  )
WHILE (@month <= 12)
  BEGIN
    INSERT INTO @tblThang VALUES (@month);
    SET @month += 1;
  END

SELECT t1.Thang, ISNULL(t2.Revenue, 0) as Revenue
FROM @tblThang as t1
  LEFT JOIN
  (
    select MONTH(o.OrderDate) as thang,
        SUM(od.Quantity * od.SalePrice) AS Revenue
    FROM Orders AS o
      JOIN OrderDetails AS od on o.OrderId = od.OrderId
    WHERE YEAR(o.OrderDate) = @year
    GROUP BY MONTH(o.OrderDate)
  ) as t2
  ON t1.Thang = t2.thang
```

## 5. Bảng tạm (Temporary table)

Bảng tạm (thường lưu trữ trong bộ nhớ cache) thường được sử dụng để lưu trữ tạm thời dữ liệu dạng bảng, tuy nhiên phạm vi tồn tại của bảng tạm thì “lâu” hơn so với biến bảng.

- **Biến bảng**: phạm vi tồn tại trong khối lệnh mà nó được khai báo.
- **Bảng tạm**: phạm vi tồn tại là trong phiên làm viên (session).

VD:

```
declare @t table
(
  A int,
  B int
)
go

select * from @t; lệnh này gặp lỗi vì biến @t không tồn tại
```

Phiên làm việc (session) là khoảng thời gian từ khi kết nối đến CSDL cho đến khi kết thúc/đóng kết nối.

Khác với biến bảng, bảng tạm được tạo ra bằng cách tương tự như khi tạo ra bảng vật lý:

- **Dùng lệnh CREATE TABLE**
- **Dùng lệnh SELECT … INTO … (thường dùng)**

Bảng tạm khi tạo ra được quản lý bởi CSDL tempdb (Databases → System Databases → tempdb)

> **Tên bảng tạm phải bắt đầu bởi dấu #**

Tốc độ làm việc của biến bảng nhanh nhất

```
create table #T
(
  A int,
  B int
)
go

insert into #T values (1,1), (2,2), (3,3)
go

select * from #T;
```

### Một số tính chất cần lưu ý:

💡 - Tên phải bắt đầu bởi dấu #
- Sử dụng được trong phiên làm việc mà tạo ra nó
- Giải phóng (xóa) khi kết thúc phiên làm việc hoặc do chúng ta chủ động xóa bằng lệnh **DROP TABLE**
- Sử dụng bảng tạm tương tự như bảng vật lý: SELECT, INSERT, UPDATE, DELETE

```
SELECT ProductId, ProductName, Price
INTO #TempProducts
FROM Products
WHERE Price between 5 and 10

DROP TABLE #TempProducts
```

Cần truy vấn dữ liệu từ database và lưu vào 1 cấu trúc dạng bảng để xử lý

- Dùng biến bảng:
  - Khai báo 1 biến bảng có cấu trúc phù hợp với kết quả của truy vấn.
  - Đưa dữ liệu vào biến bảng bằng lệnh: **INSERT ….**

    **SELECT ….**
- Dùng bảng tạm: Dùng lệnh **INSERT … INTO**

## 6. Common Table Expression (CTE)

- Truy vấn con (sub-query): sử dụng khi cần thực thi 1 câu lệnh SELECT và sử dụng kết quả câu lệnh đó bên trong 1 câu lệnh khác (SELECT, INSERT, UPDATE, DELETE)
- CTE cũng được sử dụng để thực thi 1 câu lệnh SELECT, lưu tạm thời để sử dụng cho 1 câu lệnh khác.

```
SELECT p.*, tk.SumOfQuatity
FROM Products as p
  LEFT JOIN (SELECT ProductId, SUM(Quantity) as SumOfQuatity
        FROM OrderDetails
        GROUP BY ProductId
      ) AS tk ON p.ProductId = tk.ProductId

-- Viết lại bằng cách dùng CTE:
WITH cte_Products AS
(
  SELECT ProductId, SUM(Quantity) AS SumOfQuatity
  FROM OrderDetails
  GROUP BY ProductId
)

SELECT p.*, tk.SumOfQuatity
FROM Products AS p
  LEFT JOIN cte_Products AS tk ON p.ProductId = tk.ProductId
```

### Cú pháp để khai báo CTE

```
WITH cte_name1 AS
(
      Lệnh_SELECT
)
, cte_name2 AS
(
      Lệnh_SELECT
)
….
,  cte_nameN AS
(
      Lệnh_SELECT
)

Lệnh_sử_dụng_các_CTE
```

> **LƯU Ý: Lệnh SELECT để lấy dữ liệu cho các CTE dưới có thể truy vấn dữ liệu từ các CTE đã định nghĩa trước đó**
> **Trước câu lệnh của CTE, phải có dấu ;**

```
WITH cte_Category AS
(
  SELECT  CategoryId, CategoryName
  FROM Categories
)
, cte_Products AS
(
  SELECT ProductName, CategoryId, Price
  FROM Products
  WHERE Price < 5
)

SELECT *
FROM cte_Category as t1
  JOIN cte_Products as t2
  ON t1.CategoryId = t2.CategoryId
```

Một điểm mạnh của CTE là cho phép viết truy vấn dạng đệ quy dựa vào đặc tính trong lênh để lấy dữ liệu cho CTE có thể truy vấn đến chính nó.

VD: Cho 2 ngày @date1 và @date2 (@date1 < @date2). Hiển thị 1 bảng có đủ tất cả các ngày từ @date1 đến @date2.

```
DECLARE @date1 date = '2018/02/01',
    @date2 date = '2018/02/15'; --thiếu ; sẽ lỗi

WITH cte_Ngay AS
(
  SELECT 1 AS STT, @date1 AS Ngay
  UNION ALL
  SELECT STT + 1, DATEADD(DAY, 1, t.Ngay)
  FROM Cte_Ngay AS t
  WHERE t.Ngay < @date2
)
SELECT * FROM cte_Ngay
```

VD: Cho 2 ngày @date1 và @date2 (@date1 < @date2). Hiển thị 1 bảng có đủ tất cả các ngày từ @date1 đến @date2 và số lượng đơn hàng tương ứng.

```
DECLARE @date1 date = '2018/02/01',
    @date2 date = '2018/02/15'; --thiếu ; sẽ lỗi

WITH cte_Ngay AS
(
  SELECT 1 AS STT, @date1 AS Ngay
  UNION ALL
  SELECT STT + 1, DATEADD(DAY, 1, t.Ngay)
  FROM Cte_Ngay AS t
  WHERE t.Ngay < @date2
)
, cte_ThongKe AS
(
  SELECT OrderDate AS Ngay, COUNT(*) AS SoLuongDonHang
  FROM Orders
  WHERE OrderDate BETWEEN @date1 AND @date2
  GROUP BY OrderDate
)
SELECT t1.*, ISNULL(t2.SoLuongDonHang, 0) AS SoLuongDonHang
FROM cte_Ngay AS t1
  LEFT JOIN cte_ThongKe as t2
  ON t1.Ngay = t2.Ngay
```

VD: Cho bảng DonVi như sau:

```
CREATE TABLE DonVi
(
  MaDonVi nvarchar(50) primary key,
  TenDonVi nvarchar(50) not null,
  MaDonViCha nvarchar(50) null
)
GO

INSERT INTO DonVi
VALUES  ('DV01', N'Khoa CNTT', NULL),
    ('DV02', N'Khoa Toán', NULL),
    ('DV03', N'Bộ môn CNPM', 'DV01'),
    ('DV04', N'Bộ môn KHMT', 'DV01'),
    ('DV05', N'Bộ môn Đại số', 'DV02'),
    ('DV06', N'Bộ môn Giải tích', 'DV02'),
    ('DV07', N'Tổ thuật toán', 'DV04'),
    ('DV08', N'Tổ AI', 'DV04')
  SELECT * FROM DonVi
```

!!! VD: Truy vấn dữ liệu từ bảng DonVi cho biết mỗi đơn vị là ở cấp mấy (theo cây) và đường dẫn đến đơn vị đó là như thế nào

```
INSERT INTO DonVi
VALUES  ('DV01', N'Khoa CNTT', NULL),
    ('DV02', N'Khoa Toán', NULL),
    ('DV03', N'Bộ môn CNPM', 'DV01'),
    ('DV04', N'Bộ môn KHMT', 'DV01'),
    ('DV05', N'Bộ môn Đại số', 'DV02'),
    ('DV06', N'Bộ môn Giải tích', 'DV02'),
    ('DV07', N'Tổ thuật toán', 'DV04'),
    ('DV08', N'Tổ AI', 'DV04')
  SELECT * FROM DonVi;
-- Truy vấn dữ liệu từ bảng DonVi cho biết mỗi đơn vị là ở cấp mấy (theo cây) và đường dẫn đến đơn vị đó là như thế nào
WITH cte_DonVi AS
(
  SELECT 1 AS Cap,
      CAST(MaDonVi AS nvarchar(1000)) AS DuongDan,
      MaDonVi, TenDonVi, MaDonViCha
  FROM DonVi
  WHERE MaDonViCha IS NULL
  UNION ALL
  SELECT dvCha.Cap + 1,
      CAST(CONCAT(dvCha.DuongDan, '\', dvCon.MaDonVi) AS nvarchar(1000)),
      dvCon.MaDonVi, dvCon.TenDonVi, dvCon.MaDonViCha
  FROM DonVi AS dvCon
      JOIN cte_DonVi AS dvCha ON dvCon.MaDonViCha = dvCha.MaDonVi
)
```

```
-- SELECT * FROM cte_DonVi


SELECT * FROM cte_DonVi
ORDER BY DuongDan;
```

## 7. Sử dụng con trỏ để duyệt dữ liệu

Khi truy vấn dữ liệu, chúng ta thường có nhu cầu lấy dữ liệu tại mỗi dòng để thực hiện các phép xử lý. Điều này dẫn đến cần phép duyệt từng dòng trong bảng.

Trong T-SQL, con trỏ (CURSOR) dùng để duyệt dữ liệu:

Ví dụ: Truy vấn tên và giá các mặt hàng có giá nhỏ hơn 20 và in (PRINT) ra màn hình:

💡 - B1: Khai báo biến con trỏ, trỏ vào kết quả truy vấn:
DECLARE contro CURSOR FOR
SELECT ProductName, Price
FROM Products
WHERE Price < 20;

- - B2: Mở con trỏ:
  OPEN contro;
- - B3: Dịch con trỏ vào dòng đầu tiên và đọc dữ liệu (lưu vào biến)
  DECLARE @name nvarchar(50),
  @price money;

FETCH NEXT FROM contro INTO @name, @price;

- - B4: Lặp và đọc các dòng tiếp theo:
  WHILE @@FETCH_STATUS = 0
  BEGIN
  PRINT @name;
  PRINT @price;

  FETCH NEXT FROM contro INTO @name, @price;
  END
- - B5: Đóng con trỏ (sau khi đóng, có thể OPEN lại)
  CLOSE contro;
- - B6: Giải phóng con trỏ nếu không còn sử dụng
  DEALLOCATE contro;

> Lưu ý: Chỉ nên dùng con trỏ trong trường hợp thực sự cần thiết. Nếu có cách giải quyết bằng truy vấn (con, cte, biến bảng,…) thì nên ưu tiên bằng truy vấn.