TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
Khoa Khoa học và Kỹ thuật Thông tin

Thực hành - IE103.O21.CNVN

MỘT SỐ KIẾN THỨC BỔ TRỢ CHO LẬP TRÌNH CSDL

Nội dung

Một số câu lệnh thường dùng trong ngôn ngữ SQL

Các cấu trúc điều khiển

Các hàm thường dùng trong ngôn ngữ SQL

Một số câu lệnh thường dùng trong ngôn ngữ SQL

Câu lệnh PRINT

- Lệnh `PRINT` được sử dụng để in nội dung (chuỗi ký tự, biến cục bộ, biểu thức,...).
  - **Ví dụ**
    - Ví dụ 1. In chuỗi `'Hello world'`.

```
PRINT 'Hello world'
```

    - Ví dụ 2. In số `1`.

```
PRINT 1
```

    - Ví dụ 3. In kết quả của phép toán `1+1`.

```
PRINT 1+1
```

Câu lệnh PRINT

- Ví dụ 4. In kết quả của phép cộng chuỗi 'Hello' + ' ' + 'world'.

```
PRINT 'Hello' + ' ' + 'world'
```

- Ví dụ 5. In thời gian hiện tại bằng cách sử dụng hàm GETDATE().

```
PRINT GETDATE()
```

- Ví dụ 6. In chuỗi 'Count' và số 1 bằng cách sử dụng hàm CAST() để chuyển đổi kiểu dữ liệu số 1 thành kiểu VARCHAR(5).

```
PRINT 'Count' + CAST(1 AS VARCHAR(5))
```

- Ví dụ 7. In các chuỗi và số liên tiếp nhau: 'Count ', 1, ' AND ', 1 bằng cách sử dụng hàm CONCAT() để nối các chuỗi và số với nhau.

```
PRINT CONCAT('Count ', 1, ' AND ', 1)
```

Câu lệnh PRINT

- Để in chuỗi ký tự có chứa tiếng Việt có dấu, nên thêm ký tự N vào trước chuỗi ký tự.
  - **Ví dụ**

```
PRINT N'Xin chào Việt Nam'
```

Câu lệnh DECLARE

- Dùng để khai báo biến cục bộ.
- Khai báo một biến (với giá trị khởi tạo):

```
DECLARE @Tên_biến Kiểu_dữ_liệu [ = Giá_trị ]
```

Câu lệnh DECLARE

- **Ví dụ**
  - Ví dụ 1. Khai báo biến `I` có kiểu dữ liệu `INT` với giá trị khởi tạo là `100`.

```
DECLARE @I INT = 100
PRINT @I
```

  - Ví dụ 2. Khai báo biến `TestVariable` có kiểu dữ liệu `VARCHAR(100)` với giá trị khởi tạo là `'Save Our Planet'`.

```
DECLARE @TestVariable VARCHAR(100) = 'Save Our Planet'
PRINT @TestVariable
```

Câu lệnh DECLARE

- Khai báo nhiều biến (với giá trị khởi tạo):

```
DECLARE @Tên_biến_1 Kiểu_dữ_liệu_biến_1 [ = Giá_trị_biến_1 ],
        @Tên_biến_2 Kiểu_dữ_liệu_biến_2 [ = Giá_trị_biến_2 ],
        . . .
        @Tên_biến_n Kiểu_dữ_liệu_biến_n [ = Giá_trị_biến_n ]
```

Câu lệnh DECLARE

- **Ví dụ**
  - Ví dụ 1. Khai báo biến `var1` và biến `var2` có kiểu dữ liệu `INT` với giá trị khởi tạo lần lượt là `3` và `5`.

```
DECLARE @var1 INT = 3, @var2 INT = 5
PRINT 'Var1 = ' + CAST(@var1 AS VARCHAR(5))
PRINT 'var2 = ' + CAST(@var2 AS VARCHAR(5))
```

(Sử dụng hàm `CAST()` để chuyển đổi kiểu dữ liệu)

Câu lệnh DECLARE

- Ví dụ 2. Khai báo biến phrase1 và biến phrase2 có kiểu dữ liệu NVARCHAR(10) với giá trị khởi tạo lần lượt là 'Xin chào' và 'Việt Nam'.

```
DECLARE @phrase1 NVARCHAR(10) = N'Xin chào',
        @phrase2 NVARCHAR(10) = N'Việt Nam'
PRINT @phrase1 + ' ' + @phrase2
```

Câu lệnh DECLARE

- Ngoài cách khai báo biến đi kèm giá trị khởi tạo, chúng ta có thể khai báo biến trước, sau đó gán giá trị cho biến bằng cách:
  - Sử dụng lệnh SET:

```
DECLARE @Tên_biến Kiểu_dữ_liệu
SET @Tên_biến = Giá_trị
```

Ví dụ. Khai báo biến I có kiểu dữ liệu INT, sau đó gán giá trị 25.

```
DECLARE @I INT
SET @I = 25
PRINT @I
```

Câu lệnh DECLARE

- Sử dụng lệnh SELECT:

```
DECLARE @Tên_biến Kiểu_dữ_liệu
SELECT @Tên_biến = Giá_trị
```

Ví dụ. Khai báo biến I có kiểu dữ liệu INT, sau đó gán giá trị 25.

```
DECLARE @I INT
SELECT @I = 25
PRINT @I
```

Các cấu trúc điều khiển

Cấu trúc lựa chọn IF... ELSE...

- Được sử dụng để thể hiện sự lựa chọn.
- Cú pháp:

```
IF Điều_kiện
    BEGIN
            Các_câu_lệnh_SQL
    END
ELSE
    BEGIN
            Các_câu_lệnh_SQL
    END
```

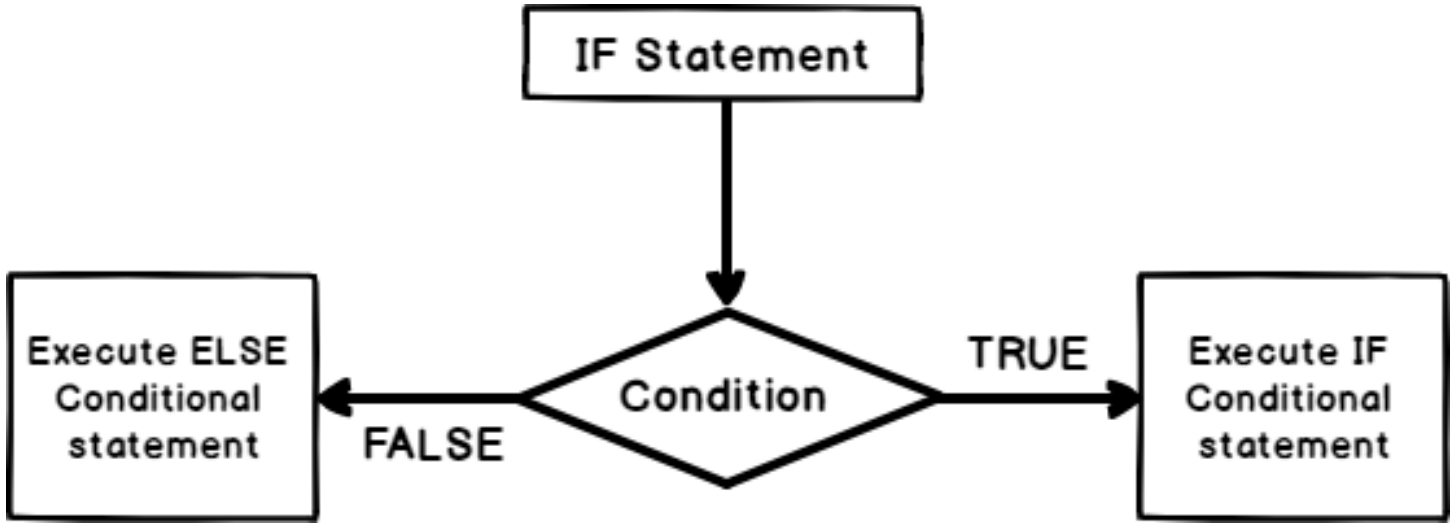
Thực hiện nếu điều kiện **đúng**

Thực hiện nếu điều kiện **sai**

Cấu trúc lựa chọn IF... ELSE...

- Nếu điều kiện **đúng** → Thực hiện đoạn lệnh trong khối `IF`.
- Nếu điều kiện **sai** → Thực hiện đoạn lệnh trong khối `ELSE`.

❑ Sơ đồ khối:



Cấu trúc lựa chọn IF... ELSE...

- **Lưu ý**
  - Từ điều kiện kiểm tra, cần nắm được:
    - Khi nào thực hiện khối lệnh `IF...` (khi nào điều kiện đúng ?) ?
    - Khi nào thực hiện khối lệnh `ELSE...` (khi nào điều kiện sai ?) ?
  - Nếu khối lệnh bên trong chỉ gồm 1 câu lệnh, có thể bỏ `BEGIN... END`.

Cấu trúc lựa chọn IF... ELSE...

- **Ví dụ**
  - Ví dụ 1. Biểu thức điều kiện không chứa biến.

```
IF 1 = 1
    PRINT 'Executed the statement as condition is TRUE'
ELSE
    PRINT 'Executed the statement as condition is FALSE'
```

  - Ví dụ 2. Biểu thức điều kiện không chứa biến.

```
IF 2 <= 0
    PRINT 'Executed the statement as condition is TRUE'
ELSE
    PRINT 'Executed the statement as condition is FALSE'
```

Cấu trúc lựa chọn IF... ELSE...

- Ví dụ 3. Biểu thức điều kiện chứa biến.

```
DECLARE @StudentMarks INT = 91

IF @StudentMarks >= 80
   PRINT 'Passed! Congratulations!'
ELSE
   PRINT 'Failed! Try again!’
```

```
IF @StudentMarks < 90
   PRINT 'Passed! Congratulations!'
ELSE
   PRINT 'Failed! Try again!'
```

Cấu trúc lựa chọn IF... ELSE...

- **Lưu ý**
  - ELSE... sẽ đi theo cặp với IF... gần nhất...
  - Ví dụ 4. Câu lệnh lồng.

```
DECLARE @StudentMarks INT = 91
IF @StudentMarks >= 80
    IF @StudentMarks >= 90
        PRINT 'Excellent!'
    ELSE
        PRINT 'Passed!'
ELSE
    PRINT 'Failed!'
```

Cấu trúc lựa chọn IF... ELSE...

- **Lưu ý**
  - Có thể bỏ đoạn lệnh `ELSE...`, khi đó cấu trúc lựa chọn chỉ còn `IF...`

```
IF Điều_kiện
    BEGIN
            Các_câu_lệnh_SQL
    END
```

→ Khi điều kiện sai thì sẽ không thực hiện gì cả.

Cấu trúc lựa chọn IF... ELSE...

- Ví dụ 5. Cấu trúc lựa chọn có IF... nhưng không có ELSE...

```
DECLARE @StudentMarks INT = 95
IF @StudentMarks >= 90
    PRINT 'Congratulations! You are in Merit list!'
```

- Ví dụ 6. Cấu trúc lựa chọn có IF... nhưng không có ELSE...

```
DECLARE @StudentMarks INT = 85
IF @StudentMarks >= 90
    PRINT 'Congratulations! You are in Merit list!'
```

Cấu trúc lựa chọn IF... ELSE...

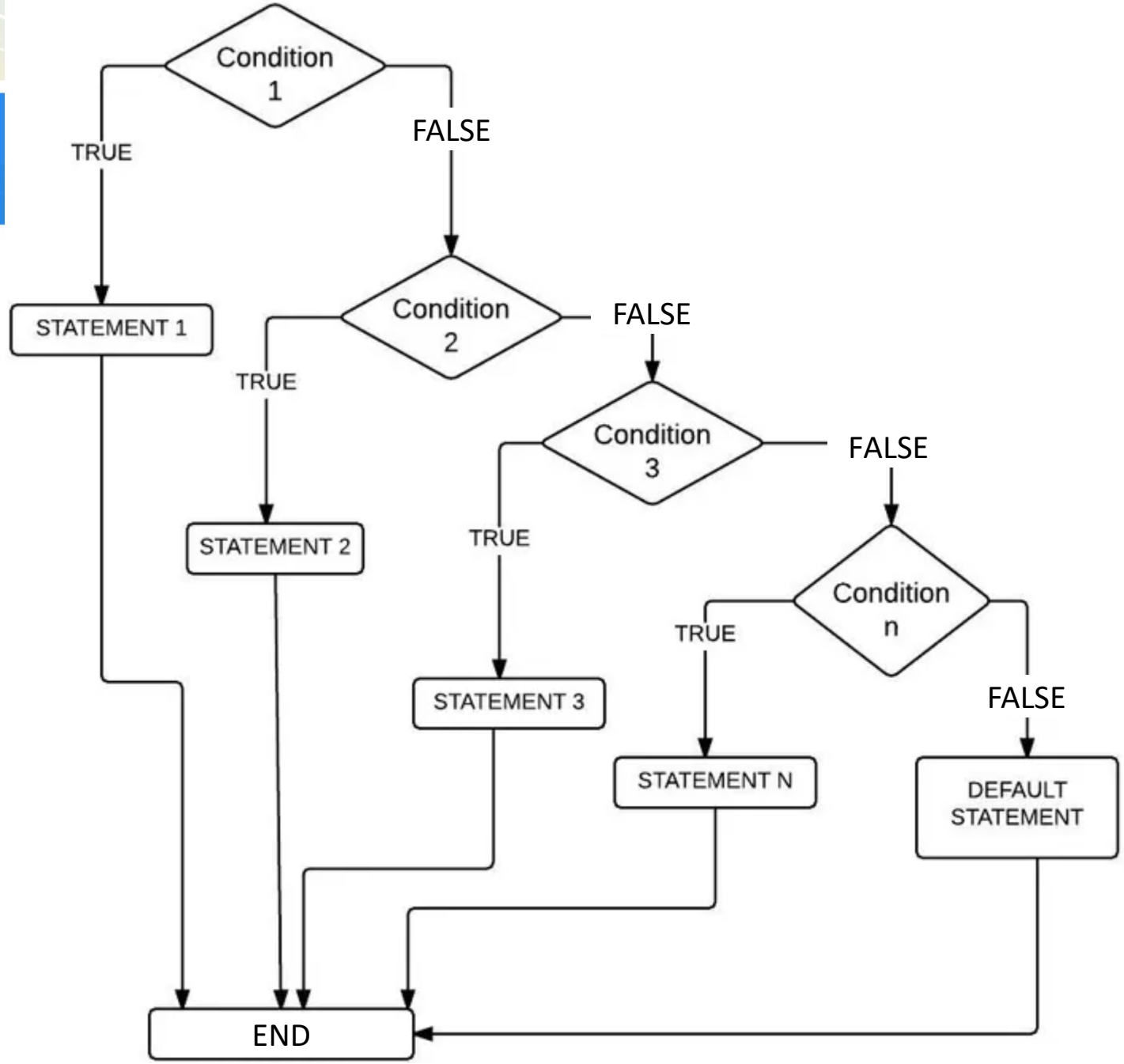
- **Cấu trúc IF... ELSE... mở rộng (IF... ELSE IF... ELSE...)**
  - Cú pháp:

```
IF Điều_kiện_1
    BEGIN
        Các_câu_lệnh_SQL
    END
ELSE IF Điều_kiện_2
    BEGIN
        Các_câu_lệnh_SQL
    END
...
ELSE
    BEGIN
        Các_câu_lệnh_SQL
    END
```

Khác với các điều kiện 1, 2,...

Cấu trúc lựa chọn IF... ELSE...

- Sơ đồ khối:



Cấu trúc lựa chọn IF... ELSE...

- Ví dụ:

```
DECLARE @A INT = 0

IF @A > 0
    PRINT 'A > 0'
ELSE IF @A < 0
    PRINT 'A < 0'
ELSE <-------------- @A = 0
    PRINT 'A = 0'
```

Cấu trúc lặp WHILE...

- Được sử dụng để lặp lại việc thực thi câu lệnh nhiều lần.
- Cú pháp:

```
WHILE Điều_kiện_lặp
BEGIN
    Các_câu_lệnh_SQL
END
```

  - Khi điều kiện lặp đúng (điều kiện lặp được thỏa mãn) thì sẽ thực hiện các câu lệnh bên trong.
  - Vòng lặp sẽ dừng lại khi điều kiện lặp bị sai.

Cấu trúc lặp WHILE...

- Sơ đồ khối:

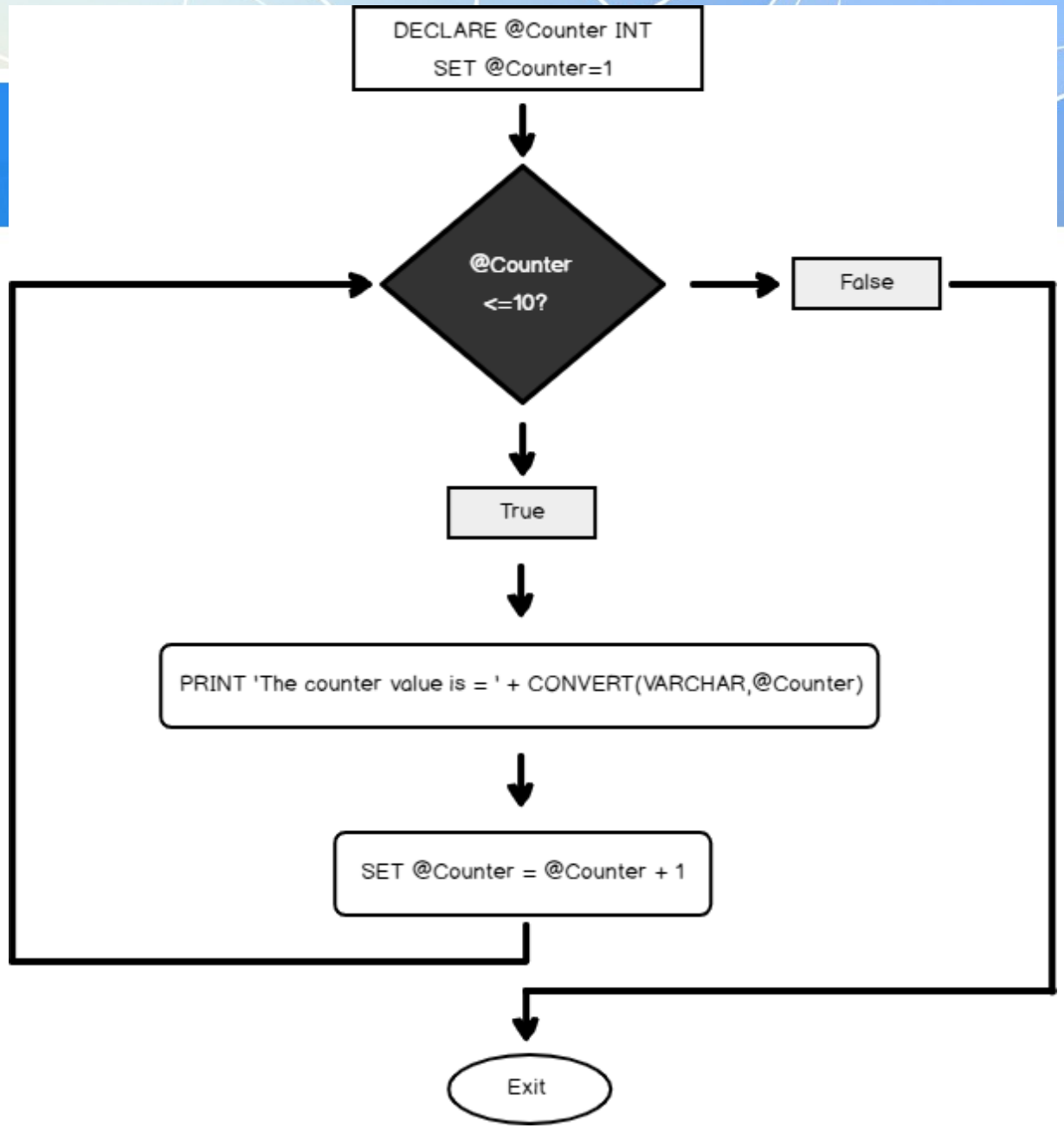
Cấu trúc lặp WHILE...

- **Ví dụ.** Vòng lặp theo biến lặp @Counter với giá trị khởi tạo là 1 và điều kiện lặp @Counter <= 10, đồng thời in giá trị của biến @Counter và tăng giá trị của biến @Counter lên 1 đơn vị.

```
DECLARE @Counter INT
SET @Counter = 1
WHILE @Counter <= 10
BEGIN
   PRINT 'The counter value is ' + CONVERT(VARCHAR, @Counter)
   SET @Counter = @Counter + 1
END
```

Cấu trúc lặp WHILE...

- Sơ đồ khối của vòng lặp:



Cấu trúc lặp WHILE...

- Trong ví dụ trên:
  - Biến lặp: `@Counter`
  - Giá trị khởi tạo: `1`
  - Điều kiện lặp: `@Counter <= 10`
  - Công việc thực hiện: In giá trị của biến `@Counter` và tăng giá trị của biến `@Counter` lên 1 đơn vị.
  - Bước thay đổi giá trị biến lặp: `@Counter += 1`
  - Khi nào dừng lại ? `@Counter > 10`

Cấu trúc lặp WHILE...

- **Lưu ý**
  - Cần chú ý đến điều kiện dừng của vòng lặp (tính dừng của vòng lặp), tránh để xảy ra trường hợp vòng lặp vô tận.

    (Khi nào vòng lặp dừng lại ? → Khi không thỏa điều kiện lặp nữa)
  - Nếu khối lệnh bên trong chỉ gồm 1 câu lệnh, có thể bỏ `BEGIN... END`.

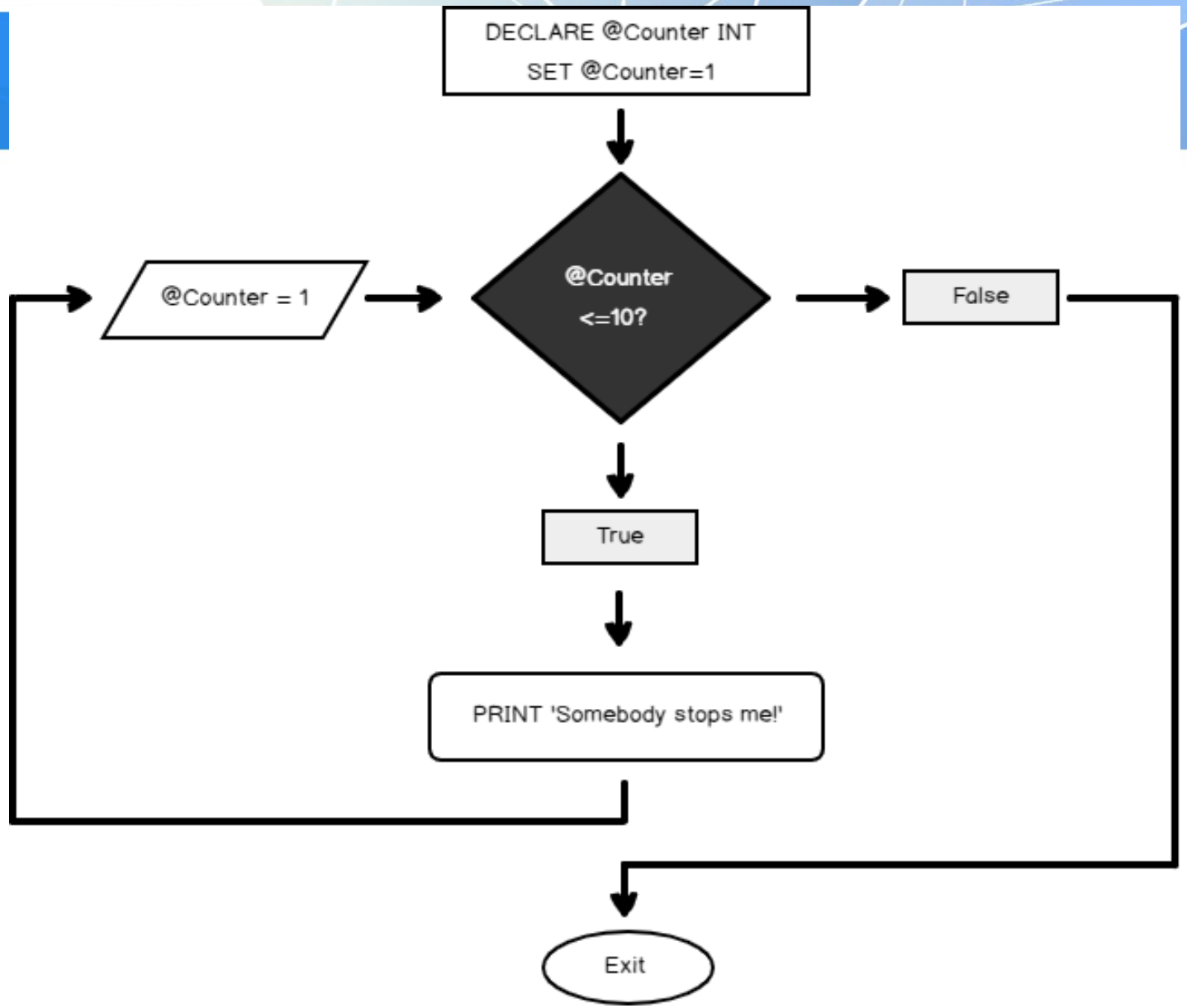
Cấu trúc lặp WHILE...

- **Ví dụ.** Vòng lặp vô tận (Điều kiện lặp luôn luôn thỏa → Vòng lặp không bao giờ dừng lại).

```
DECLARE @Counter INT
SET @Counter = 1
WHILE @Counter <= 10
   PRINT 'Somebody stops me!'
```

Cấu trúc lặp WHILE...

- Sơ đồ khối của vòng lặp:



Cấu trúc lặp WHILE...

- **Câu lệnh lặp FOR...**
  - Trong SQL Server, không có cấu trúc lặp FOR...
  - Biểu diễn cấu trúc lặp FOR... theo cấu trúc lặp WHILE...:

```
DECLARE @Biến_lặp INT = 0
WHILE @Biến_lặp < Số_lần_lặp
BEGIN
    Các_câu_lệnh_SQL
    SET @Biến_lặp = @Biến_lặp + 1
END
```

Cấu trúc lặp WHILE...

- **Ví dụ.** Cài đặt vòng lặp với 10 lần lặp.

```
DECLARE @I INT = 0
WHILE @I < 10
BEGIN
    PRINT 'Loop no. ' + CAST(@I AS VARCHAR(4))
    SET @I = @I + 1
END
```

Các hàm thường dùng trong ngôn ngữ SQL

Một số hàm toán học

- **Hàm `ABS()`:** Trả về giá trị tuyệt đối của một số.
  - **Ví dụ:** `SELECT ABS(-1234.56)`.
    - Kết quả: `1234.56`.
- **Hàm `PI()`:** Trả về số Pi trong toán học.
  - **Ví dụ:** `SELECT PI()`.
    - Kết quả: `3.14159265358979`.
- **Hàm `POWER(m, n)`:** Trả về kết quả của phép tính lũy thừa m^n.
  - **Ví dụ:** `SELECT POWER(3, 2)`.
    - Kết quả: `9`.

Một số hàm toán học (TT)

- **Hàm ROUND():** Trả về số được làm tròn.
  - **Ví dụ 1:** `SELECT ROUND(123.4567, 2)`.
    - Kết quả: `123.4600`.
  - **Ví dụ 2:** `SELECT ROUND(123.4567, -1)`.
    - Kết quả: `120.000`.

Một số hàm toán học (TT)

- **Hàm SQUARE(m):** Trả về giá trị bình phương (lũy thừa 2) của số m.
  - **Ví dụ:** `SELECT SQUARE(64)`.
    - Kết quả: 4096.
- **Hàm SQRT(m):** Trả về kết quả căn bậc hai số m.
  - **Ví dụ:** `SELECT SQRT(9)`.
    - Kết quả: 3.

Một số hàm toán học (TT)

- **Hàm `FLOOR(m)`:** Trả về số nguyên lớn nhất sao cho nhỏ hơn hoặc bằng m.
  - **Ví dụ:** `SELECT FLOOR(12.3)`.
    - Kết quả: 12.
- **Hàm `CEILING(m)`:** Trả về số nguyên nhỏ nhất sao cho lớn hơn hoặc bằng m.
  - **Ví dụ:** `SELECT CEILING(23.45)`.
    - Kết quả: 24.

Một số hàm toán học (TT)

- **Hàm `SIGN(m)`:** Trả về số biểu thị dấu của số m theo quy tắc sau:
  - Nếu m > 0: Trả về 1.0.
  - Nếu m = 0: Trả về 0.
  - Nếu m < 0: Trả về -1.0.
  - **Ví dụ 1:** `SELECT SIGN(255.5).`
    - Kết quả: `1.0`.
  - **Ví dụ 2:** `SELECT SIGN(-12).`
    - Kết quả: `-1.0`.
  - **Ví dụ 3:** `SELECT SIGN(0).`
    - Kết quả: `0`.

Hàm xử lý chuỗi ký tự

- **Hàm `UPPER()`:** Trả về chữ in hoa.
  - **Ví dụ:** `SELECT UPPER('Hello')`.
    - Kết quả: `'HELLO'`.
- **Hàm `LOWER()`:** Trả về chữ in thường.
  - **Ví dụ:** `SELECT LOWER('Hello')`.
    - Kết quả: `'hello'`.
- **Hàm `LEN()`:** Trả về số lượng ký tự của chuỗi.
  - **Ví dụ:** `SELECT LEN('Hello')`.
    - Kết quả: 5.

Hàm xử lý chuỗi ký tự (TT)

- **Hàm `LEFT(M, m)`, `RIGHT(M, m)`:** Cắt ra chuỗi con gồm m ký tự tính từ bên trái/bên phải chuỗi M.
  - **Ví dụ 1:** `SELECT LEFT('Hello world', 3)`.
    - Kết quả: `'Hel'`.
  - **Ví dụ 2:** `SELECT RIGHT('Hello world', 5)`.
    - Kết quả: `'world'`.

Hàm xử lý chuỗi ký tự (TT)

- **Hàm `LTRIM(M)`, `RTRIM(M)`:** Loại bỏ khoảng trắng thừa ở đầu (bên trái) hoặc ở cuối (bên phải) chuỗi M.
  - **Ví dụ 1:** `SELECT LTRIM('   123');`
    - Kết quả: `'123'`.
  - **Ví dụ 2:** `SELECT RTRIM('123   ');`
    - Kết quả: `'123'`.
- **Hàm `TRIM(M)` :** Loại bỏ khoảng trắng thừa ở đầu (bên trái) và ở cuối (bên phải) chuỗi M.
  - **Ví dụ:** `SELECT TRIM('   123   ');`
    - Kết quả: `'123'`.

Hàm xử lý chuỗi ký tự (TT)

- **Hàm `CONCAT(str1, str2,…)`:** Nối các chuỗi ký tự str1, str2,... với nhau.
  - **Ví dụ:** `SELECT CONCAT('SQL', ' is', ' fun!')`.
    - Kết quả: `'SQL is fun!'`.
- **Hàm `CONCAT_WS(sep, str1, str2,…)`:** Nối hai hoặc nhiều chuỗi ký tự str1, str2,... với nhau bằng ký tự sep ngăn cách xen giữa.
  - **Ví dụ:** `SELECT CONCAT_WS('-', 'SQL', 'is', 'fun!')`.
    - Kết quả: `'SQL-is-fun!'`.

Hàm xử lý chuỗi ký tự (TT)

- **Hàm `REPLACE(str, old_str, new_str)`:** Thay thế chuỗi con old_str trong chuỗi str bởi chuỗi con mới new_str.
  - **Ví dụ:** `SELECT REPLACE('SQL Tutorial', 'SQL', 'HTML')`.
    - Kết quả: `'HTML Tutorial'`.
- **Hàm `SUBSTRING(str, start, length)`:** Trích xuất chuỗi con từ vị trí start với độ dài length trong chuỗi str.
  - **Ví dụ:** `SELECT SUBSTRING('SQL Tutorial', 1, 3)`.
    - Kết quả: `'SQL'`.

Hàm xử lý ngày, tháng, năm

- **Hàm `GETDATE()`, `CURRENT_TIMESTAMP`:** Trả về ngày tháng và thời gian hiện tại của hệ thống theo định dạng "YYYY-MM-DD hh:mm:ss.mmm".
  - **Ví dụ 1:** `SELECT GETDATE()`.
    - Kết quả: `'2023-04-30 15:30:59.917'`.
  - **Ví dụ 2:** `SELECT CURRENT_TIMESTAMP`.
    - Kết quả: `'2023-04-30 15:30:59.917'`.

Hàm xử lý ngày, tháng, năm (TT)

- **Hàm `DATEPART(datepart, date)`:** Trả về thành phần xác định datepart của một ngày date được truyền vào. Hàm này trả về kết quả là một giá trị số nguyên.
  - date: Ngày truyền vào để lấy các phần tương ứng với tham số datepart.
  - datepart: Đại diện một phần của tham số date.
    - year, yy, yyyy: Năm.
    - quarter, q, qq: Quý.
    - month, mm, m: Tháng.
    - week, ww, wk: Tuần.

Hàm xử lý ngày, tháng, năm (TT)

- hour, hh: Giờ.
- minute, mi, n: Phút.
- second, ss, s: Giây.
- ...

▶ **Ví dụ:** `SELECT DATEPART(year, '2023-04-30 15:30:59.917')`.

- Kết quả: 2023.

Hàm xử lý ngày, tháng, năm (TT)

- **Hàm `DAY(date)`:** Trả về ngày của tham số date.
  - Ví dụ: `SELECT DAY('2017/08/13 09:08')`.
    - Kết quả: 13.
- **Hàm `MONTH(date)`:** Trả về tháng của tham số date.
  - Ví dụ: `SELECT MONTH('2017/08/13 09:08')`.
    - Kết quả: 8.
- **Hàm `YEAR(date)`:** Trả về năm của tham số date.
  - Ví dụ: `SELECT YEAR('2017/08/13 09:08')`.
    - Kết quả: 2017.

Hàm kết hợp

- **Hàm `COUNT()`:** Đếm số lượng phần tử của tập hợp hoặc số lượng bản ghi được trả về bởi một câu lệnh `SELECT`.
  - **Các biến thể:**
    - `COUNT(*)`: Đếm số dòng.
    - `COUNT(<tên_thuộc_tính>)`: Đếm số giá trị khác NULL của thuộc tính.
    - `COUNT(DISTINCT <tên_thuộc_tính>)`: Đếm số giá trị khác nhau và khác NULL của thuộc tính.

Hàm kết hợp (TT)

- **Hàm MIN():** Tìm giá trị nhỏ nhất của một tập hợp hoặc một cột được trả về bởi một câu lệnh SELECT.
- **Hàm MAX():** Tìm giá trị lớn nhất của một tập hợp hoặc một cột được trả về bởi một câu lệnh SELECT.
- **Hàm AVG():** Tính giá trị trung bình của các phần tử của một tập hợp hoặc một cột được trả về bởi một câu lệnh SELECT.
- **Hàm SUM():** Tính tổng của các phần tử của một tập hợp hoặc một cột được trả về bởi một câu lệnh SELECT.

Hàm kết hợp (TT)

- **Lưu ý**
  - Các hàm `COUNT()`, `MAX()`, `MIN()` được áp dụng cho kiểu dữ liệu bất kỳ.
  - Các hàm `SUM()` và `AVG()` chỉ áp dụng cho kiểu số.
  - Sử dụng `DISTINCT` để loại bỏ các dòng trùng.
  - Ngoại trừ hàm `COUNT(*)`, giá trị NULL không được tính đến trong các hàm khác.

Một số hàm nâng cao

- **Hàm `CAST(value AS type)`, `CONVERT(type, value)`:** Chuyển đổi một giá trị `value` (thuộc bất kỳ kiểu dữ liệu nào) sang một kiểu dữ liệu khác `type`.
  - Kiểu dữ liệu mới `type` có thể là một trong các kiểu dữ liệu như: Can be one of the following: bigint, int, smallint, tinyint, bit, decimal, numeric, money, smallmoney, float, real, datetime, smalldatetime, char, varchar, text, nchar, nvarchar, ntext, binary, varbinary,…

Một số hàm nâng cao (TT)

- **Ví dụ 1:** `SELECT CAST(25.65 AS INT)`.
  - Kết quả: 25.
- **Ví dụ 2:** `SELECT CAST(25.65 AS VARCHAR)`.
  - Kết quả: `'25.65'`.
- **Ví dụ 3:** `SELECT CAST('2017-08-25' AS DATETIME)`.
  - Kết quả: `'2017-08-25 00:00:00.000'`.

Một số hàm nâng cao (TT)

- **Ví dụ 4:** `SELECT CONVERT(INT, 25.65)`.
  - Kết quả: 25.
- **Ví dụ 5:** `SELECT CONVERT(VARCHAR, 25.65)`.
  - Kết quả: `'25.65'`.
- **Ví dụ 6:** `SELECT CONVERT(DATETIME, '2017-08-25')`.
  - Kết quả: `'2017-08-25 00:00:00.000'`.

Một số hàm nâng cao (TT)

- **Hàm `IIF(condition, value_if_true, value_if_false)`:** Trả về giá trị `value_if_true` nếu điều kiện `condition` đúng, ngược lại (nếu điều kiện `condition` sai) thì trả về giá trị `value_if_false`.
  - **Ví dụ 1:** `SELECT IIF(500<1000, 'YES', 'NO')`.
    - Kết quả: `'YES'`.
  - Ví dụ 2: `SELECT IIF('hello'='bye', 'YES', 'NO')`.
    - Kết quả: `'NO'`.

CHÚC CÁC BẠN HỌC TẬP TỐT !