



**COLE.VN**  
Connecting knowledge

# Thủ tục (Procedure) – Hàm (Function)

Trình bày: Nguyễn Văn Phúc

# NỘI DUNG CHÍNH

- 1 Thủ tục (Procedure)
- 2 Hàm (Function)
- 3 Câu lệnh rẽ nhánh (if...else, case when)
- 4 Vòng lặp (for, while)
- 5 Con trỏ (cursor)
- 6 ;with cte

# Thủ tục (Procedure), Hàm (Function)

- Procedure (thủ tục), Function (Hàm) là một đoạn chương trình kịch bản (programming scripts), gồm một hoặc nhiều câu lệnh SQL (block statement) được lưu trực tiếp trong database để có thể gọi và thực thi.
  - Là tập hợp 1 hoặc nhiều câu lệnh SQL thành 1 nhóm đơn vị xử lý logic
  - Được lưu trữ trên Database Server, có thể sử dụng lại nhiều lần
  - Khi gọi lần đầu tiên thì SQL Server thực thi và lưu trữ vào bộ nhớ đệm (plan cache), những lần gọi tiếp theo SQL server sẽ sử dụng lại Plan cache nên sẽ cho tốc độ xử lý tối ưu
- Lợi ích của Stored Procedure:
  - Module hóa: Có thể gọi nhiều lần để thực thi
  - Hiệu suất: Thực thi nhanh hơn
- Nhược điểm của Stored Procedure:
  - Thiếu tính khả chuyển (Chuyển đổi giữa các RDBMS)
  - Tải Database Server
  - Hạn chế ngôn ngữ lập trình

# Thủ tục (Procedure), Hàm (Function)

## ➤ Câu lệnh tạo Store Procedure:

```
CREATE { PROCEDURE | PROC } [schema_name.]procedure_name  
[ @parameter [type_schema_name.] datatype [ = default ] [ OUT | OUTPUT | READONLY ]  
, @parameter [type_schema_name] datatype [ = default ] [ OUT | OUTPUT | READONLY ] ]  
[ WITH ENCRYPTION ]  
AS  
BEGIN  
    [declaration_section]  
    executable_section  
END;
```

```
CREATE PROCEDURE name ([parameterlist]) [options] sqlcode
```

## ➤ Tạo Function:

```
CREATE FUNCTION name ([parameterlist]) RETURNS datatype [options] sqlcode
```

# Thủ tục (Procedure), Hàm (Function)

- Thay đổi (Update) Store Procedure:

```
ALTER PROCEDURE procedure_name;
```

```
ALTER FUNCTION function_name;
```

- Xóa Store Procedure, Function:

```
DROP PROCEDURE procedure_name;
```

```
DROP FUNCTION function_name;
```

- Thực thi Store Procedure, Function:

```
EXEC procedure_name;
```

```
EXEC procedure_name @param1=value1, @param2=value2;
```

```
EXEC procedure_name value1, value2;
```

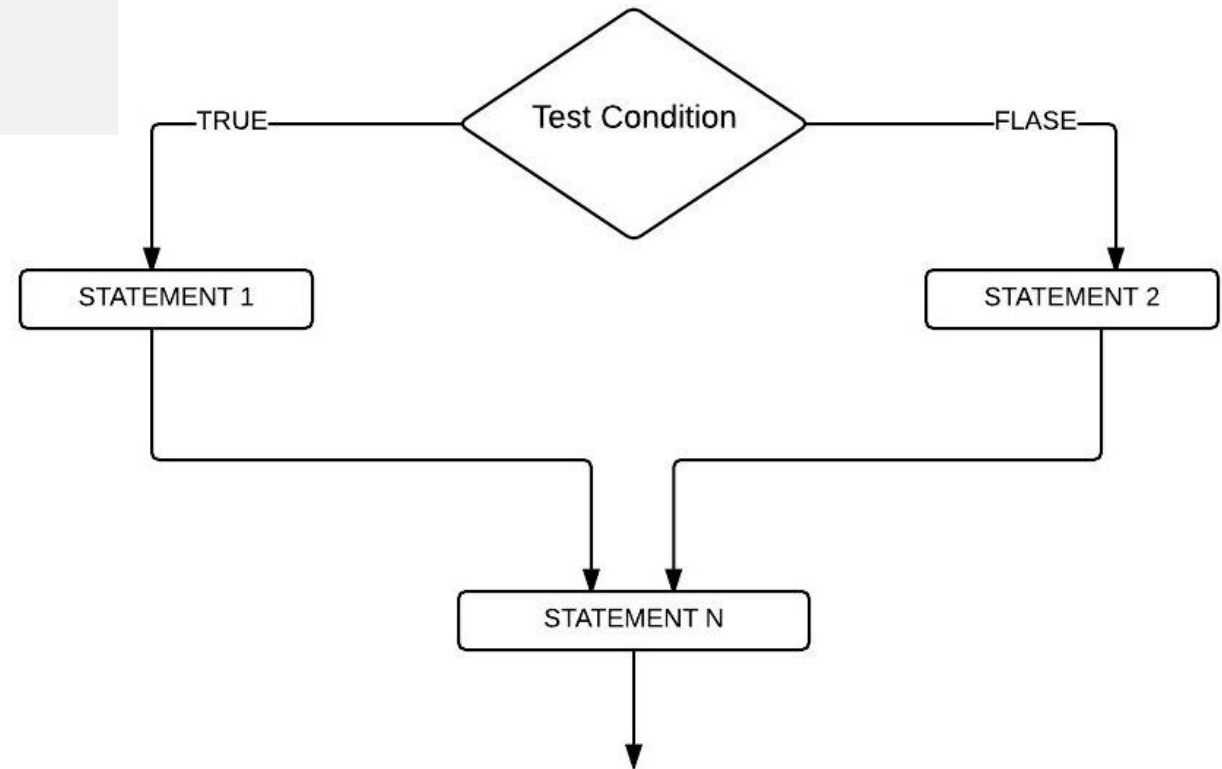
# Thủ tục (Procedure), Hàm (Function)

|                 | Thủ tục (PROCEDURE)                                     | Hàm (FUNCTION)                                    |
|-----------------|---|---|
| Cách gọi        | EXEC  | Sử dụng các câu lệnh trong sql như SELECT, UPDATE |
| Giá trị trả về  | Có thể một hoặc nhiều kết quả SELECT và các tham số out | Trả về một giá trị duy nhất thông qua RETURN      |
| Các tham số     | Tham số đầu vào (input), đầu ra (output)                | Chỉ các giá trị tham số vào (input)               |
| Gọi thủ tục/hàm | Có thể gọi các thủ tục và hàm khác                      | Chỉ có thể gọi các hàm khác                       |
|                 | Có thể sử dụng Transaction                              | Không sử dụng Transaction                         |

# Câu lệnh rẽ nhánh

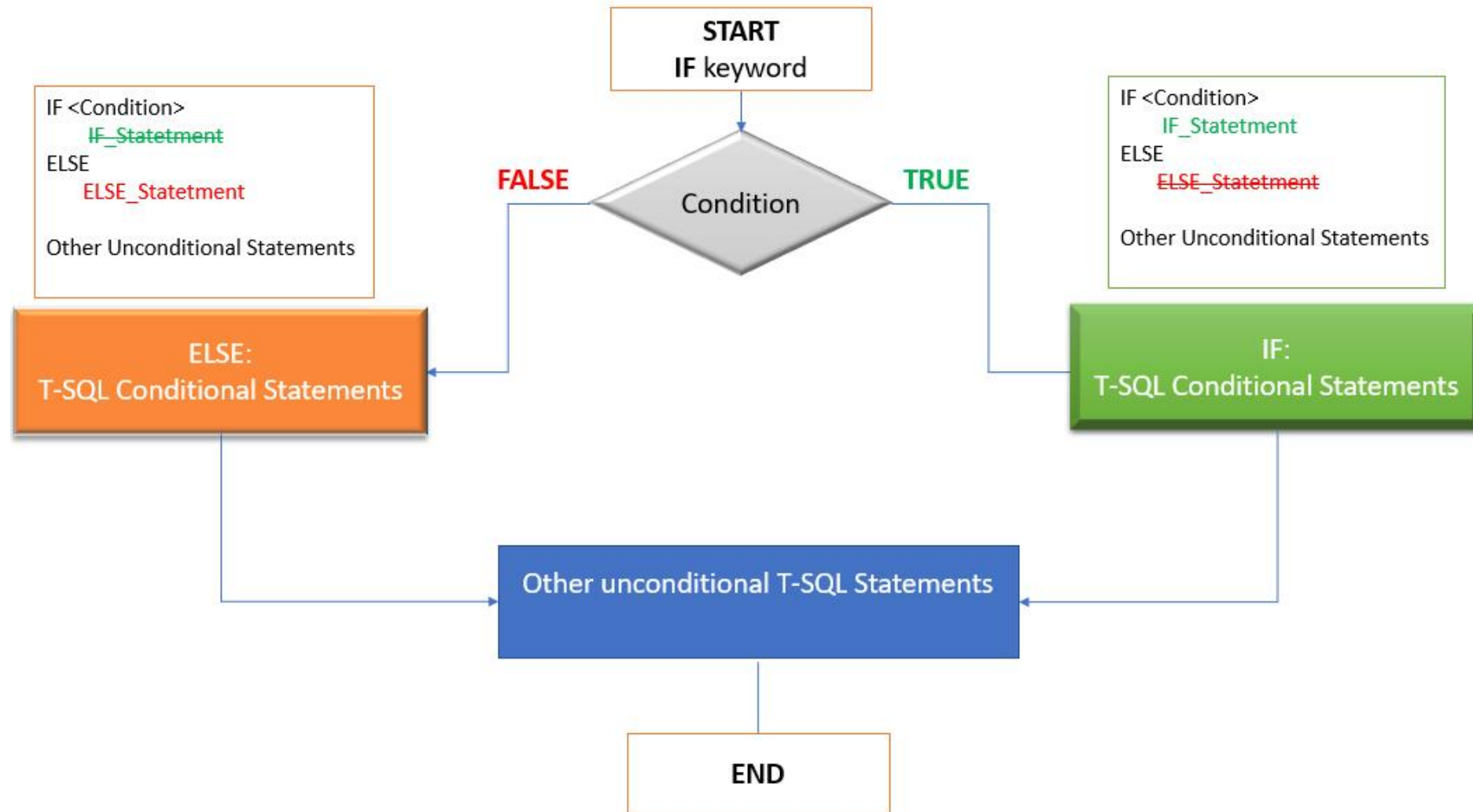
- If...else: là một câu lệnh luồng điều khiển cho phép bạn thực thi hoặc bỏ qua một khối lệnh dựa trên một điều kiện được chỉ định

```
IF boolean_expression  
BEGIN  
    { statement_block }  
END
```





# Câu lệnh rẽ nhánh



# Câu lệnh rẽ nhánh

## ➤ Case when

```
CASE  
  WHEN condition1 THEN result1  
  WHEN condition2 THEN result2  
  WHEN conditionN THEN resultN  
  ELSE result  
END;
```

```
IF boolean_expression  
BEGIN  
  { statement_block }  
END
```

# Vòng lặp WHILE

- **While:** Là một câu lệnh luồng điều khiển cho phép bạn thực hiện lặp lại một khối câu lệnh miễn là điều kiện được chỉ định là ĐÚNG.

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

- Ví dụ:

```
DECLARE @counter INT = 1;

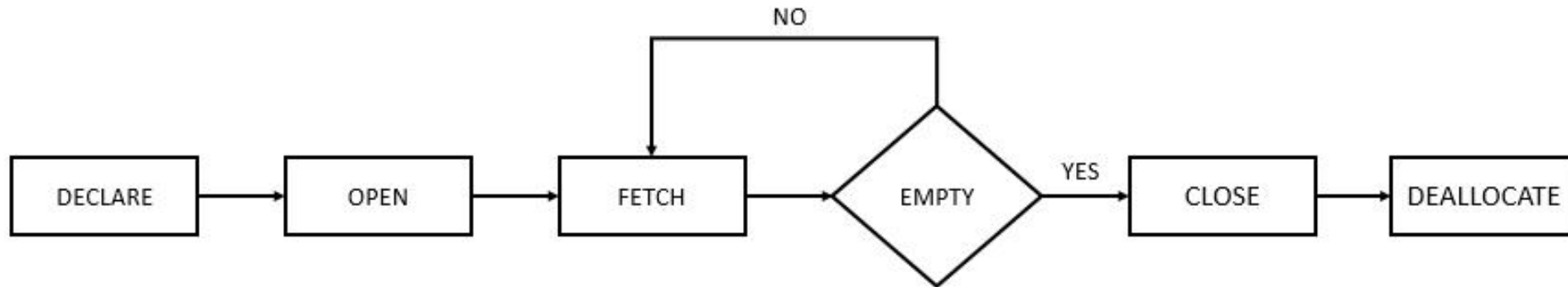
WHILE @counter <= 5
BEGIN
    PRINT @counter;
    SET @counter = @counter + 1;
END
```

# Thực hành

- Thủ tục:
  1. Lấy thông tin đầy đủ của một order
  2. Lấy thông tin chi tiết của từng dòng bán hàng order\_items
- Viết hàm:
  1. Viết hàm tính payment của 1 order item, 1 order
  2. Viết hàm tính day of week: fcn\_DOW
    - Input: ngày
    - Output: thứ trong tuần (chủ nhật là 1)

# Con trỏ - CURSOR

- Con trỏ cơ sở dữ liệu: là một đối tượng cho phép duyệt qua các hàng của tập kết quả, nó cho phép xử lý từng hàng được trả về bởi một truy vấn.
- Các bước sử dụng con trỏ:



# Con trỏ - CURSOR

➤ Câu lệnh tương ứng với các bước sử dụng con trỏ:

- **Bước 1**: Khai báo con trỏ

```
DECLARE cursor_name CURSOR  
FOR select_statement;
```

- **Bước 2**: Mở con trỏ

```
OPEN cursor_name;
```

- **Bước 3**: Tìm nạp 1 hàng từ con trỏ vào 1 hoặc nhiều biến

```
FETCH NEXT FROM cursor INTO variable_list;
```

# Con trỏ - CURSOR

## ➤ Các bước sử dụng con trỏ:

- **Bước 4**: Đóng con trỏ

```
CLOSE cursor_name;
```

- **Bước 5**: Phân bổ con trỏ

```
DEALLOCATE cursor_name;
```

## ➤ Ví dụ: Dừng con trỏ

- In tên và thông tin của tất cả các sản phẩm
- Duyệt qua các nhân viên. In tên nhân viên đó và tên quản lý tương ứng. Nếu không có quản lý ghi người đó là Quản lý cấp cao

## Bảng tạm – WITH CTE – Định nghĩa

```
with customer_cte (id, name, manager_id, manager_name)
as
(
    select
        a.staff_id
        ,a.first_name + ' ' + a.last_name
        ,b.staff_id
        ,b.first_name + ' ' + b.last_name
    from sales.staffs a
        join sales.staffs b on b.staff_id=a.manager_id
)
select * from customer_cte
```



# Bảng tạm – WITH CTE – Đề quy

- Đề quy: Cấp quản lý của nhân viên

```
with cte_staff(staff_id, full_name, level, manager_id, manager_full_name)
as
(
    select
        staff_id
        ,first_name+' '+last_name as full_name
        ,0 as level
        ,null as manager_id
        ,first_name+' '+last_name as manager_name
    from sales.staffs
    where manager_id is null
    union all
    select
        b.staff_id
        ,b.first_name+' '+b.last_name as full_name
        ,a.level+1 as level
        ,a.staff_id as manager_id
        ,a.manager_full_name as manager_full_name
    from cte_staff as a
        join sales.staffs as b on a.staff_id=b.manager_id
)
select * from cte_staff
order by level
```

# Bài tập

## ➤ Viết procedure thực hiện:

```
-- viết procedure tạo đơn hàng:
-- sales.orders:
-- order_id: tự tăng
-- customer_id: lấy ngẫu nhiên trong bảng sales.customers
-- order_status: mặc định 4
-- order_date: cộng ngẫu nhiên 0 đến 30 ngày vào ngày hiện tại
-- required_date: ngẫu nhiên trong khoảng order_date + 20 ngày
-- shipped_date: ngẫu nhiên trong khoảng required_date + 15 ngày
-- store_id: ngẫu nhiên trong bảng sales.stores
-- staff_id: ngẫu nhiên trong bảng sales.staffs

-- sales.order_items: số line random từ 3 đến 15
-- order_id: order_id đã tạo ở trên
-- item_id: tăng dần theo số lượng item trong order này
-- product_id: ngẫu nhiên trong bảng production.products
-- quantity: ngẫu nhiên trong khoảng 1 đến 12
-- list_price: giá của sản phẩm đã chọn
-- discount: ngẫu nhiên trong các giá trị: 0, 5%, 10%, 15%, 20%
```

# THANK YOU !

**COLE.VN**  
Connecting knowledge