

### 3. Biến (Variable)

#### 2. Gán giá trị cho biến

- Sử dụng câu lệnh `SET` để gán giá trị cho biến.
- Hoặc gán trực tiếp bằng `SELECT ... INTO`.

Ví dụ:

```
SET total_sales = 100;

SELECT COUNT(*) INTO total_sales
FROM orders
WHERE order_date = CURDATE();
```

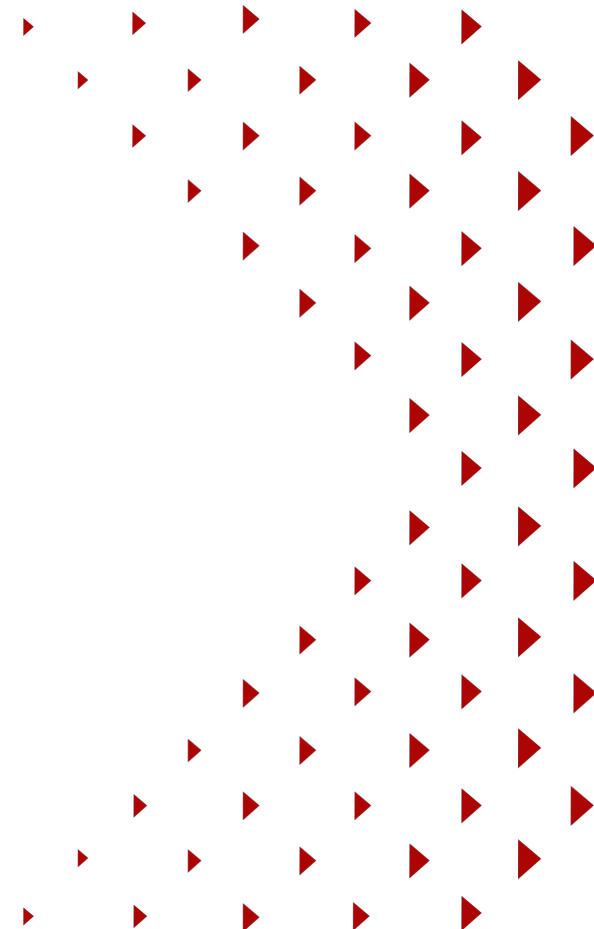


## **BÀI 6:**

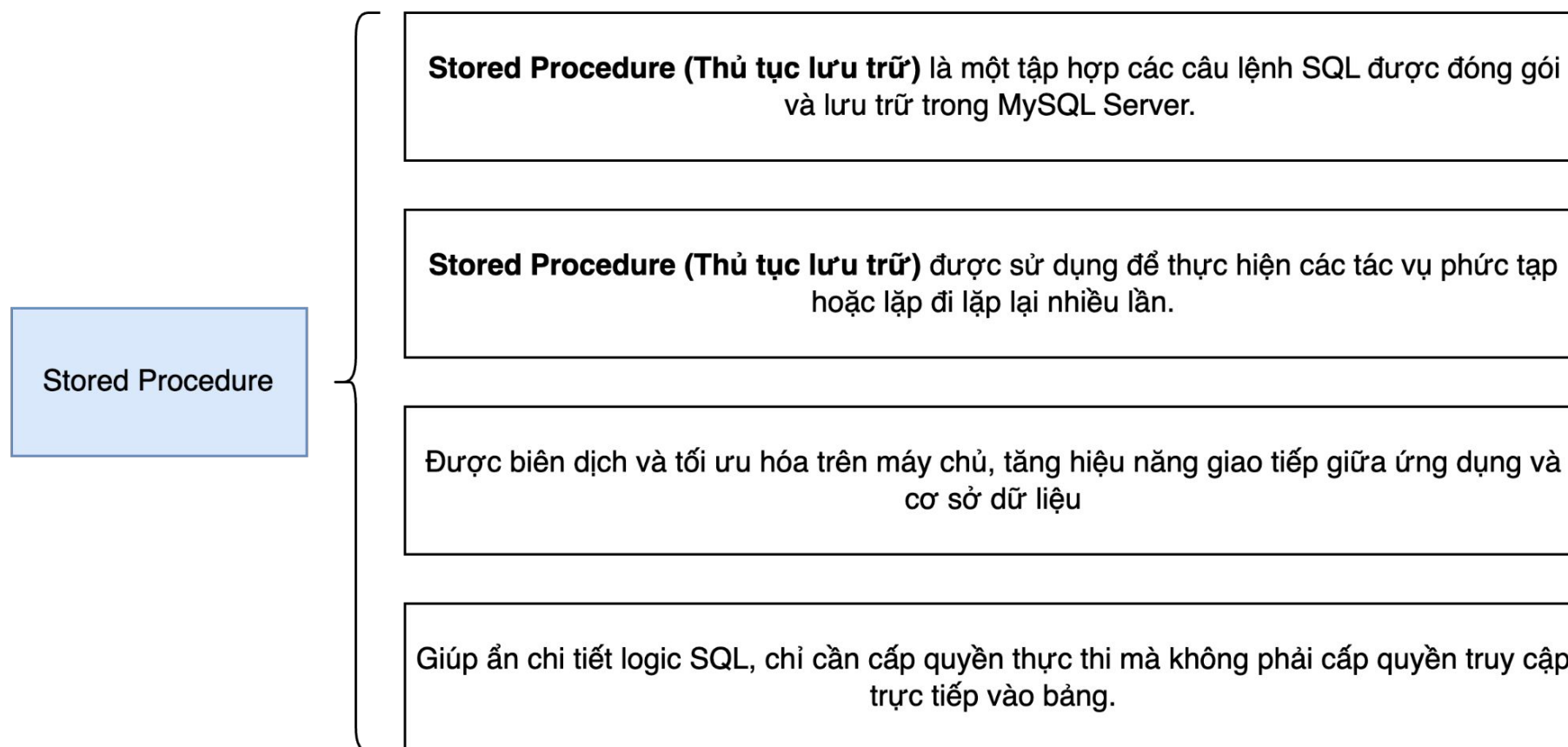
# **Procedure**

Module: Fundamental Database

Phiên bản: 1.0

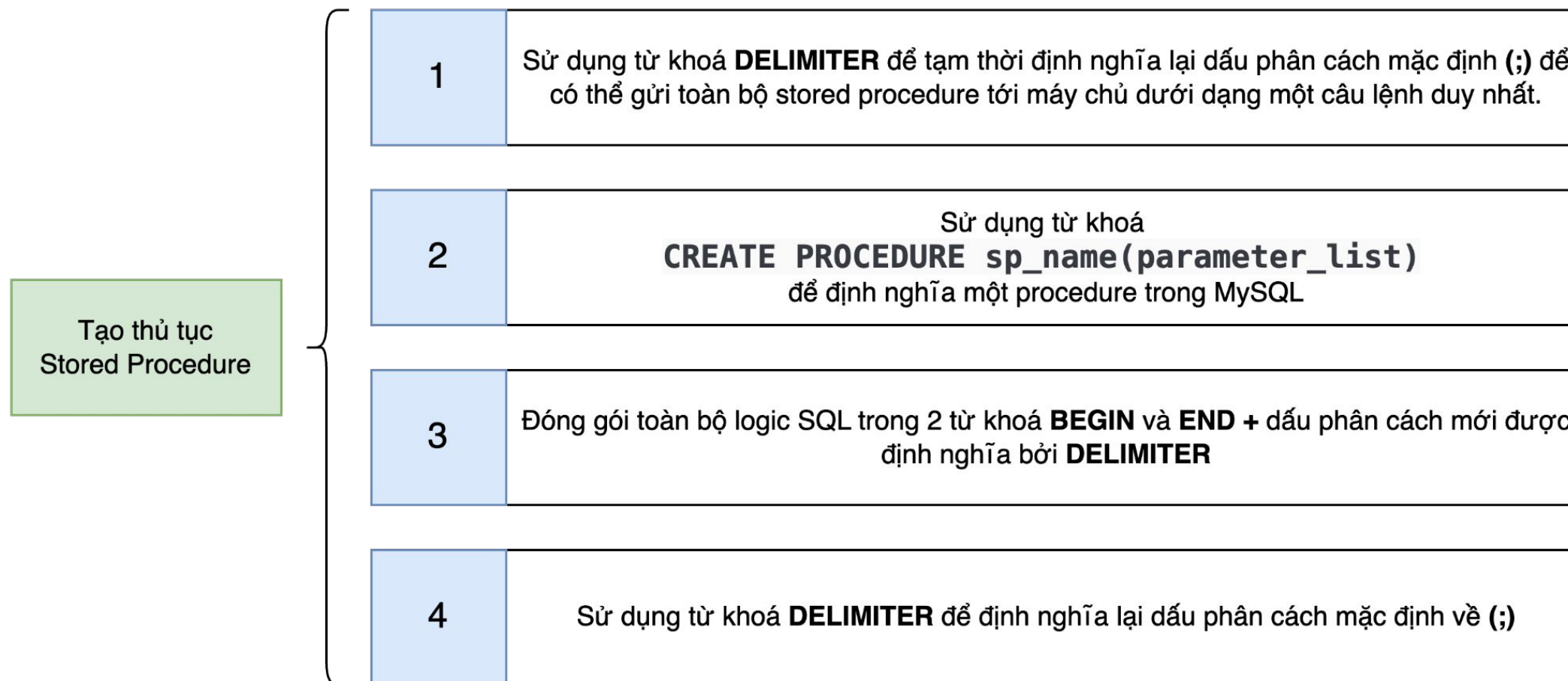


# 1. Tổng quan về procedure trong MySQL



- 1. Tổng quan về procedure trong MySQL**
- 2. Tạo và sử dụng một procedure trong MySQL**
- 3. Biến (Variable)**
- 4. Tham số IN và OUT trong Procedure**
- 5. Cấu trúc điều khiển rẽ nhánh if-else trong stored procedure**
- 6. Vòng lặp trong stored procedure**

## 2. Tạo và sử dụng một stored procedure



# 1. Tổng quan về procedure trong MySQL

```
SELECT  
    customerName,  
    city,  
    state,  
    postalCode,  
    country  
FROM  
    customers  
ORDER BY customerName;
```

## 2. Tạo và sử dụng một stored procedure

```
DELIMITER $$

CREATE PROCEDURE GetCustomers()
BEGIN
    SELECT
        customerName,
        city,
        state,
        postalCode,
        country
    FROM
        customers
    ORDER BY customerName;

END$$
DELIMITER ;
```

## 2. Tạo và sử dụng một stored procedure

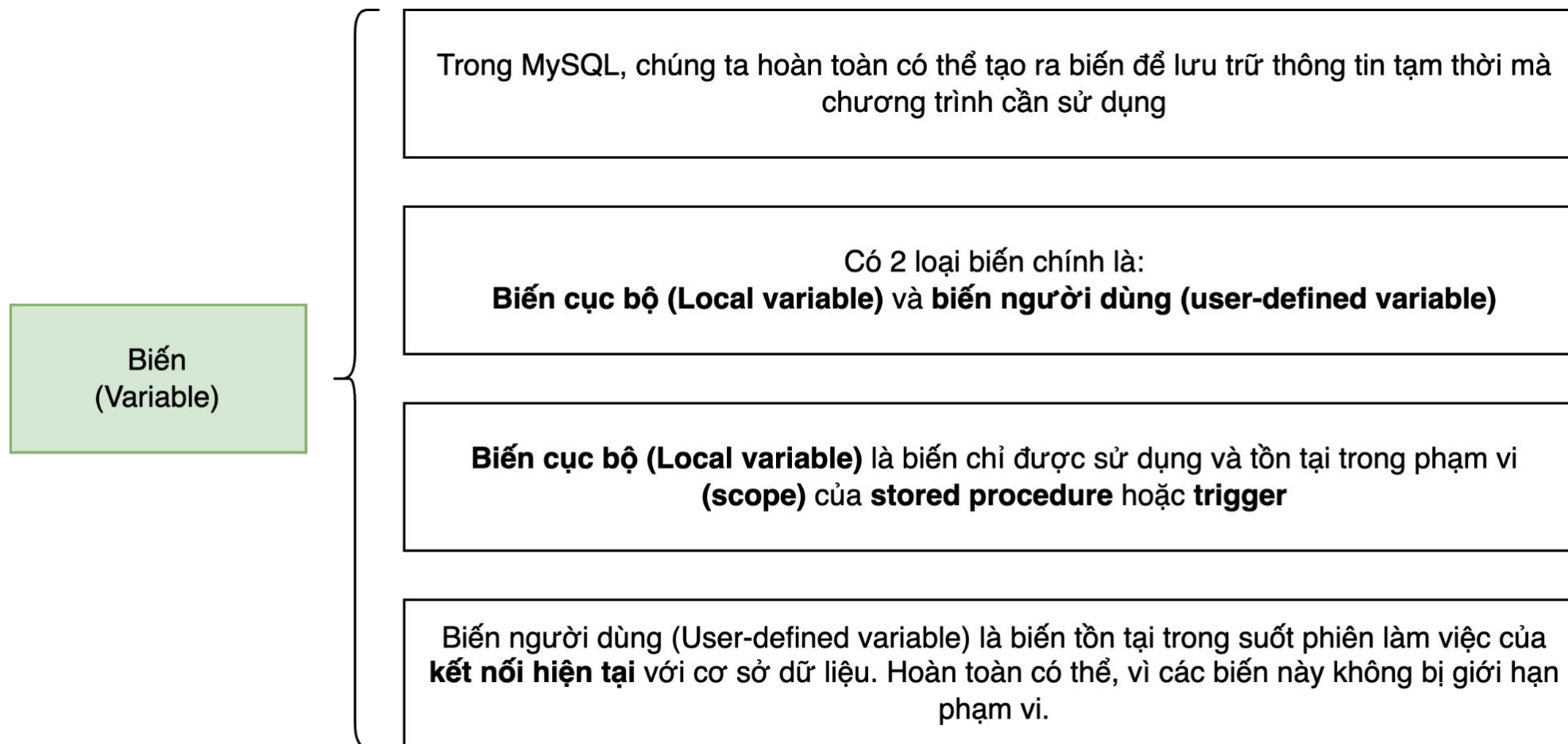
```
CALL GetCustomers();
```



## 2. Tạo và sử dụng một stored procedure

```
DROP PROCEDURE [IF EXISTS] sp_name;
```

# 3. Biến (Variable)



### 3. Biến (Variable)

```
SET @user_var = 20; -- Biến người dùng  
SELECT @user_var;   -- Truy cập được trong phiên làm việc hiện tại
```

### 3. Biến (Variable)

Kết hợp với Stored Procedure:

```
DELIMITER //  
CREATE PROCEDURE DemoUserVariable()  
BEGIN  
    SET @user_var = 30; -- Gán giá trị cho biến người dùng  
    SELECT @user_var;    -- Sử dụng biến người dùng  
END //  
DELIMITER ;
```

Gọi Stored Procedure:

```
CALL DemoUserVariable();  
SELECT @user_var; -- Biến này vẫn tồn tại sau khi stored procedure kết thúc
```

# 3. Biến (Variable)

## Biến cục bộ (Local Variable)

Trong MySQL, biến cục bộ (local variable) thường được sử dụng trong **stored procedures** hoặc **stored functions**

Đây là các biến chỉ tồn tại trong phạm vi của khối lệnh mà chúng được khai báo (chẳng hạn trong một stored procedure) và sẽ bị hủy khi quá trình thực thi của khối lệnh đó kết thúc.

Sử dụng câu lệnh DECLARE để khai báo biến cục bộ.

Biến cục bộ phải được khai báo ở đầu khối lệnh trong stored procedure hoặc stored function, ngay sau câu lệnh BEGIN. Biến phải có kiểu dữ liệu rõ ràng (như INT, VARCHAR, DATE, v.v.)

## 3. Biến (Variable)

### 1. Khai báo biến cục bộ

- Sử dụng câu lệnh `DECLARE` để khai báo biến cục bộ.
- Biến cục bộ phải được khai báo ở đầu khối lệnh trong stored procedure hoặc stored function, ngay sau câu lệnh `BEGIN`.
- Biến phải có kiểu dữ liệu rõ ràng (như INT, VARCHAR, DATE, v.v.).

```
DECLARE variable_name datatype [DEFAULT default_value];
```

Ví dụ:

```
DECLARE total_sales INT DEFAULT 0;  
DECLARE customer_name VARCHAR(100);  
DECLARE order_date DATE;
```

## 3. Biến (Variable)

### 3. Sử dụng biến cục bộ

- Biến cục bộ có thể được sử dụng trong bất kỳ phần nào của stored procedure hoặc stored function, như trong biểu thức, câu lệnh điều kiện ( `IF` ), vòng lặp ( `WHILE` ), hoặc các câu lệnh SQL khác.

## 4. Tham số IN và OUT trong procedure

Trong MySQL, **IN**, **OUT** là các loại tham số (parameters) được sử dụng trong **stored procedures** để truyền dữ liệu vào và/hoặc ra khỏi procedure.

- **IN Parameter:** Tham số chỉ dùng để truyền dữ liệu vào procedure. Đây là kiểu mặc định.
- **OUT Parameter:** Tham số chỉ dùng để truyền dữ liệu *ra khỏi* procedure. Giá trị của tham số này có thể thay đổi bên trong procedure và được trả về cho người gọi.



### 3. Biến (Variable)

Giả sử bạn có bảng `products` với cột `quantity` chứa số lượng sản phẩm trong kho, và bạn muốn tính tổng số lượng này trong một stored procedure:


```
CREATE PROCEDURE CalculateTotalQuantity()  
BEGIN  
    DECLARE total_quantity INT DEFAULT 0; -- Khai báo biến cục bộ  
  
    -- Gán giá trị tổng số lượng sản phẩm vào biến  
    SELECT SUM(quantity) INTO total_quantity  
    FROM products;  
  
    -- In ra kết quả  
    SELECT total_quantity AS TotalQuantity;  
END;
```

## 4. Tham số IN và OUT trong procedure

### 1. Cú pháp

Khi định nghĩa stored procedure, bạn khai báo loại tham số với cú pháp:

sql

 Copy code

```
CREATE PROCEDURE procedure_name(IN param1 datatype, OUT param2 datatype) BEGIN
    -- Code logic
END;
```

## 5. Cấu trúc điều khiển if-else trong stored procedure

Trong MySQL, câu lệnh `IF-ELSE` có thể được sử dụng trong các stored procedures, triggers, hoặc khi viết các câu lệnh SQL trực tiếp.

### Câu lệnh `IF-ELSE` trong Stored Procedure hoặc Trigger

- Khi sử dụng trong stored procedure hoặc trigger, `IF-ELSE` giúp chúng ta thực hiện các lệnh dựa trên điều kiện kiểm tra.

## 5. Cấu trúc điều khiển if-else trong stored procedure

```
DELIMITER $$ -- Chuyển đổi delimiter để làm việc với stored procedure
CREATE PROCEDURE CheckNumber(IN input_number INT)
BEGIN
    IF input_number > 0 THEN
        SELECT 'Number is positive';
    ELSEIF input_number < 0 THEN
        SELECT 'Number is negative';
    ELSE
        SELECT 'Number is zero';
    END IF;
END$$
DELIMITER ;
```

## 5. Cấu trúc điều khiển if-else trong stored procedure

Giải thích:

- `IF input_number > 0 THEN` : Kiểm tra nếu `input_number` lớn hơn 0.
- `ELSEIF input_number < 0 THEN` : Kiểm tra nếu `input_number` nhỏ hơn 0.
- `ELSE` : Nếu cả hai điều kiện trên đều không thỏa mãn, mặc định là `input_number` bằng 0.

## 6. Vòng lặp trong stored procedure

Trong MySQL, không có vòng lặp theo cách thức trực tiếp như trong các ngôn ngữ lập trình khác (như for, while, do-while trong C++ hoặc Java). Tuy nhiên, MySQL hỗ trợ các cơ chế lặp qua các câu lệnh trong các stored procedures hoặc triggers, thông qua các lệnh điều khiển dòng như `LOOP`, `WHILE`, `REPEAT`.

## 6. Vòng lặp trong stored procedure

### 1. Vòng lặp LOOP

- Dùng để thực hiện một chuỗi các câu lệnh lặp đi lặp lại vô hạn hoặc cho đến khi thỏa mãn điều kiện dừng.
- Cú pháp:

```
DECLARE counter INT DEFAULT 0;
LOOP_LABEL: LOOP
    -- Các câu lệnh trong vòng lặp
    SET counter = counter + 1;
    IF counter >= 5 THEN
        LEAVE LOOP_LABEL; -- Thoát khỏi vòng lặp
    END IF;
END LOOP LOOP_LABEL;
```

## 6. Vòng lặp trong stored procedure

### 2. Vòng lặp WHILE

- Vòng lặp này kiểm tra điều kiện trước khi thực hiện các câu lệnh trong vòng lặp. Nếu điều kiện ban đầu là `FALSE`, vòng lặp sẽ không chạy.
- Cú pháp:

```
DECLARE counter INT DEFAULT 0;  
WHILE counter < 5 DO  
    -- Các câu lệnh trong vòng lặp  
    SET counter = counter + 1;  
END WHILE;
```



## 6. Vòng lặp trong stored procedure

### 3. Vòng lặp REPEAT

- Vòng lặp này kiểm tra điều kiện sau khi thực hiện các câu lệnh trong vòng lặp. Vòng lặp sẽ luôn thực hiện ít nhất một lần.
- Cú pháp:

```
DECLARE counter INT DEFAULT 0;  
REPEAT  
    -- Các câu lệnh trong vòng lặp  
    SET counter = counter + 1;  
UNTIL counter >= 5  
END REPEAT;
```

- ❑ **Nắm được cách sử dụng stored procedure trong mysql**
- ❑ **Nắm được cách sử dụng biến trong mysql**
- ❑ **Nắm được cách sử dụng các tham số in và out trong stored procedure**
- ❑ **Nắm được cách sử dụng các cấu trúc điều khiển trong mysql như if-else, vòng lặp**



# KẾT THÚC

HỌC VIỆN ĐÀO TẠO LẬP TRÌNH CHẤT LƯỢNG NHẬT BẢN