STORED PROCEDURE
TRIGGER
TRANSACTIONS

STORED PROCEDURE

Stored Procedure

- Stored Procedure là một **nhóm câu lệnh SQL** được lưu trữ sẵn trong database và có thể được gọi bằng tên. Nó giống như một "hàm" trong lập trình, có thể nhận tham số đầu vào và trả về kết quả.

```
DELIMITER //
CREATE PROCEDURE tên_procedure ([IN|OUT|INOUT] tham_số kiểu_dữ_liệu, ...)
BEGIN
-- Các câu lệnh SQL
END //
DELIMITER ;
```

Stored Procedure

- Stored Procedure không có tham số

```
DELIMITER //
CREATE PROCEDURE get_all_customers()
BEGIN
   SELECT * FROM customers;
END //
DELIMITER ;

-- Gọi procedure
CALL get_all_customers();
```

Stored Procedure

- Stored Procedure có tham số IN

```
DELIMITER //
CREATE PROCEDURE get_customer_by_id(IN customer_id INT)
BEGIN
     SELECT * FROM customers WHERE id = customer_id;
END //
DELIMITER ;

-- Gọi procedure với tham số
CALL get_customer_by_id(1);
```

#	id	name	email	phone	address	created_at	updated_at
1	1	Nguyễn Văn An	newemail@email.com	0901234567	Hà Nội	2025-05-10 13:56:27	2025-05-10 14:07:00

Stored Procedure

- Stored Procedure có tham số OUT

```
DELIMITER //
CREATE PROCEDURE count_customers(OUT total_count INT)
BEGIN SELECT COUNT(*) INTO total_count FROM customers;
END // DELIMITER ;

-- Gọi procedure và lấy giá trị OUT
CALL count_customers(@count); SELECT @count;
```

#	@count
1	3

Stored Procedure

- [Northwind] Đếm số lượng đơn hàng đã bán sản phẩm có tên là @ProductName. Kết quả phải được gán cho @count

```
DELIMITER //
CREATE PROCEDURE count_order_product(IN ProductName VARCHAR(40), OUT count INT)
BEGIN
    SELECT COUNT(DISTINCT o.orderId) INTO count
    FROM SalesOrder o
    JOIN OrderDetail od ON o.orderId = od.orderId
    JOIN Product p ON od.productId = p.productId
    WHERE p.productName = ProductName;
END //
DELIMITER ;
```

Stored Procedure

- **Ưu điểm**
  - Thực thi ngay trên server, giảm network traffic
  - Có thể gọi lại nhiều lần từ nhiều ứng dụng khác nhau
  - Logic phức tạp được lưu trữ ngay trong database
  - Xử lý ngay tại mức cơ sở dữ liệu
- **Nhược điểm**
  - Không dễ theo dõi và debug
  - Khó quản lý phiên bản

Stored Procedure - Use cases

- **Xử lý giao dịch phức tạp**
  - Chuỗi hành động kiểm tra trước khi tiến hành tạo đơn hàng (order)
- **Kiểm tra quy tắc nghiệp vụ trước khi thêm/cập nhật dữ liệu**
  - Chuỗi hành động: kiểm tra role hợp lệ [‘admin', ‘staff', ‘customer']
- **Tạo báo cáo phức tạp** *
  - Báo cáo doanh thu theo nhiều tiêu chí
- **Tự động hóa và Scheduled Tasks**
  - Tự động cập nhật trạng thái đơn hàng quá hạn

TRIGGER

Trigger

- Trigger là thủ tục lưu trữ (stored procedure) đặc biệt được **tự động thực thi** khi xảy ra một sự kiện cụ thể trên bảng, ví dụ trước hoặc sau khi thực hiện INSERT, UPDATE, hoặc DELETE.
- Đặc điểm chính của Trigger:
  - Trigger được kích hoạt tự động, không cần gọi trực tiếp
  - Mỗi trigger được liên kết với một bảng cụ thể
  - Có thể chạy BEFORE hoặc AFTER một sự kiện

Trigger - Cú pháp

Từ khóa

NEW: Tham chiếu đến dữ liệu mới (INSERT, UPDATE)

OLD: Tham chiếu đến dữ liệu cũ (UPDATE, DELETE)

```
CREATE TRIGGER tên_trigger
{BEFORE | AFTER} {INSERT | UPDATE | DELETE}
ON tên_bảng FOR EACH ROW BEGIN
-- Các câu lệnh SQL
END;
```

```
-- Xem danh sách trigger
SHOW TRIGGERS;
-- Xem chi tiết một trigger
SHOW CREATE TRIGGER tên_trigger;
-- Xóa trigger
DROP TRIGGER tên_trigger;
```

Trigger

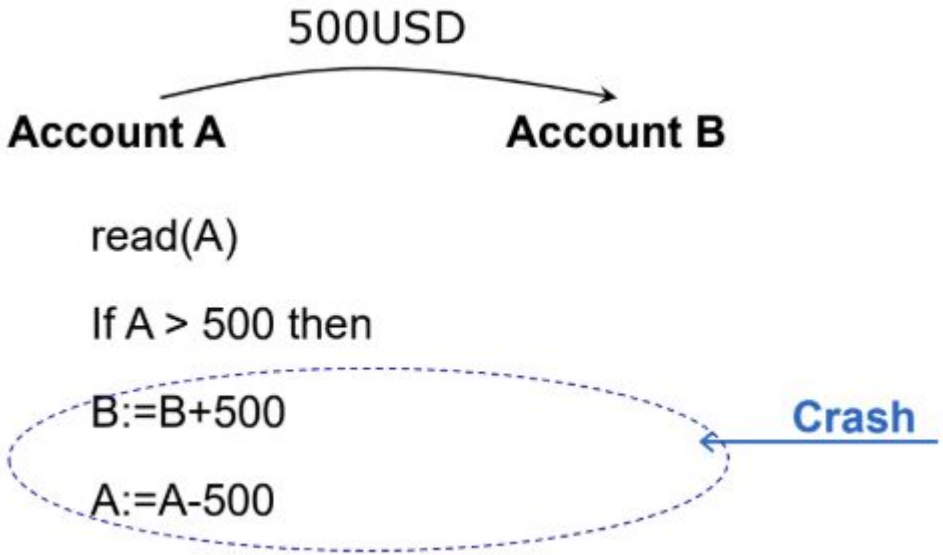
- **Ưu điểm**
  - Thực thi tự động các quy tắc nghiệp vụ
  - Kiểm soát dữ liệu trước khi lưu
  - Tính nhất quán khi đảm bảo dữ liệu tuân thủ quy tắc
  - Xử lý ngay tại mức cơ sở dữ liệu
- **Nhược điểm**
  - Không dễ theo dõi và debug
  - Có thể làm **giảm hiệu năng** các thao tác DML

Trigger - Use cases

- Kiểm tra quy tắc nghiệp vụ trước khi thêm/cập nhật dữ liệu
  - Mã sinh viên bắt đầu bằng ‘SV_’, ví dụ ‘SV_001’
- Audit logging
  - Ghi lại thay đổi khi có sự kiện INSERT/UPDATE/DELETE vào những bảng quan trọng

TRANSACTIONS

Transaction



Transaction

- Transaction là một nhóm các câu lệnh SQL được thực hiện như **một đơn vị công việc duy nhất**. Tất cả các câu lệnh trong transaction phải thành công hoặc không thành công cùng nhau.

Transaction - ACID

- **Atomicity (Tính nguyên tử)**
  - Tất cả operations trong transaction được thực hiện hoặc không được thực hiện gì cả.
- **Consistency (Tính nhất quán)**
  - Transaction đưa database từ 1 trạng thái hợp lệ đến trạng thái hợp lệ khác.
- **Isolation (Tính cô lập)**
  - Các transaction thực hiện độc lập với nhau.
- **Durability (Tính bền vững)**
  - Một khi transaction được commit, thay đổi sẽ được lưu trữ vĩnh viễn.

Transaction

- Tạo 1 transaction.

```
START TRANSACTION;
-- Các câu lệnh SQL
[COMMIT | ROLLBACK];
```

- Kết thúc 1 transaction sử dụng COMMIT, ROLLBACK
  - COMMIT **lưu** các thay đổi vào cơ sở dữ liệu
  - ROLLBACK **hủy** các thay đổi thực hiện trong giao dịch và cơ sở dữ liệu được phục hồi về trạng thái trước giao dịch
- `SAVEPOINT` định nghĩa một điểm đánh dấu trong một giao dịch.

Transaction

```
# Start transaction
begin_transaction()

# Run Queries
order = create_order(customer_id=1, total_amount=150000)
inventory_updated = update_inventory(product_id=101, quantity=2)

# If there's an error or queries don't do their job, rollback!
if not order or not inventory_updated:
    rollback_transaction(connection)
    print("Transaction failed and rolled back")
else:
    # Else commit the queries
    commit_transaction(connection)
    print("Transaction successful and committed")
```