



Seminar

# Procedure - Function - Transaction



# Nội dung

---

1

**Stored procedure**

2

**Function**

3

**Transaction**



# Stored Procedure

## SQL Server

Lập trình và sử dụng

# Stored Procedure

- Stored Procedure là thủ tục được biên dịch và lưu trữ sẵn trong CSDL

```
CREATE PROCEDURE XinChao  
    @hoTen nvarchar(50)  
AS  
    print N'Xin chào ' + @hoTen  
GO
```

- Lợi ích:
  - Tăng hiệu năng
  - Độc lập với chương trình ứng dụng
  - Giảm vấn đề nghẽn đường truyền mạng (client-server)
  - Bảo mật cơ sở dữ liệu

# Tạo Stored Procedure bằng lệnh T-SQL

**CREATE PROCEDURE** procedure\_name

Tạo stored  
procedure

@parameter1 data\_type [output],  
@parameter2 data\_type [output],...

Khai báo tham  
số (input, output)

**AS**

[khai báo các biến cho xử lý]  
{Các câu lệnh transact-sql}

**GO**

Xử lý stored  
procedure

# Cập nhật & Xóa Stored Procedure

- Cập nhật stored procedure

```
ALTER PROCEDURE  procedure_name
@parameter1 data_type      [output],
@parameter2 data_type      [output],... /*các tham số*/
AS
    [khai báo các biến cho xử lý]
    {Các câu lệnh transact-sql}
GO
```

- Xóa stored procedure

```
DROP PROCEDURE  procedure_name
DROP PROC  procedure_name
```



# Thực thi Stored Procedure

```
EXECUTE procedure_name parameter_value1,  
parameter_value2,...
```

```
EXEC procedure_name parameter_value1,  
parameter_value2,...
```

```
CREATE PROCEDURE XinChao  
    @hoTen nvarchar(50)  
AS  
    print N'Xin chào ' + @hoTen  
GO
```

```
EXEC XinChao N'Hiệp'
```

# Truyền giá trị cho Stored Procedure

Tạo stored procedure

```
CREATE PROCEDURE TEST_A
    @para_1    nvarchar(50),
    @para_2    int,
AS
...
GO
```

Truyền giá trị dựa trên vị trí parameter lúc tạo procedure

```
EXEC TEST_A N'Xin chào !!!', 123
```

Truyền giá trị dựa trên tên của parameter

```
EXEC TEST_A @para_2 = 123, @para_1 = N'Xin chào !!!'
```



# Kiểu dữ liệu của parameter

---

- Kiểu dữ liệu scalar (char, nvarchar, int, datetime, ...)
- Kiểu dữ liệu table



# Ví dụ

*--Khai báo kiểu dữ liệu mới*

```
CREATE TYPE DSCTDonHang AS TABLE
```

```
(  
    MaSP int UNIQUE,  
    DonGia float,  
    SoLuong int  
)
```

*--Ví dụ thêm dữ liệu vào bảng @temp*

```
DECLARE @temp DSCTDonHang
```

```
INSERT @temp VALUES(1,15,3)
```

```
SELECT * FROM @temp
```

# Ví dụ (tt)

```
CREATE PROC USP_THEMPDH
```

```
    @TEMP AS DSCTDATHANG readonly,
```

```
    @MADATHANG int,
```

```
    @MAKHACHHANG int
```

```
AS
```

```
BEGIN
```

```
    --Thêm phiếu đặt hàng
```

```
    INSERT PHIEUDATHANG (MADATHANG, NGAYDAT, MAKHACHHANG)
```

```
    VALUES(@MADATHANG, GETDATE(), @MAKHACHHANG)
```

```
    --Thêm chi tiết phiếu đặt hàng
```

```
    INSERT CHITIETPHIEUDAT ( MASANPHAM, DONGIA, SOLUONG, MADATHANG)
```

```
    SELECT *, @MADATHANG FROM @TEMP
```

```
END
```

# Ví dụ (tt)

*--Khai báo danh sách chi tiết đơn hàng*

```
DECLARE @TEMP DSCTDATHANG
```

*--Thêm chi tiết vào danh sách*

```
INSERT @TEMP  
VALUES(1, 10, 2),  
      (2, 20, 3)
```

*--Xem nội dung bảng @temp*

```
SELECT * FROM @TEMP
```

*--Thực thi thủ tục*

```
EXEC USP_THEMPDH @TEMP, 'DH001', 'KH1'
```

# Trả về kết quả từ Stored Procedure

---

- Trả về giá trị bằng tham số **output**
- Trả về giá trị bằng lệnh **return**
- Trả về bảng dữ liệu bằng lệnh **select**



# Trả về kết quả bằng tham số output

```
ALTER PROC Tru
    @So1    int,
    @So2    int,
    @Kq     int output
AS
    set @Kq = @So1 - @So2
GO
```

```
DECLARE @test int
EXEC Tru 1, 2, @test output
PRINT @test
```



# Trả về giá trị bằng lệnh return

```
CREATE PROC Test  
    @Lenh    int  
AS
```

```
    if (@Lenh = 1)  
        return 1
```

1

```
    if (@Lenh = 2)  
    begin  
        declare @float float  
        set @float = 2.6  
  
        return @float  
    end
```

2

```
    if (@Lenh = 3)  
    begin  
        declare @char varchar(50)  
        set @char = 'hello'  
  
        return @char  
    end
```

```
GO
```

```
declare @test float  
EXEC @test = Test 3  
print @test
```

Chỉ trả về được số  
nguyên

Báo lỗi

# Trả về bảng dữ liệu bằng lệnh select

```
CREATE PROC TestSelect
AS
    SELECT * FROM SINHVIEN

    SELECT * FROM LOAILOP
GO

EXEC TestSelect
```

```
INSERT TABLE_A (COL1, COL2)
EXEC TestSelect
```

Results

Messages

	ma	hoTen	namSinh	danT...	maLop
1	0212001	Nguyễn Vĩnh An	1984	Kinh	TH2002/01
2	0212002	Nguyễn Thanh Bình	1985	Kinh	TH2002/01
3	0212003	Nguyễn Thanh Cường	1984	Kinh	TH2002/02
4	0212004	Nguyễn Quốc Duy	1983	Kinh	TH2002/02
5	0312001	Phan Tuấn Anh	1985	Kinh	VL2003/01
6	0312002	Huỳnh Thanh Sang	1984	Kinh	VL2003/01

	ma	maKhoaHoc	maKhoa	maChuongTrinh	soThuTu
1	TH2002/01	K2002	CNTT	CQ	1
2	TH2002/02	K2002	CNTT	CQ	2
3	VL2003/01	K2003	VL	CQ	1

✓

Query executed successfully.

127.0.0.1 (9.0 RTM)

sa (51)

QLSV

00:00:00

9 rows

# Thủ tục lồng nhau

```
Create proc A  
AS  
Begin  
    -- Các lệnh  
End
```

```
Create proc B  
AS  
Begin  
    EXEC A  
    -- Các lệnh  
End
```



# Function

## SQL Server

Giới thiệu, lập trình và sử dụng 1 số loại Function cơ bản

# Function

---

- Function là hàm được biên dịch và lưu trữ sẵn trong cơ sở dữ liệu
- Phân loại function:
  - Function trả về giá trị vô hướng (Scalar Function)
  - Function trả về bảng dữ liệu



# Function trả về giá trị vô hướng

```
CREATE FUNCTION Cong  
    (@so1 int, @so2 int)  
RETURNS int  
AS  
BEGIN  
    RETURN @so1 + @so2  
END  
GO
```

```
SELECT dbo.Cong(1, 2)
```

Bắt buộc phải có  
BEGIN...END

Khi gọi Scalar  
Function phải sử  
dụng thêm tên  
schema

Có thể kết hợp  
function trong biểu  
thức



# Function đóng vai trò view

```
CREATE FUNCTION TimNhanVien  
    (@MaPhong char(5))
```

```
RETURNS table
```

```
AS
```

**ERROR** ~~BEGIN~~

```
RETURN (SELECT nv.MaNV, nv.HoTen  
        FROM NhanVien nv  
        WHERE nv.MaPhong = @MaPhong)
```

```
END
```

```
GO
```

Không có  
BEGIN...END trong  
inline table-valued  
function

```
SELECT *  
FROM TimNhanVien('PB001')
```

Có thể gọi  
function trong  
lệnh FROM

Không bắt buộc  
có tên schema

# Function không được phép

---

- Gọi stored procedure
- Thi hành insert, update, delete, ....
- Không sử dụng được TRY ... CATCH hoặc RAISERROR





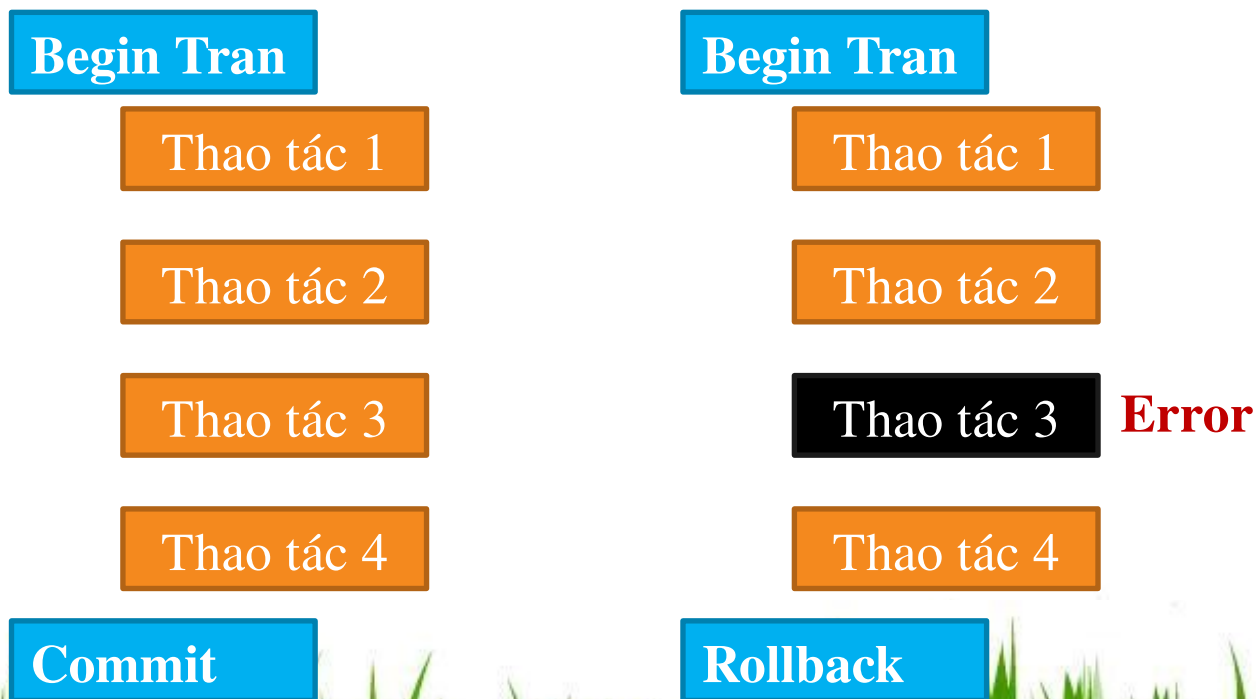
# Transaction

## SQL Server

Khái niệm và 1 số lệnh cơ bản trong giao tác

# Transaction (Giao tác)

- Là tập các thao tác có thứ tự, truy xuất dữ liệu trên CSDL. Giao tác chuyển CSDL từ trạng thái nhất quán này sang trạng thái nhất quán khác



# Tính chất giao tác - ACID

---

- Tính nguyên tố - **A**tomicity
- Tính nhất quán – **C**onsistency
- Tính cô lập – **I**solation
- Tính bền vững – **D**urability

# Tính chất giao tác - ACID

---

- - Tính nguyên tố – **Atomicity**
- Atomicity means that the transactions are an all-or-nothing entity—carrying out all the steps or none at all.
- - Tính nhất quán – **Consistency**

Consistency ensures that the data is valid both before and after the transaction. Data integrity must be maintained (foreign key references, for example), and internal data structures need to be in a valid state.



# Tính chất giao tác - ACID

---

## - Tính cô lập – Isolation

Isolation is a requirement that transactions not be dependent on other transactions that may be taking place concurrently (either at the same time or overlapping). One transaction can't see another transaction's data that is in an intermediate state, but instead sees the data as it was either before the transaction began or after the transaction completes.

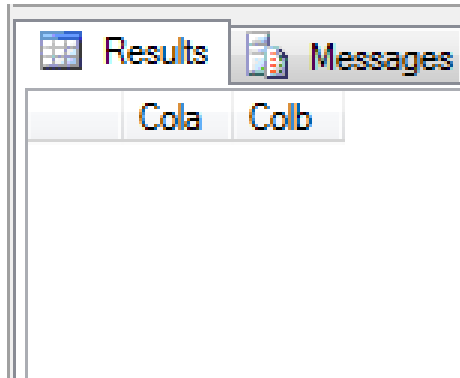
## - Tính bền vững – Durability

Durability means that the transaction's effects are fixed after the transaction has committed, and any changes will be recoverable after system failures.

# Cơ chế quản lý giao tác mặc định của SQL Server

```
CREATE TABLE TestBatch (Cola INT PRIMARY KEY, Colb CHAR(3))
GO
INSERT INTO TestBatch VALUES (1, 'aaa')
                                (2, 'bbb')
                                (3, 'ccc') /* Syntax error */
GO
SELECT * FROM TestBatch
```

Mỗi lệnh là 1 transaction



Cola	Colb
------	------

Không trả về dòng nào

# Cơ chế quản lý giao tác mặc định của SQL Server

```
INSERT INTO TestBatch VALUES (1, 'aaa')
INSERT INTO TestBatch VALUES (2, 'bbb')
INSERT INTO TestBatch VALUES (1, 'ccc') /* Duplicate key
error */
GO
SELECT * FROM TestBatch
```

	Cola	Colb
1	1	aaa
2	2	bbb

Trả về dòng 1,  
2

# Cú pháp khai báo tường minh

## Đơn giản

```
BEGIN TRANSACTION

    -- STATEMENTS GO HERE

COMMIT
```

## Có xử lý

```
BEGIN TRANSACTION

    BEGIN TRY
        -- STATEMENTS GO HERE
    END TRY
    BEGIN CATCH
        ROLLBACK TRANSACTION
    END CATCH

COMMIT
```

```
BEGIN TRANSACTION

    -- STATEMENTS GO HERE
    IF @@ERROR <> 0
        ROLLBACK TRANSACTION

COMMIT
```

**@@trancount** – biến quản lý số giao tác đang chạy

# Ví dụ

```
CREATE PROC USP_CAU1  
    @MATK CHAR(12),  
    @SODU FLOAT OUT
```

Store procedure

```
AS
```

```
BEGIN TRAN
```

Begin tran: Bắt đầu giao tác

```
    BEGIN TRY
```

```
        IF NOT EXISTS (SELECT *  
                        FROM TAIKHOAN  
                        WHERE MATK = @MATK)
```

```
        BEGIN
```

```
            PRINT @MATK + N' KHÔNG TỒN TẠI'  
            ROLLBACK TRAN
```

```
        END
```

```
        IF EXISTS (SELECT *  
                  FROM TAIKHOAN  
                  WHERE MATK = @MATK AND TINHTRANG = N'ĐÃ KHÓA')
```

```
        BEGIN
```

```
            PRINT @MATK + N' ĐÃ BỊ KHÓA'  
            ROLLBACK TRAN
```

```
        END
```

```
        SET @SODU = (SELECT SODU FROM TAIKHOAN WHERE MATK = @MATK)
```

```
    END TRY
```

```
    BEGIN CATCH
```

```
        PRINT N'LỖI HỆ THỐNG'  
        ROLLBACK TRAN
```

```
    END CATCH
```

```
COMMIT TRAN --Hoặc ROLLBACK TRAN
```

Commit : Kết thúc giao tác (thành công)

☐ Dữ liệu sẽ được xác nhận trên CSDL

Rollback: Kết thúc giao tác (thất bại)

☐ Dữ liệu sẽ được khôi phục về trạng thái bở



# Rollback

- **Lỗi do hệ thống:** Lỗi do những câu lệnh **INSERT, UPDATE, DELETE**
  - Dựa vào biến **@@error**  
[0: thành công, != 0: mã lỗi]

```
IF @@error != 0
BEGIN
-- Các câu lệnh xử lý khi bị lỗi

END
```

- **Lưu ý:** Sau mỗi câu lệnh Select, Insert, Update, Delete thì biến **@@error** chứa trạng thái (thành công/ thất bại) của việc thực thi câu lệnh.



# Rollback

- Xét store procedure: **spThemDGNguoiLon**
  - Bước 1: Xác định mã đọc giả
  - Bước 2: Insert vào bảng **DocGia**
  - Bước 3: Kiểm tra tuổi của đọc giả
  - Bước 4: Nếu không đủ tuổi thì thông báo lỗi và kết thúc
  - Bước 5: Ngược lại thì Insert vào bảng **NgLon**

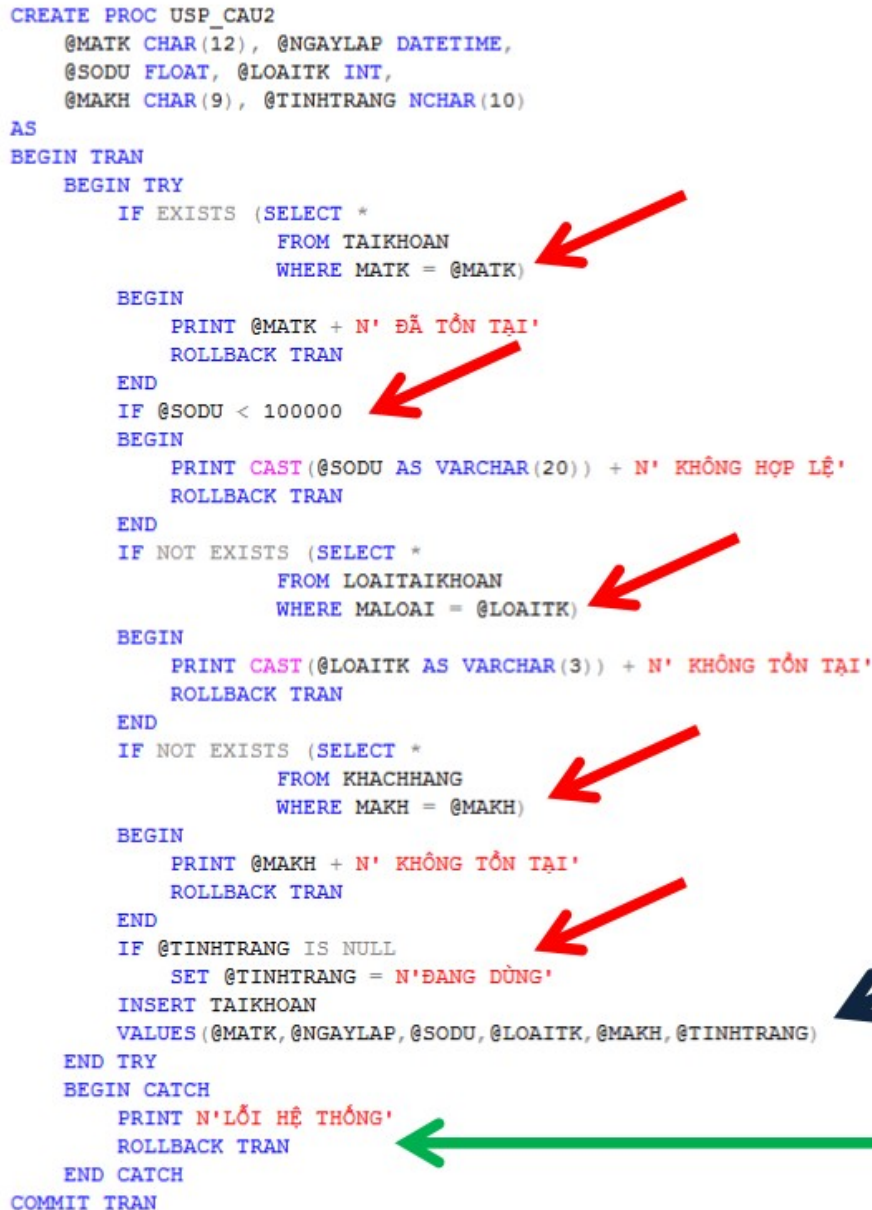
- **Lỗi do người dùng:**

- Đọc giả thêm vào nhỏ hơn 18 tuổi
- Xác định lỗi dựa vào đoạn code do người dùng viết.
- Ví dụ:

```
IF @tuoi < 18
BEGIN
-- Các câu lệnh xử lý khi bị lỗi
END
```

# Rollback

```
CREATE PROC USP_CAU2
    @MATK CHAR(12), @NGAYLAP DATETIME,
    @SODU FLOAT, @LOAITK INT,
    @MAKH CHAR(9), @TINHTRANG NCHAR(10)
AS
BEGIN TRAN
    BEGIN TRY
        IF EXISTS (SELECT *
                    FROM TAIKHOAN
                    WHERE MATK = @MATK)
        BEGIN
            PRINT @MATK + N' ĐÃ TỒN TẠI'
            ROLLBACK TRAN
        END
        IF @SODU < 100000
        BEGIN
            PRINT CAST(@SODU AS VARCHAR(20)) + N' KHÔNG HỢP LỆ'
            ROLLBACK TRAN
        END
        IF NOT EXISTS (SELECT *
                      FROM LOAITAIKHOAN
                      WHERE MALOAI = @LOAITK)
        BEGIN
            PRINT CAST(@LOAITK AS VARCHAR(3)) + N' KHÔNG TỒN TẠI'
            ROLLBACK TRAN
        END
        IF NOT EXISTS (SELECT *
                      FROM KHACHHANG
                      WHERE MAKH = @MAKH)
        BEGIN
            PRINT @MAKH + N' KHÔNG TỒN TẠI'
            ROLLBACK TRAN
        END
        IF @TINHTRANG IS NULL
            SET @TINHTRANG = N'ĐANG DÙNG'
        INSERT TAIKHOAN
        VALUES (@MATK, @NGAYLAP, @SODU, @LOAITK, @MAKH, @TINHTRANG)
    END TRY
    BEGIN CATCH
        PRINT N'LỖI HỆ THỐNG'
        ROLLBACK TRAN
    END CATCH
COMMIT TRAN
```



The diagram illustrates the execution flow of the stored procedure USP\_CAU2. Red arrows point to the following lines of code, indicating where a rollback occurs:

- Line 15: `IF EXISTS (SELECT * FROM TAIKHOAN WHERE MATK = @MATK)`
- Line 20: `ROLLBACK TRAN` (after the first existence check)
- Line 25: `IF @SODU < 100000`
- Line 30: `ROLLBACK TRAN` (after the balance check)
- Line 35: `IF NOT EXISTS (SELECT * FROM LOAITAIKHOAN WHERE MALOAI = @LOAITK)`
- Line 40: `ROLLBACK TRAN` (after the account type check)
- Line 45: `IF NOT EXISTS (SELECT * FROM KHACHHANG WHERE MAKH = @MAKH)`
- Line 50: `ROLLBACK TRAN` (after the customer check)
- Line 55: `IF @TINHTRANG IS NULL`
- Line 60: `SET @TINHTRANG = N'ĐANG DÙNG'`
- Line 65: `INSERT TAIKHOAN VALUES (@MATK, @NGAYLAP, @SODU, @LOAITK, @MAKH, @TINHTRANG)`
- Line 70: `END TRY`
- Line 75: `PRINT N'LỖI HỆ THỐNG'`
- Line 80: `ROLLBACK TRAN` (in the catch block)

A blue arrow points to the `INSERT TAIKHOAN` statement (line 65), indicating the point where the transaction is committed. A green arrow points to the `ROLLBACK TRAN` statement in the catch block (line 80), indicating the point where the transaction is rolled back.



**Thank you!**

