



# Cơ sở dữ liệu nâng cao



# CHƯƠNG IV

## **STORED PROCEDURE & GIAO DỊCH**

# Nội dung trình bày

---

1. **Stored Procedure**
2. **Giao dịch**



## 1. Stored Procedure

## 1.1 Các tùy chọn lập trình thủ tục trong Transact-SQL

Kiểu	Nhóm câu Lệnh (Batch)	Lưu trữ	Thực thi	Tham số
<b>Mã kịch bản (Script)</b>	Gồm nhiều nhóm câu lệnh	Trong file trên ổ đĩa	Từ công cụ client như Management Studio hoặc SQLCMD	Không
<b>Stored Procedure</b>	Duy nhất	Trong đối tượng của CSDL	Bởi ứng dụng hoặc trong mã kịch bản SQL	Có
<b>Hàm người dùng định nghĩa</b>	Duy nhất	Trong đối tượng của CSDL	Bởi ứng dụng hoặc trong mã kịch bản SQL	Có
<b>Trigger</b>	Duy nhất	Trong đối tượng của CSDL	Tự động bởi server khi một truy vấn hành động cụ thể xảy ra	Không

## 1.2 Stored Procedure

---

- Stored Procedure là một tập các câu lệnh T-SQL thực hiện một nhiệm vụ cụ thể, được đặt tên và lưu trữ trong CSDL dưới dạng đã biên dịch.
- Stored procedure cung cấp một phương pháp hữu ích cho việc thực thi lặp lại cùng một nhiệm vụ
  - Giúp tái sử dụng code
  - Khi thực thi lại một nhiệm vụ, sử dụng lời gọi Stored Procedure thay vì viết và thực thi lại cùng một tập hợp các câu lệnh.
- Cách sử dụng các biến, cấu trúc điều khiển trong Stored Procedure tương tự như mã kịch bản

## 1.2 Stored Procedure

- Ví dụ
  - Đoạn mã kịch bản tạo Stored Procedure spCopyInvoices, thực hiện copy dữ liệu từ bảng Invoices sang bảng Invoice Copy

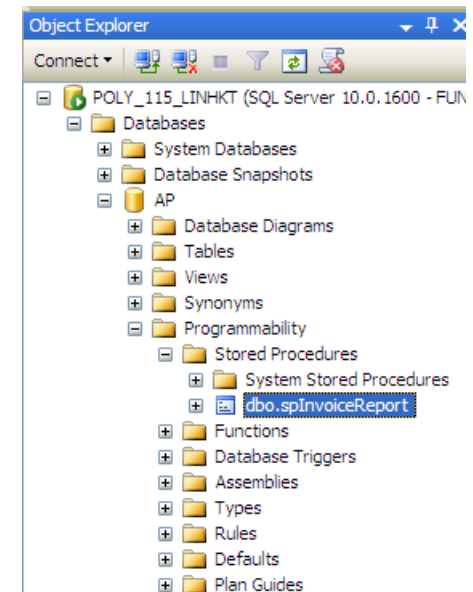
USE AP

```
IF OBJECT_ID('spCopyInvoices') IS NOT NULL
    DROP PROC spCopyInvoices
GO
```

```
CREATE PROC spCopyInvoices
AS
```

```
IF OBJECT_ID('InvoiceCopy') IS NOT NULL
    DROP TABLE InvoiceCopy
```

```
SELECT * INTO InvoiceCopy FROM Invoices
```



- Mỗi lần thực hiện copy dữ liệu, chỉ cần thực hiện lời gọi SP
- **EXEC spCopyInvoices**

## 1.3 Cú pháp

---

- Cú pháp:

**CREATE {PROC|PROCEDURE} <tên thủ tục>**

**[<Danh sách tham số>]**

**[WITH [RECOMPILE] [, ENCRYPTION] [, <mệnh đề EXECUTE AS>]]**

**AS <Các câu lệnh SQL>**

- Chú ý:
  - Tên thủ tục
    - Chứa tối đa 128 kí tự
    - Nên đặt tên với tiền tố sp
  - Câu lệnh CREATE PROC phải là câu lệnh đầu tiên của một nhóm câu lệnh



## 1.4 Khai báo và làm việc với tham số

---

- Hai loại tham số:
  - Tham số đầu vào
    - Tham số bắt buộc
      - Bắt buộc phải truyền giá trị cho tham số này
    - Tham số tùy chọn:
      - Đã được gán giá trị mặc định.
      - Nếu không truyền giá trị trong lời gọi thủ tục. Tham số sẽ nhận giá trị mặc định.
  - Tham số đầu ra
    - Sử dụng để lấy kết quả trả về từ thủ tục
    - Khai báo bằng từ khóa OUTPUT

- Cú pháp <danh sách tham số>

**@<tham số 1> <kiểu dữ liệu> [= <mặc định>] [OUTPUT|OUT]**

**[, @<tham số 2> <kiểu dữ liệu> [= <mặc định>]  
[OUTPUT|OUT]]...**

## 1.4 Khai báo và làm việc với tham số

---

### **Lưu ý:**

- Giá trị mặc định có thể NULL. Trong trường hợp này, nếu người sử dụng không cung cấp tham số, SQL Server sẽ thi hành thủ tục theo các tham số khác
- Giá trị mặc định có thể bao gồm các ký tự đại diện (% , \_ , [ ], [^] ) nếu thủ tục sử dụng tham số với từ khóa LIKE.

## 1.4 Khai báo và làm việc với tham số

- Ví dụ
  - Stored Procedure sử dụng một tham số đầu ra và hai tham số đầu vào

```
CREATE PROC splnvtotal3
    @InvTotal    money OUTPUT,
    @DateVar     smalldatetime = NULL,
    @VendorVar   varchar(40) = '%'
AS
```

Tham số đầu ra  
Tham số đầu vào bắt buộc  
Tham số đầu vào tùy chọn

-- Nếu @Date Var không được truyền giá trị  
-- Gán giá trị cho @DateVar bằng ngày hóa đơn nhỏ nhất

```
IF @DateVar IS NULL
    SELECT @DateVar = MIN(InvoiceDate) FROM Invoices
```

-- Truy xuất tổng số tiền của các hóa đơn có ngày hóa đơn lớn hơn @DateVar  
-- của nhà cung cấp có VendorName được lọc theo giá trị biến @VendorVar

```
SELECT @InvTotal = SUM(InvoiceTotal)
FROM Invoices JOIN Vendors
ON Invoices.VendorID = Vendors.VendorID
WHERE (InvoiceDate >= @DateVar) AND
      (VendorName LIKE @VendorVar)
```

## 1.4 Khai báo và làm việc với tham số

---

- Hai cách truyền giá trị cho tham số
  - Truyền theo tên
  - Truyền theo vị trí
- Lời gọi thủ tục truyền tham số theo vị trí

```
DECLARE @MyInvTotal money
EXEC spInvTotal3 @MyInvTotal OUTPUT, '2008-06-01', 'P%'
SELECT @MyInvTotal
```

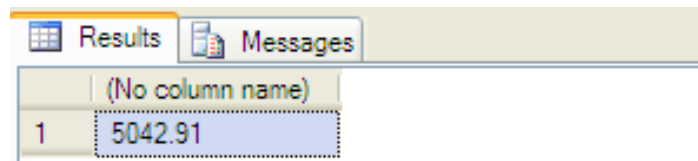
## 1.4 Khai báo và làm việc với tham số

- Lời gọi thủ tục truyền tham số theo tên

```
DECLARE    @MyInvTotal money
EXEC      splnvtot3 @DateVar = '2008-06-01', @VendorVar = 'P%',
           @InvTotal = @MyInvTotal OUTPUT
```

- Lời gọi thủ tục không truyền giá trị cho tham số tùy chọn

```
DECLARE    @MyInvTotal money
EXEC      splnvtot3 @DateVar = '2008-06-01', @InvTotal = @MyInvTotal OUTPUT
```



The screenshot shows the SQL Server Enterprise Manager interface. The 'Results' tab is active, displaying a single row of data. The first column is labeled '(No column name)' and the second column contains the value '5042.91'.

	(No column name)
1	5042.91

## 1.5 Làm việc với giá trị trả về

---

- Giá trị trả về của thủ tục
  - Sử dụng tham số OUTPUT để trả về giá trị có kiểu dữ liệu bất kì
  - Sử dụng hàm RETURN để trả về giá trị số nguyên
- Cú pháp câu lệnh RETURN

**RETURN <biểu thức số nguyên>**

## 1.5 Làm việc với giá trị trả về

---

- Ví dụ
  - Trả về số lượng hóa đơn của nhà cung cấp có VendorName thỏa mãn điều kiện lọc @VendorVar và có ngày hóa đơn lớn hơn giá trị của biến @DateVar

```
CREATE PROC    splnvcunt
    @DateVar    smalldatetime = NULL,
    @VendorVar    varchar(40) = '%'
AS

IF @DateVar IS NULL
    SELECT @DateVar = MIN(InvoiceDate) FROM Invoices

DECLARE @InvCount int

SELECT @InvCount = COUNT(InvoiceID)
    FROM Invoices JOIN Vendors
        ON Invoices.VendorID = Vendors.VendorID
WHERE ((InvoiceDate >= @DateVar) AND
    (VendorName LIKE @VendorVar))

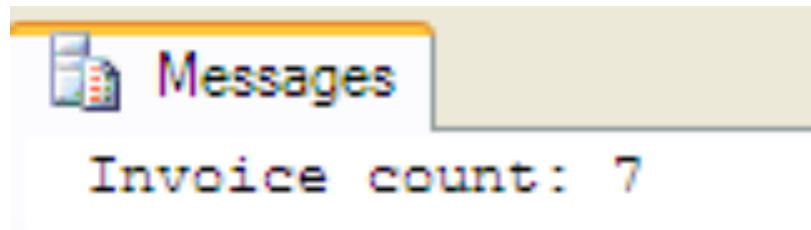
RETURN @InvCount
```

## 1.5 Làm việc với giá trị trả về

---

- Ví dụ
  - Lệnh gọi Stored Procedure

```
DECLARE @InvCount int
EXEC @InvCount = spInvCount '2008-06-01', 'P%'
PRINT 'Invoice count: ' + CONVERT(varchar, @InvCount)
```





## 1.6 Kiểm tra tính hợp lệ của dữ liệu và tạo thông báo lỗi

---

- Xử lý lỗi
  - Câu lệnh TRY ... CATCH
- Ngăn chặn lỗi
  - Kiểm tra tính hợp lệ của dữ liệu.
  - Nếu dữ liệu không hợp lệ, sử dụng câu lệnh RAISERROR để sinh lỗi.

## 1.6 Kiểm tra tính hợp lệ của dữ liệu và tạo thông báo lỗi

---

### ■ Cú pháp

**RAISERROR** ({<ID của thông báo lỗi> | <chuỗi thông báo>}, <độ nghiêm trọng>, <trạng thái>)

### ■ Chú ý:

- Câu lệnh RAISERROR được truyền tham số **<ID của thông báo lỗi>** sẽ tạo một lỗi hệ thống
- Câu lệnh RAISERROR được truyền tham số **<chuỗi thông báo>** sẽ tạo một thông báo lỗi chính là tham số được truyền vào.
- Tham số **<Độ nghiêm trọng>** chỉ ra mức độ nghiêm trọng của lỗi.
  - Giá trị từ 11 -> 19. Lỗi được tạo ra sẽ được xử lý trong khối CATCH
  - Giá trị từ 20 -> 25. Kết nối từ client tới CSDL sẽ bị ngắt đột ngột

## 1.6 Kiểm tra tính hợp lệ của dữ liệu và tạo thông báo lỗi

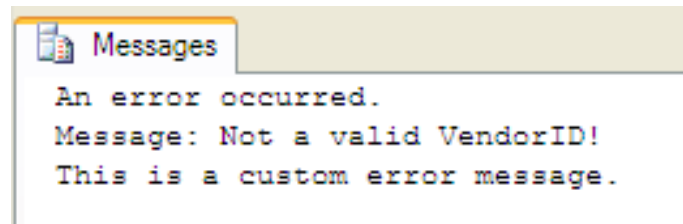
- Ví dụ
  - Chỉ thực hiện câu lệnh chèn dữ liệu vào bảng Invoices khi tham số đầu vào VendorID hợp lệ (đã tồn tại trong bảng Vendors)

```
CREATE PROC splInsertInvoice
    @VendorID int, @InvoiceNumber varchar(50),
    @InvoiceDate smalldatetime, @InvoiceTotal money,
    @TermsID int, @InvoiceDueDate smalldatetime
AS
IF EXISTS (SELECT * FROM Vendors WHERE VendorID = @VendorID)
    BEGIN
        INSERT Invoices
        VALUES (@VendorID,@InvoiceNumber,
                @InvoiceDate,@InvoiceTotal,0,0,
                @TermsID,@InvoiceDueDate,NULL)
    END
ELSE
    BEGIN
        RAISERROR('Not a valid VendorID!', 11, 1)
    END
```

## 1.6 Kiểm tra tính hợp lệ của dữ liệu và tạo thông báo lỗi

- Ví dụ
  - Mã kịch bản gọi Stored Procedure

```
BEGIN TRY
    EXEC spInsertInvoice
        799,'ZXK-799', '2008-07-01', 299.95,1, '2008-08-01'
END TRY
BEGIN CATCH
    PRINT 'An error occurred.'
    PRINT 'Message: ' + CONVERT(varchar, ERROR_MESSAGE())
    IF ERROR_NUMBER() = 50000
        PRINT 'This is a custom error message.'
END CATCH
```



## 1.7 Xem nội dung của một Stored Procedure trong CSDL

---

- Sử dụng hàm **SP\_helptext**
  - **Sp\_helptext** 'Tên Stored Procedure'
- **Chú ý:** Người dùng không thể xem nội dung của một Sp được định nghĩa sử dụng tùy chọn **ENCRYPTION**

## 1.8 Xóa/ Chỉnh sửa nội dung thủ tục

---

- Cú pháp của câu lệnh **DROP PROC**

**DROP {PROC|PROCEDURE} <tên thủ tục> [, ...]**

- Cú pháp của câu lệnh **ALTER PROC**

**ALTER {PROC|PROCEDURE} <tên thủ tục>**

**[<Danh sách tham số>]**

**[WITH [RECOMPILE] [, ENCRYPTION] [, <mệnh đề EXECUTE AS>]]**

**AS <Các câu lệnh SQL>**

## 1.8 Xóa/ Chỉnh sửa nội dung thủ tục

---

- Ví dụ
  - **Câu lệnh tạo Stored Procedure**

```
CREATE PROC spVendorState
    @State varchar(20)
AS
SELECT VendorName FROM Vendors WHERE VendorState = @State
```

- **Câu lệnh chỉnh sửa nội dung Stored Procedure**

```
ALTER PROC spVendorState
    @State varchar(20) = NULL
AS
IF @State IS NULL
    SELECT VendorName FROM Vendors
ELSE
    SELECT VendorName FROM Vendors WHERE VendorState = @State
```

- **Câu lệnh xóa thủ tục**

```
DROP PROC spVendorState
```



## 2. Giao dịch



## 2.1 Vấn đề không nhất quán khi chèn dữ liệu

- Xét dữ liệu trong hai bảng **Invoices** và **InvoiceItemLine** tương ứng với **InvoiceId = 12**

Results

Messages

	InvoiceID	VendorID	InvoiceNumber	InvoiceDate	InvoiceTotal	PaymentTotal	CreditTotal	TermsID	InvoiceDueDate	PaymentDate	InvoiceInfor
1	12	96	I77271-001	2008-04-26 00:00:00	662.00	662.00	0.00	2	2008-05-16 00:00:00	2008-05-13 00:00:00	NULL

	InvoiceID	InvoiceSequence	AccountNo	InvoiceLineItemAmount	InvoiceLineItemDescription
1	12	1	580	50.00	DiCicco's
2	12	2	570	75.60	Kinko's
3	12	3	570	58.40	Office Max
4	12	4	540	478.00	Publishers Marketing

InvoiceTotal = ∑ InvoiceLineItem

$$\text{InvoiceTotal} = \sum \text{InvoiceLineItemAmount}$$



- Câu lệnh Insert 1 và 2 thực thi thành công.
- Câu lệnh Insert 3 thực thi thất bại.

```
DECLARE @InvoiceID int
```

```
INSERT INTO Invoices  
VALUES (34, 'ZXA-080', '2008-08-30', 14092.59, 0, 0,  
3, '2008-09-30', NULL)
```

```
SET @InvoiceID = @@IDENTITY
```

```
INSERT INTO InvoiceLineItems  
VALUES (@InvoiceID, 1, 160, 4447.23, 'HW Upgrade')
```

```
INSERT INTO InvoiceLineItems  
VALUES (@InvoiceID, 2, 167, 9645.36, 'OS upgrade')
```

## 2.2 Giao dịch

---

- Để đảm bảo tính nhất quán của dữ liệu trong ví dụ trên.
  - Nếu một câu lệnh chèn trong ba câu lệnh chèn thực thi thất bại. Tất cả các câu lệnh chèn phải được hủy bỏ

=> Các câu lệnh chèn trên phải được đặt trong một giao dịch

- Giao dịch
  - Giao dịch là một nhóm thao tác cơ sở dữ liệu được kết hợp thành một đơn vị logic.

## 2.2 Giao dịch

- Ví dụ

```
DECLARE @InvoiceID int
```

```
BEGIN TRY  
    BEGIN TRAN
```

```
    INSERT Invoices  
    VALUES (34,'ZXA-080','2008-08-30',14092.59,  
            0,0,3,'2008-09-30',NULL, NULL)
```

```
    SET @InvoiceID = @@IDENTITY
```

```
    INSERT InvoiceLineItems  
    VALUES (@InvoiceID,1,160,4447.23,'HW upgrade')
```

```
    INSERT InvoiceLineItems  
    VALUES (@InvoiceID,2,167,9645.36,'OS upgrade')
```

```
    COMMIT TRAN  
END TRY
```

```
BEGIN CATCH  
    ROLLBACK TRAN  
END CATCH
```

- 3 câu lệnh **INSERT** chỉ được cập nhật thực sự vào CSDL khi giao dịch được **COMMIT**

- Khi một câu lệnh chèn xảy ra lỗi. Câu lệnh **CATCH** sẽ xử lý lỗi này và thực hiện câu lệnh **ROLLBACK TRAN** -> Toàn bộ lệnh chèn sẽ bị hủy bỏ.

## 2.3 Các trường hợp sử dụng giao dịch

---

- Nên sử dụng giao dịch trong các trường hợp sau:
  - Khi viết mã hai hay nhiều truy vấn thao tác tác động tới các dữ liệu có liên kết
  - Khi cập nhật tham chiếu khóa ngoại.
  - Khi chuyển hàng từ bảng này sang bảng khác.
  - Khi bạn viết truy vấn SELECT trước một truy vấn thao tác, và giá trị được chèn vào từ truy vấn thao tác này lại dựa trên kết quả của truy vấn SELECT.
  - Khi sự thất bại của tập câu lệnh SQL nào đó sẽ vi phạm tính toàn vẹn dữ liệu.

## 2.4 Các câu lệnh xử lý giao dịch

Câu lệnh	Mô tả
<b>BEGIN {TRAN   TRANSACTION}</b>	Đánh dấu điểm bắt đầu giao dịch.
<b>SAVE {TRAN   TRANSACTION} save_point</b>	Thiết lập điểm lưu trữ mới bên trong giao dịch.
<b>COMMIT [TRAN   TRANSACTION]</b>	Đánh dấu điểm kết thúc giao dịch và thực hiện thay đổi trong giao dịch, đồng thời thay đổi vĩnh viễn trên CSDL
<b>ROLLBACK [[TRAN   TRANSACTION] [save_point]]</b>	Roll-back (Quay lui) giao dịch tới điểm bắt đầu giao dịch hoặc đến điểm lưu trữ xác định.

- Chú ý:
  - Có thể bỏ từ khóa TRAN khi viết câu lệnh COMMIT hoặc ROLLBACK. Tuy nhiên nên viết mã có từ khóa này
  - Nếu không khai báo điểm bắt đầu cho giao dịch.
    - SQL Server sẽ xử lý mỗi câu lệnh là một giao dịch và tự động COMMIT
    - Nếu câu lệnh gây lỗi sẽ tự động ROLLBACK

## 2.4 Các câu lệnh xử lý giao dịch

- Ví dụ
  - Kiểm tra trước khi Commit giao dịch

**BEGIN TRAN**

**DELETE Invoices**

**WHERE VendorID = 34**

-- Nếu số hàng bị xóa bởi câu lệnh DELETE lớn hơn 1. Thực hiện

-- quay lui, không thực hiện lệnh xóa. Chỉ xóa khi số hàng bị ảnh hưởng bởi

-- lệnh DELETE nhỏ hơn 1

**IF @@ROWCOUNT > 1**

**BEGIN**

**ROLLBACK TRAN**

**PRINT 'Deletions rolled back.'**

**END**

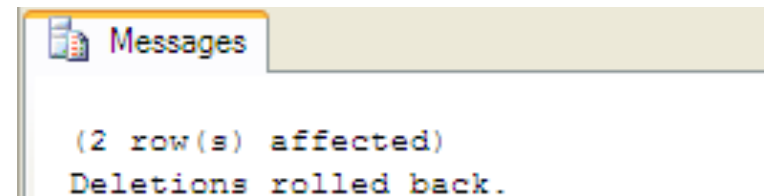
**ELSE**

**BEGIN**

**COMMIT TRAN**

**PRINT 'Deletions committed to the database.'**

**END**



## 2.5 Giao dịch lồng

---

- Giao dịch lồng (nested transaction) là giao dịch được viết bên trong một giao dịch khác.
- Mỗi khi câu lệnh BEGIN TRAN được thực thi, hàm hệ thống @@TRANCOUNT được tăng thêm 1.
- Khi thực thi câu lệnh COMMIT TRAN
  - Nếu @@TRANCOUNT > 1, các thay đổi sẽ không được commit. Thay vào đó @@TRANCOUNT giảm đi 1.
  - Nếu @@TRANCOUNT = 1, mọi thay đổi đã được thực hiện trên CSDL trong suốt giao dịch sẽ được commit và @@TRANCOUNT được gán bằng 0.
- Câu lệnh ROLLBACK TRAN roll-back toàn bộ các giao dịch đang hoạt động và thiết lập giá trị cho @@TRANCOUNT về 0.



## 2.5 Giao dịch lồng

- Ví dụ

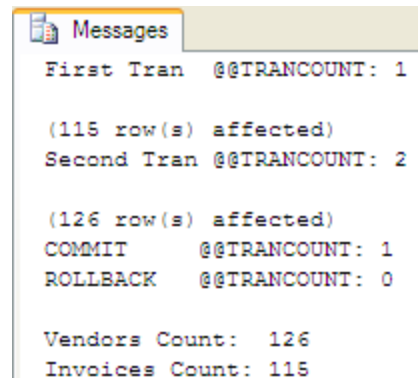
```
BEGIN TRAN -- Bắt đầu giao dịch 1
PRINT 'First Tran @@TRANCOUNT: ' + CONVERT(varchar, @@TRANCOUNT)
DELETE Invoices
BEGIN TRAN -- Bắt đầu giao dịch 2
    PRINT 'Second Tran @@TRANCOUNT: ' + CONVERT(varchar, @@TRANCOUNT)
    DELETE Vendors
COMMIT TRAN          -- Câu lệnh COMMIT này làm giảm giá trị @@TRANCOUNT.
                     -- Không COMMIT câu lệnh 'DELETE Vendors'.
PRINT 'COMMIT        @@TRANCOUNT: ' + CONVERT(varchar, @@TRANCOUNT)
ROLLBACK TRAN

PRINT 'ROLLBACK      @@TRANCOUNT: ' + CONVERT(varchar, @@TRANCOUNT)
PRINT ' '

DECLARE @VendorsCount int, @InvoicesCount int

SELECT @VendorsCount = COUNT (*) FROM Vendors
SELECT @InvoicesCount = COUNT (*) FROM Invoices

PRINT 'Vendors Count: ' + CONVERT (varchar , @VendorsCount)
PRINT 'Invoices Count: ' + CONVERT (varchar , @InvoicesCount)
```



The screenshot shows the 'Messages' window in SQL Server. It displays the output of the T-SQL script, including the transaction count for each transaction, the number of rows affected by the DELETE statements, and the final counts for Vendors and Invoices after the COMMIT and ROLLBACK operations.

```
Messages
First Tran  @@TRANCOUNT: 1

(115 row(s) affected)
Second Tran @@TRANCOUNT: 2

(126 row(s) affected)
COMMIT      @@TRANCOUNT: 1
ROLLBACK    @@TRANCOUNT: 0

Vendors Count: 126
Invoices Count: 115
```



## 2.6 Điểm lưu trữ (Save Point)

---

- Câu lệnh ROLLBACK TRAN có thể quay lui giao dịch tới điểm bắt đầu giao dịch hoặc đến điểm lưu trữ xác định.
- Để câu lệnh ROLLBACK TRAN quay lui giao dịch đến điểm lưu trữ xác định, thực hiện như sau
  - Tạo điểm lưu trữ sử dụng câu lệnh SAVE TRAN
  - Viết câu lệnh ROLLBACK TRAN kèm theo tên điểm lưu trữ
- Nếu câu lệnh ROLLBACK TRAN không đi kèm tên điểm lưu trữ. Câu lệnh này sẽ quay lui toàn bộ giao dịch

## 2.6 Điểm lưu trữ (Save Point)

- Ví dụ

```
IF OBJECT_ID ( 'tempdb..#VendorCopy' ) IS NOT NULL  
DROP TABLE tempdb.. #VendorCopy
```

```
SELECT VendorID, VendorName INTO #VendorCopy FROM Vendors WHERE VendorID < 5
```

```
BEGIN TRAN
```

```
DELETE #VendorCopy WHERE VendorID = 1
```

```
SAVE TRAN Vendor1 -- Điểm lưu trữ 1
```

```
DELETE #VendorCopy WHERE VendorID = 2
```

```
SAVE TRAN Vendor2 -- Điểm lưu trữ 2
```

```
DELETE #VendorCopy WHERE VendorID = 3
```

```
SELECT * FROM #VendorCopy
```

```
ROLLBACK TRAN Vendor2 -- Quay lui về điểm lưu trữ 2
```

```
SELECT * FROM #VendorCopy
```

```
ROLLBACK TRAN Vendor1 -- Quay lui về điểm lưu trữ 1
```

```
SELECT * FROM #VendorCopy
```

```
COMMIT TRAN
```

```
SELECT * FROM #VendorCopy
```

Results		Messages	
	VendorID	VendorName	
1	4	Jobtrak	
	VendorID	VendorName	
1	3	Register of Copyrights	
2	4	Jobtrak	
	VendorID	VendorName	
1	2	National Information Data Ctr	
2	3	Register of Copyrights	
3	4	Jobtrak	
	VendorID	VendorName	
1	2	National Information Data Ctr	
2	3	Register of Copyrights	
3	4	Jobtrak	

# Tổng kết

---

- Các nội dung đã học trong bài
  - Stored Procedure (SP)
    - Là một tập các câu lệnh T-SQL thực hiện một nhiệm vụ cụ thể, được đặt tên và lưu trữ trong CSDL dưới dạng đã biên dịch.
    - Khi thực thi lại một nhiệm vụ, sử dụng lời gọi Stored Procedure thay vì viết và thực thi lại cùng một tập hợp các câu lệnh.

- Cú pháp

**CREATE {PROC|PROCEDURE} <tên thủ tục>**

**[<Danh sách tham số>]**

**[WITH [RECOMPILE] [, ENCRYPTION] [, <mệnh đề EXCECUTE AS>]]**

**AS <Các câu lệnh SQL>**

# Tổng kết

---

- Hai loại tham số
  - Tham số đầu vào
    - Tham số đầu vào bắt buộc
    - Tham số đầu vào tùy chọn
  - Tham số đầu ra: Trả về giá trị cho đối tượng gọi SP
- Hai cách trả về giá trị cho một SP
  - Sử dụng tham số đầu ra
  - Sử dụng câu lệnh RETURN
- Gọi Stored Procedure: Sử dụng câu lệnh EXEC
  - Hai cách truyền tham số trong lời gọi
    - Truyền theo vị trí
    - Truyền theo tên

# Tổng kết

---

- Giao dịch
  - Giao dịch là một nhóm thao tác cơ sở dữ liệu được kết hợp thành một đơn vị logic.
  - Chỉ khi giao dịch được COMMIT, các câu lệnh trong giao dịch mới thực sự thay đổi CSDL
  - Khi một câu lệnh trong một giao dịch gây lỗi. Giao dịch sẽ được quay lui (ROLLBACK) về điểm bắt đầu giao dịch hoặc về điểm lưu trữ giao dịch

# **HẾT CHƯƠNG IV**

***Thank you for listening!***