Buổi 9
Stored Procedure và Trigger

Trong khuôn khổ tài liệu buổi 9, chúng ta sử dụng CSDL WebBanHang:

- File *[WebBanHang] Mo ta CSDL.png*: Lược đồ CSDL mô tả cấu trúc bảng và khóa ngoại.
- File *[WebBanHang] CSDL.sql*: Phát sinh CSDL.

I. Stored Procedure

1. Khái niệm

Stored procedure là tập hợp các câu lệnh SQL thực thi một yêu cầu, tác vụ nào đó, được lưu trữ trong CSDL dưới dạng 1 thủ tục (*procedure*). Có thể xem stored procedure giống như hàm (*function*) kiểu `void` trong ngôn ngữ lập trình.

Việc cài đặt stored procedure cho CSDL giúp cho người quản trị CSDL cũng như lập trình viên tiết kiệm được thời gian và công sức.

2. Tạo Stored Procedure

Cú pháp tạo 1 stored procedure trong SQL Server:

```
CREATE PROCEDURE <tên procedure>
AS
BEGIN
      <danh sách câu lệnh>
END;
```

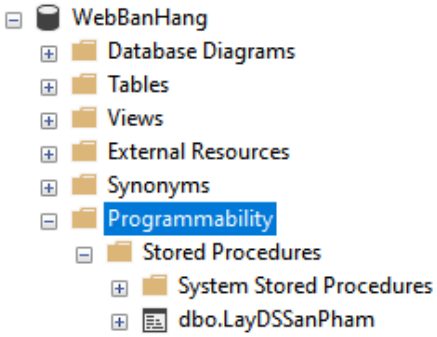
Trong đó:

- `<tên procedure>` là tên đặt cho stored procedure, tuân theo quy tắc đặt tên trong SQL.
- Từ khóa `BEGIN` và `END` có thể lược bỏ.

Ví dụ: Tạo stored procedure liệt kê danh sách các sản phẩm:

```
CREATE PROCEDURE LayDSSanPham
AS
BEGIN
      SELECT *
      FROM SanPham
END;
```

Sau khi thực thi đoạn lệnh trên, stored procedure sẽ được tạo ra và lưu trong mục *Programmability* → *Stored Procedures* của CSDL hiện tại:



Để thực thi stored procedure đã tạo, chúng ta dùng 1 trong 2 câu lệnh sau:

```
EXECUTE <tên procedure>;
EXEC <tên procedure>;
```

Ví dụ:

```
EXEC LayDSSanPham;
```

3. Tham số cho Stored Procedure

Tương tự với hàm trong ngôn ngữ lập trình, stored procedure cũng có thể nhận tham số (*parameter*) để thực thi trong câu truy vấn.

a) Khai báo tham số

Cú pháp khai báo tham số cho stored procedure:

```
CREATE PROCEDURE <tên procedure> (<danh sách tham số>)
AS
BEGIN
      <danh sách câu lệnh>
END;
```

Trong đó:

- `<danh sách tham số>` được liệt kê theo từng cặp `@<tên tham số> <kiểu dữ liệu>`, phân cách với nhau bằng dấu phẩy.
- `<kiểu dữ liệu>` là các kiểu dữ liệu cơ bản của SQL.
- Cặp dấu ngoặc `()` của danh sách tham số có thể lược bỏ.

Ví dụ: Tạo stored procedure liệt kê danh sách các sản phẩm có giá nằm trong một khoảng giá trị nào đó:

```
CREATE PROCEDURE LayDSSanPham (@gia_min int, @gia_max int)
AS
BEGIN
      SELECT *
      FROM SanPham
      WHERE GiaTien >= @gia_min AND GiaTien <= @gia_max
END;
```

b) Thực thi stored procedure kèm theo tham số

Có 2 cách thực thi stored procedure kèm theo tham số:

- Cách 1: Truyền giá trị cho tham số theo đúng thứ tự:

```
EXEC LayDSSanPham 40000, 50000;
```

- Cách 2: Truyền giá trị cho tham số một cách tường minh, không cần theo đúng thứ tự:

```
EXEC LayDSSanPham @gia_max = 50000, @gia_min = 40000;
```

c) Gán giá trị mặc định cho tham số

Đôi khi chúng ta có nhu cầu thực thi stored procedure với không đầy đủ tham số, ví dụ lấy danh sách sản phẩm có giá tối thiểu 40.000 đồng (không có giá tối đa). Do đó, chúng ta có thể gán giá trị mặc định cho tham số ngay khi tạo stored procedure:

Ví dụ:

```
CREATE PROCEDURE LayDSSanPham (@gia_min int = 0, @gia_max int = 999999999)
AS
BEGIN
      SELECT *
      FROM SanPham
      WHERE GiaTien >= @gia_min AND GiaTien <= @gia_max
END;
```

Như vậy, khi thực thi stored procedure, tham số nào không được truyền giá trị sẽ được lấy giá trị mặc định.

Ví dụ:

- Liệt kê tất cả sản phẩm:

```
EXEC LayDSSanPham
```

- Liệt kê các sản phẩm có giá từ 40.000:

```
EXEC LayDSSanPham 40000
```

- Liệt kê các sản phẩm có giá tối đa 50.000:

```
EXEC LayDSSanPham @gia_max = 50000
```

- Liệt kê các sản phẩm có giá từ 40.000 đến 50.000:

```
EXEC LayDSSanPham 40000, 50000
```

4. Khai báo biến

Stored procedure là tập hợp các câu lệnh cần để thực hiện một tác vụ, do đó, có thể trong quá trình thực thi stored procedure, chúng ta cần khai báo biến để lưu trữ dữ liệu tạm thời.

Cú pháp khai báo biến:

```
DECLARE @<tên biến> <kiểu dữ liệu>;
```

Trong đó:

- `@<tên biến>` tuân theo quy tắc đặt tên trong SQL.
- `<kiểu dữ liệu>` là các kiểu dữ liệu cơ bản của SQL.

Ví dụ:

```
DECLARE @gia_avg float;
DECLARE @sl_min int, @sl_max int;
```

Cú pháp gán giá trị cho biến:

```
SET @<tên biến> = <giá trị>;
```

Lưu ý: Biến không được gán giá trị sẽ mang giá trị mặc định là `NULL`.

Biến sau khi đã được khai báo và gán giá trị có thể được sử dụng trong các câu lệnh của stored procedure tương tự như tham số, vì bản chất tham số cũng là biến.

Ví dụ: Tạo stored procedure liệt kê danh sách sản phẩm có giá lớn hơn giá trung bình của tất cả sản phẩm:

```
CREATE PROCEDURE LayDSSanPhamLonHonTB
AS
BEGIN
	DECLARE @gia_avg float;

	SET @gia_avg = (
		SELECT AVG(GiaTien)
		FROM SanPham
	)

	SELECT *
	FROM SanPham
	WHERE GiaTien > @gia_avg
END;
```

5. Sửa và xóa Stored Procedure

Cú pháp để sửa stored procedure:

```
ALTER PROCEDURE <tên procedure> (<danh sách tham số>)
AS
BEGIN
      <danh sách câu lệnh>
END;
```

Trong đó:

- `<tên procedure>` là tên của stored procedure muốn sửa.
- `<danh sách tham số>` và `<danh sách tham số>` có thể được cài đặt mới hoàn toàn.

Để xóa stored procedure, click chuột phải vào tên của stored procedure đó và chọn *Delete*. Ngoài ra, có thể xóa bằng câu lệnh SQL.

Cú pháp để xóa stored procedure:

```
DROP PROCEDURE <tên procedure>
```

hoặc

```
DROP PROC <tên procedure>
```

II. Trigger

1. Khái niệm

Trigger có thể xem là 1 stored procedure đặc biệt, sẽ tự động được gọi khi có 1 sự kiện xảy ra trong CSDL. Các sự kiện thường gặp là `INSERT`, `UPDATE` và `DELETE`. Khi các sự kiện này xảy ra, dữ liệu trong CSDL sẽ bị thay đổi, do đó, cần phải đảm bảo tính đúng đắn và tính toàn vẹn của dữ liệu bằng cách tạo ra các trigger để thực hiện các thao tác này một cách tự động.

Ví dụ:

- Sau khi hóa đơn được lập, cần giảm số lượng tồn kho của sản phẩm mà khách hàng mua
- Sau khi hóa đơn bị hủy, cần trả lại số lượng tồn kho của sản phẩm vừa bị hủy.
- Sau khi hóa đơn mua thêm 1 sản phẩm, cần cập nhật lại tổng tiền của hóa đơn.
- …

2. Tạo trigger

Cú pháp tạo 1 trigger trong SQL Server:

```
CREATE TRIGGER <tên trigger>
ON <tên bảng>
AFTER  [INSERT | UPDATE | DELETE]
AS
BEGIN
      <danh sách câu lệnh>
END;
```

Trong đó:

- `<tên trigger>` là tên đặt cho trigger[1], tuân theo quy tắc đặt tên trong SQL.
- Mệnh đề `ON` quy định trigger sẽ thuộc về bảng nào trong CSDL. Khi có sự kiện tác động tới bảng này, trigger sẽ được gọi.
- Mệnh đề `AFTER` quy định trigger sẽ được gọi sau sự kiện nào (`INSERT`, `UPDATE`, `DELETE`).

[1] Tên của trigger thường bắt đầu bằng TRG_

Với các sự kiện tác động tới dữ liệu trong bảng bằng 3 câu lệnh INSERT, UPDATE, DELETE, SQL cung cấp các bảng phụ phát sinh sau đây:

- Câu lệnh INSERT sẽ phát sinh bảng inserted chứa các dòng vừa được thêm vào bảng.
- Câu lệnh DELETE sẽ phát sinh bảng deleted chứa các dòng vừa bị xóa khỏi bảng.
- Câu lệnh UPDATE xem như là xóa dòng cũ và thêm dòng mới nên phát sinh cả 2 bảng inserted và deleted.

Ví dụ: Tạo trigger tự động giảm số lượng tồn kho cho sản phẩm khi thêm dòng vào bảng CTHoaDon:

```
CREATE TRIGGER TRG_GiamSLTonKho
ON CTHoaDon
AFTER INSERT
AS
BEGIN
	UPDATE SanPham
	SET SoLuongTonKho = SoLuongTonKho - (
		SELECT SUM(SoLuong)
		FROM inserted
		WHERE MaSP = SanPham.MaSP
	)
	FROM SanPham INNER JOIN inserted ON SanPham.MaSP = inserted.MaSP
END;
```

Ví dụ: Tạo trigger tự động trả lại số lượng tồn kho cho sản phẩm khi xóa dòng trong bảng CTHoaDon:

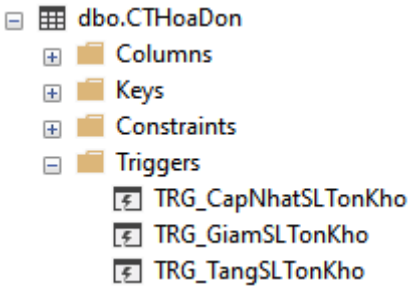
```
CREATE TRIGGER TRG_TangSLTonKho
ON CTHoaDon
AFTER DELETE
AS
BEGIN
	UPDATE SanPham
	SET SoLuongTonKho = SoLuongTonKho + (
		SELECT SUM(SoLuong)
		FROM deleted
		WHERE MaSP = SanPham.MaSP
	)
	FROM SanPham INNER JOIN deleted ON SanPham.MaSP = deleted.MaSP
END;
```

Ví dụ: Tạo trigger tự động cập nhật số lượng tồn kho cho sản phẩm khi cập nhật số lượng trong bảng CTHoaDon:

```
CREATE TRIGGER TRG_CapNhatSLTonKho
ON CTHoaDon
AFTER UPDATE
AS
BEGIN
	UPDATE SanPham
	SET SoLuongTonKho = SoLuongTonKho + (
		SELECT SUM(SoLuong)
		FROM deleted
```

```
            WHERE MaSP = SanPham.MaSP
        ) - (
            SELECT SUM(SoLuong)
            FROM inserted
            WHERE MaSP = SanPham.MaSP
        )
        FROM
            SanPham INNER JOIN deleted ON SanPham.MaSP = deleted.MaSP
                    INNER JOIN inserted ON SanPham.MaSP = inserted.MaSP
END;
```

Sau khi thực thi đoạn lệnh trên, trigger sẽ được tạo ra và lưu trong mục *Triggers* của bảng đã chọn (quy định tại mệnh đề ON).



3. Trigger Instead of

Trigger INSTEAD OF là 1 dạng trigger đặc biệt cho phép bỏ qua câu lệnh INSERT, UPDATE hoặc DELETE đang tác động tới 1 bảng, thay vào đó thực hiện các câu lệnh đã được cài đặt trong trigger.

Cú pháp tạo trigger INSTEAD OF tương tự như trigger bình thường, tuy nhiên mệnh đề AFTER được đổi thành mệnh đề INSTEAD OF:

```
CREATE TRIGGER <tên trigger>
ON <tên bảng>
INSTEAD OF  [INSERT | UPDATE | DELETE]
AS
BEGIN
    <danh sách câu lệnh>
END;
```

Ví dụ: Khi người dùng muốn xóa 1 sản phẩm, thay vì xóa hẳn bằng câu lệnh DELETE thì cập nhật lại trạng thái cho sản phẩm đó để tránh bị lỗi ràng buộc khóa chính:

```
CREATE TRIGGER TRG_CapNhatTrangThaiSP
ON SanPham
INSTEAD OF DELETE
AS
BEGIN
    UPDATE SanPham
    SET TrangThai = 0
    FROM SanPham INNER JOIN deleted ON SanPham.MaSP = deleted.MaSP
END;
```

III. Bài tập

1. Tạo stored procedure cho phép liệt kê danh sách sản phẩm theo 1 hoặc một số tiêu chí sau:
    - Giá tiền nằm trong một khoảng.
    - Tên sản phẩm (tìm gần đúng).
    - Thông tin sản phẩm (tìm gần đúng).

- Số lượng tồn kho không dưới một giá trị nào đó.
- Thuộc một loại sản phẩm nào đó (tìm theo tên loại sản phẩm).

2. Tạo trigger cho phép cập nhật tổng tiền của hóa đơn khi thêm, xóa, sửa 1 dòng trong bảng CTHoaDon.
3. Tạo trigger cho tất cả các bảng có thuộc tính TrangThai để thay vì xóa 1 dòng bằng câu lệnh DELETE thì cập nhật lại trạng thái cho dòng đó.
4. Tạo trigger cho phép cập nhật số lượng tồn kho của sản phẩm khi thêm, xóa, sửa 1 dòng trong bảng CTHoaDon