Thủ tục lưu trữ hệ thống và kích hoạt
Stored Procedure & Trigger

Nguyễn Hồng Phương
Email: phuong.nguyenhong@hust.edu.vn
Site: http://users.soict.hust.edu.vn/phuongnh
Bộ môn Hệ thống thông tin
Viện Công nghệ thông tin và Truyền thông
Đại học Bách Khoa Hà Nội

Nội dung

1. Stored Procedure
   1.1. Khái niệm
   1.2. Cú pháp
2. Trigger
   2.1. Khái niệm
   2.2. Cú pháp

https://freetuts.net/hoc-sql-server/sql-server-nang-cao

https://quantrimang.com/procedure-thu-tuc-trong-sql-server-159768

1. Stored Procedure

1.1. Khái niệm

1.2. Cú pháp

1.1. Khái niệm stored procedure

- Thủ tục lưu trữ (stored procedure - SP) là một tập hợp các phát biểu T-SQL mà SQL Server biên dịch thành một kế hoạch thực thi, được lưu trữ trong CSDL của SQL Server.
- Khi chạy một SP lần đầu, mô hình truy vấn của SP được đặt vào trong bộ nhớ.
- Các SP là các tập lệnh chạy nhanh.

1.1. Khái niệm stored procedure (tiếp)

- Có 3 loại SP:
  - SP hệ thống: do SQL Server cung cấp, có tên bắt đầu bằng tiền tố "sp_", được dùng để quản lý SQL Server và hiển thị các thông tin về CSDL và người dùng
  - SP mở rộng: là những thư viện liên kết động (DLL), được viết bằng các ngôn ngữ như C, C++,..., mà SQL Server có thể nạp và thực thi
    - SP bên ngoài: có tên bắt đầu bằng "xp_"
    - SP người dùng định nghĩa

1.1. Khái niệm stored procedure (tiếp)

- Lợi ích của việc sử dụng SP:
  - Sau khi được thực hiện, lược đồ của thủ tục được lưu trữ trong vùng đệm của thủ tục => trong cùng phiên làm việc, nếu sử dụng nhiều lần thủ tục này thì hiệu quả sẽ cao hơn.
  - Đóng gói các quy tắc nghiệp vụ. Sau khi chúng được đóng gói, nhiều ứng dụng có thể sử dụng quy tắc này. Nếu chức năng thay đổi, ta chỉ việc thay đổi ở một nơi.
  - Có thể truyền đối số vào và nhận dữ liệu trả về.

1.1. Khái niệm stored procedure (tiếp)

- Có thể được thiết lập chạy tự động khi SQL Server khởi động.
- Được gọi một cách rõ ràng. Không giống như trigger, các SP phải được gọi bởi ứng dụng, kịch bản, batch, tác vụ của bạn.
- Hữu dụng trong quản trị và bảo trì CSDL.
- Có thể gán quyền cho một người sử dụng chạy một SP cho dù người đó không có quyền trên các bảng cơ sở.

1.2. Cú pháp tạo SP

- Có thể dùng câu lệnh T-SQL, Enterprise Manager hoặc wizard để tạo SP.
- Cú pháp trong SQL Server:

```
CREATE  PROC[EDURE] procedure_name
   {;number}
[{@parameter data_type}[=default |
   NULL][VARYING][OUT PUT]]
[WITH {RECOMPILE | ENCRYPTION |
   RECOMPILE,ENCRYPTION}]
[FOR REPLICATION]
AS sql_statement
```

Ví dụ

```
USE MyDB
GO
IF EXISTS(SELECT name FROM sysobjects
WHERE name='pAuthors' AND type='P')
DROP PROCEDURE pAuthors
GO
CREATE PROCEDURE pAuthors
AS SELECT au_fname, au_lname
FROM authors
ORDER BY au_fname DESC
GO
```

- Chạy thủ tục này:
  - EXEC pAuthors

- Xem nội dung SP
  - EXEC sp_helptext pAuthors
- Xóa thủ tục:
  - DROP PROCEDURE procedure_name

Tạo một nhóm các thủ tục

```
CREATE PROC group_sp;1
AS SELECT * FROM authors
GO
CREATE PROC group_sp;2
AS SELECT au_lname FROM authors
GO
CREATE PROC group_sp;3
AS SELECT DISTINCT city FROM authors
GO
```

- Để nhận được danh sách tất cả các thành phố mà các tác giả sống, sử dụng câu lệnh sau:
  - EXEC group_sp;3

Các tham số

- @parameter data_type [=default | NULL] [VARYING] [OUTPUT]
- @parameter: tên của tham số bên trong thủ tục, có thể khai báo đến 1024 tham số bên trong một SP.
- data_type: bất kỳ kiểu dữ liệu nào do hệ thống định nghĩa hoặc do người dùng định nghĩa, ngoại trừ kiểu dữ liệu hình ảnh.
- Default: chỉ rõ giá trị mặc định cho tham số.
- VARYING: áp dụng cho recordset được trả về.
- OUTPUT: xác định đây là một tham số trả về.

Ví dụ:

- Viết một thủ tục lưu trữ nhận 5 thông số, tính giá trị trung bình và kết xuất ra:

```
CREATE PROCEDURE scores
@score1 smallint,
@score2 smallint,
@score3 smallint,
@score4 smallint,
@score5 smallint,
@myAvg smallint OUTPUT
AS SELECT @myAvg = (@score1 + @score2 +
@score3 + @score4 + @score5) / 5
```

Truyền/nhận giá trị cho/từ tham số

- Truyền đúng theo trật tự vị trí

```
DECLARE @AvgScore smallint
EXEC scores 10, 9, 8, 8, 10, @AvgScore OUTPUT
SELECT 'The Average Score is: ',@AvgScore
Go
```

- Truyền không theo trật tự

```
DECLARE @AvgScore smallint
EXEC scores
@score1=10, @score3=9, @score2=8, @score4=8,
@score5=10, @myAvg = @AvgScore OUTPUT
SELECT 'The Average Score is: ',@AvgScore
Go
```

Truyền/nhận giá trị cho/từ tham số (tiếp)

- Từ khóa RETURN

```
CREATE PROC MyReturn
@t1 smallint, @t2 smallint, @retval smallint
AS SELECT @retval = @t1 + @t2
RETURN @retval
```

- Chạy thủ tục trên:

```
DECLARE @myReturnValue smallint
EXEC @myReturnValue = MyReturn 9, 9, 0
SELECT 'The return value is: ',@myReturnValue
```

- Tùy chọn WITH RECOMPILE
  - trong phát biểu CREATE PROCEDURE: Toàn bộ thủ tục được biên dịch lại mỗi khi nó chạy, thủ tục có thể được tối ưu cho các tham số mới.
  - trong phát biểu EXEC PROCEDURE: Biên dịch thủ tục lưu trữ riêng cho lần thực thi đó và lưu trữ kế hoạch mới trong vùng đệm của thủ tục cho các lệnh EXEC PROCEDURE sau này
- Nếu một SP được tạo với tùy chọn ENCRYPTION =>không xem được nội dung của nó

2. Trigger

2.1. Khái niệm

2.2. Cú pháp

2.1 Khái niệm

- Là một thủ tục lưu trữ hệ thống (stored procedure) đặc biệt, được thực thi một cách tự động khi có sự kiện gây biến đổi dữ liệu như Update, Insert hay Delete
- Được dùng để đảm bảo toàn vẹn dữ liệu hay thực hiện các quy tắc nghiệp vụ nào đó.
- Khi nào sử dụng trigger?
  - khi các biện pháp đảm bảo toàn vẹn dữ liệu khác như Constraint không thể thỏa mãn yêu cầu của ứng dụng

2.1 Khái niệm (tiếp)

- Constraint thuộc loại toàn vẹn dữ liệu khai báo: kiểm tra dữ liệu trước khi cho phép nhận vào bảng
- Trigger thuộc loại toàn vẹn dữ liệu thủ tục nên việc Insert, Update, Delete xảy ra rồi mới kích hoạt trigger.
  - Đôi khi, do nhu cầu thay đổi dây chuyền, có thể sử dụng trigger
- Đặc điểm của trigger
  - một trigger có thể làm nhiều công việc, có thể được kích hoạt bởi nhiều sự kiện

2.1 Khái niệm (tiếp)

- trigger không thể được tạo ra trên bảng tạm hoặc bảng hệ thống
- trigger chỉ có thể được kích hoạt tự động bởi các sự kiện mà không thể chạy thủ công được.
- có thể áp dụng trigger cho view
- khi trigger được kích hoạt
  - dữ liệu mới được insert sẽ được chứa trong bảng "inserted"
  - dữ liệu mới được delete sẽ được chứa trong bảng "deleted"
  - đây là hai bảng tạm nằm trên bộ nhớ, và chỉ có giá trị bên trong trigger

2.2. Cú pháp

- Có thể dùng T-SQL hoặc Enterprise Manager để tạo trigger
- Không được dùng các phát biểu sau trong định nghĩa trigger: ALTER DATABASE, CREATE DATABASE, DISK INIT, DISK RESIZE, DROP DATABASE, LOAD DATABASE, LOAD LOG, RECONFIGURE, RESTORE DATABASE, RESTORE LOG

- Bảng tạm: deleted và inserted
  - được tham khảo như bảng thật nhưng được lưu trong bộ nhớ trong chứ không phải trên đĩa.
  - giá trị trong bảng này chỉ được truy xuất trong trigger. Ngay khi trigger hoàn tất thì các bảng này cũng không thể truy xuất được nữa.

Ví dụ:

- Tạo trigger In_ThemNCC trên bảng NHA_CC: in thông báo bất cứ khi nào dữ liệu được thêm vào bảng

```
USE MyDB
GO
IF EXISTS(SELECT name FROM sysobjects
WHERE name='In_ThemNCC' AND Type='TR')
DROP TRIGGER In_ThemNCC
GO
CREATE TRIGGER In_ThemNCC
ON NHA_CC
FOR INSERT
AS
PRINT 'BANG NHA_CC da duoc them du lieu'
GO
```

Tạo deleted trigger

- Tạo bảng tbl_MatHangXoa để lưu mặt hàng bị xóa khỏi bảng MatHang (đã có)

```
CREATE TABLE tbl_MatHangXoa(
MaHang Int NOT NULL,
TenHang Nvarchar(50) NOT NULL,
MoTa Nvarchar(100),
SoLuong SmallIn Default 0
)
```

- Tạo deleted trigger trên bảng MatHang cho sự kiện delete

```
CREATE TRIGGER tg_MatHangXoa
ON MatHang
FOR DELETE
AS
INSERT INTO tbl_MatHangXoa SELECT * FROM deleted
```

Tạo inserted trigger

- Tạo inserted trigger cho bảng BanHang. Mỗi khi có một dòng mới được thêm vào (hàng được bán) thì trigger sẽ làm nhiệm vụ trừ đi số lượng trong bảng MatHang

```
CREATE TRIGGER tg_CapNhatSoLuong
ON BanHang
FOR INSERT
AS
SELECT *
FROM inserted
UPDATE MatHang
SET MatHang.SoLuong = MatHang.SoLuong – inserted.SoLuong
WHERE MatHang.MaHang = inserted.MaHang
```

Tạo update trigger

```
CREATE TRIGGER tg_KiemTraCapNhatGia
ON MatHang
FOR UPDATE
AS
DECLARE @gia_cu smallmoney, @gia_moi smallmoney
SELECT @gia_cu = DonGia FROM deleted
PRINT 'Gia cu ='
PRINT CONVERT(varchar(6), @gia_cu)
SELECT @gia_moi = DonGia FROM inserted
PRINT 'Gia moi ='
PRINT CONVERT(varchar(6), @gia_moi)
IF(@gia_moi > (@gia_cu*1.10))
BEGIN
        PRINT 'Gia moi tang qua 10%, khong cap nhat'
        ROLLBACK
END
ELSE
PRINT 'Gia moi chap nhan duoc'
```

```
CREATE TRIGGER tg_KT
ON DeTai    FOR UPDATE   AS
DECLARE @kp_cu int, @kp_moi int
SELECT @kp_cu = KinhPhi FROM deleted
PRINT 'Kinh phi cu ='
PRINT CONVERT(varchar(6),@kp_cu)
SELECT @kp_moi = KinhPhi FROM inserted
PRINT 'Kinh phi moi ='
PRINT CONVERT(varchar(6),@kp_moi)
IF(@kp_moi > (@kp_cu * 1.10))
BEGIN
PRINT 'Tang qua 10%, khong chap nhan'
ROLLBACK
END
ELSE
PRINT 'Chap nhan. Cap nhat'
------
UPDATE DeTai
SET KinhPhi = KinhPhi * 1.05
WHERE DTID = 'DT01'
```

Lời hay ý đẹp

"Cho nhiều hơn nhận,
ngay kẻ tham cũng phải mến bạn.
Có đủ cơ trí ở trong lòng,
ngay ở lúc bất ngờ bạn cũng không bị dồn
đến bước đường cùng"

Vegetable Roots