Bài 8. Lập trình thủ tục trong SQL Server.

- Mục đích, yêu cầu: Cung cấp cho sinh viên kiến thức về lập trình thủ tục trong SQL Server.

- Hình thức tổ chức dạy học: Lý thuyết, trực tiếp + trực tuyến + tự học

- Thời gian: Lý thuyết(trên lớp: 3; online: 2) Tự học, tự nghiên cứu: 10

- Nội dung chính:

Lập trình thủ tục trong SQL Server

I. Thủ tục lưu trữ

1.1. Các khái niệm

Như đã đề cập ở các chương trước, SQL được thiết kế và cài đặt như là một ngôn ngữ để thực hiện các thao tác trên cơ sở dữ liệu như tạo lập các cấu trúc trong cơ sở dữ liệu, bổ sung, cập nhật, xoá và truy vấn dữ liệu trong cơ sở dữ liệu. Các câu lệnh SQL được người sử dụng viết và yêu cầu hệ quản trị cơ sở dữ liệu thực hiện theo chế độ tương tác.

Các câu lệnh SQL có thể được nhúng vào trong các ngôn ngữ lập trình, thông qua đó chuỗi các thao tác trên cơ sở dữ liệu được xác định và thực thi nhờ vào các câu lệnh, các cấu trúc điều khiển của bản thân ngôn ngữ lập trình được sử dụng.

Với thủ tục lưu trữ, một phần nào đó khả năng của ngôn ngữ lập trình được đưa vào trong ngôn ngữ SQL. Một thủ tục là một đối tượng trong cơ sở dữ liệu bao gồm một tập nhiều câu lệnh SQL được nhóm lại với nhau thành một nhóm với những khả năng sau:

• Các cấu trúc điều khiển (IF, WHILE, FOR) có thể được sử dụng trong thủ tục.

• Bên trong thủ tục lưu trữ có thể sử dụng các biến như trong ngôn ngữ lập trình nhằm lưu giữ các giá trị tính toán được, các giá trị được truy xuất được từ cơ sở dữ liệu.

•Một tập các câu lệnh SQL được kết hợp lại với nhau thành một khối lệnh bên trong

một thủ tục. Một thủ tục có thể nhận các tham số truyền vào cũng như có thể trả về các giá trị thông qua các tham số (như trong các ngôn ngữ lập trình). Khi một thủ tục lưu trữ đã được định nghĩa, nó có thể được gọi thông qua tên thủ tục, nhận các tham số truyền vào, thực thi các câu lệnh SQL bên trong thủ tục và có thể trả về các giá trị sau khi thực hiện xong.

Sử dụng các thủ tục lưu trữ trong cơ sở dữ liệu sẽ giúp tăng hiệu năng của cơ sở dữ liệu, mang lại các lợi ích sau:

- Đơn giản hoá các thao tác trên cơ sở dữ liệu nhờ vào khả năng module hoá các thao tác này.
- Thủ tục lưu trữ được phân tích, tối ưu khi tạo ra nên việc thực thi chúng nhanh hơn nhiều so với việc phải thực hiện một tập rời rạc các câu lệnh SQL tương đương theo cách thông thường.
- Thủ tục lưu trữ cho phép chúng ta thực hiện cùng một yêu cầu bằng một câu lệnh đơn giản thay vì phải sử dụng nhiều dòng lệnh SQL. Điều này sẽ làm giảm thiểu sự lưu thông trên mạng.
- Thay vì cấp phát quyền trực tiếp cho người sử dụng trên các câu lệnh SQL và trên các đối tượng cơ sở dữ liệu, ta có thể cấp phát quyền cho người sử dụng thông qua các thủ tục lưu trữ, nhờ đó tăng khả năng bảo mật đối với hệ thống.

1.2. Tạo thủ tục lưu trữ

Thủ tục lưu trữ được tạo bởi câu lệnh CREATE PROCEDURE với cú pháp như sau:

```
CREATE PROCEDURE tên_thủ_tục [(danh_sách_tham_số)]
[WITH
RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION] AS
Các_câu_lệnh_của_thủ_tục
```

Trong đó:

`tên_thủ_tục` Tên của thủ tục cần tạo. Tên phải tuân theo qui tắc định danh và

không được vượt quá 128 ký tự. `danh_sách_tham_số` Các tham số của thủ tục được khai báo ngay sau tên thủ tục và nếu thủ tục có nhiều tham số thì các khai báo phân cách nhau bởi dấu phẩy. Khai báo của mỗi một tham số tối thiểu phải bao gồm hai phần:

• tên tham số được bắt đầu bởi dấu @.

• kiểu dữ liệu của tham số

```
@mamonhoc nvarchar(10)
```

`RECOMPILE` Thông thường, thủ tục sẽ được phân tích, tối ưu và dịch sẵn ở lần gọi đầu tiên. Nếu tuỳ chọn WITH RECOMPILE được chỉ định, thủ tục sẽ được dịch lại mỗi khi được gọi.

`ENCRYPTION` Thủ tục sẽ được mã hoá nếu tuỳ chọn WITH ENCRYPTION được chỉ định. Nếu thủ tục đã được mã hoá, ta không thể xem được nội dung của thủ tục. `các_câu_lệnh_của_thủ_tục` Tập hợp các câu lệnh sử dụng trong nội dung thủ tục. Các câu lệnh này có thể đặt trong cặp từ khoá `BEGIN...END` hoặc có thể không.

Giả sử ta cần thực hiện một chuỗi các thao tác như sau trên cơ sở dữ liệu

1. Bổ sung thêm môn học *cơsởdữ liệu* có mã *TI-005* và số đơn vị học trình là 5 vào bảng MONHOC
2. Lên danh sách nhập điểm thi môn *cơsởdữ liệu* cho các sinh viên học lớp có mã *C24102*(tức là bổ sung thêm vào bảng DIEMTHI các bản ghi với cột MAMONHOC nhận giá trị *TI-005*, cột MASV nhận giá trị lần lượt là mã các sinh viên học lớp có mã *C24105* và các cột điểm là NULL).

Nếu thực hiện yêu cầu trên thông qua các câu lệnh SQL như thông thường, ta phải thực thi hai câu lệnh như sau:

```
INSERT INTO MONHOC VALUES('TI005','Cơ sở dữ liệu',5)
INSERT INTO DIEMTHI(MAMONHOC,MASV) SELECT 'TI005',MASV
```

```
FROM SINHVIEN WHERE MALOP='C24102'
```

Thay vì phải sử dụng hai câu lệnh như trên, ta có thể định nghĩa môt thủ tục lưu trữ với các tham số vào là @mamonhoc, @tenmonhoc, @sodvhtvà @malop như sau:

```
CREATE PROC sp_LenDanhSachDiem( @mamonhocNVARCHAR(10),
        @tenmonhocNVARCHAR(50), @sodvhtSMALLINT,
        @malopNVARCHAR(10))
AS
BEGIN
INSERT INTO monhoc
VALUES(@mamonhoc,@tenmonhoc,@sodvht) INSERT INTO
diemthi(mamonhoc,masv)
SELECT @mamonhoc,masv FROM sinhvien WHERE malop=@malop
END
sp_LenDanhSachDiem 'TI005','Cơ sở dữ liệu',5,'C24102'
```

1.3. Lời gọi thủ tục lưu trữ

Như đã thấy ở ví dụ ở trên, khi một thủ tục lưu trữ đã được tạo ra, ta có thể yêu cầu hệ quản trị cơ sở dữ liệu thực thi thủ tục bằng lời gọi thủ tục có dạng:

```
tên_thủ_tục[danh_sách_các_đối_số]
```

Số lượng các đối số cũng như thứ tự của chúng phải phù hợp với số lượng và thứ tự của các tham số khi định nghĩa thủ tục. Trong trường hợp lời gọi thủ tục được thực hiện bên trong một thủ tục khác, bên trong một trigger hay kết hợp với các câu lệnh SQL khác, ta sử dụng cú pháp như sau:

```
EXECUTE tên_thủ_tục [danh_sách_các_đối_số]
```

Thứ tự của các đối số được truyền cho thủ tục có thể không cần phải tuân theo thứ tự của các tham số như khi định nghĩa thủ tục nếu tất cả các đối số được viết dưới dạng:

```
@tên_tham_số= giá_trị
```

Lời gọi thủ tục ở ví dụ trên có thể viết như sau:

```
sp_LenDanhSachDiem @malop='C24102', @tenmonhoc='Cơ sở
dữ liệu', @mamonhoc='TI005', @sodvht=5
```

1.4. Sử dụng biến trong thủ tục

Ngoài những tham số được truyền cho thủ tục, bên trong thủ tục còn có thể sử dụng các biến nhằm lưu giữ các giá trị tính toán được hoặc truy xuất được từ cơ sở dữ liệu. Các biến trong thủ tục được khai báo bằng từ khoá DECLARE theo cú pháp như sau:

```
DECLARE @tên_biến kiểu_dữ_liệu
```

Tên biến phải bắt đầu bởi ký tự @ và tuân theo qui tắc về định danh. Ví dụ dưới đây minh hoạ việc sử dụng biến trong thủ tục. Trong định nghĩa của thủ tục dưới đây sử dung các biến chứa các giá trị truy xuất được từ cơ sở dữ liệu.

```
CREATE PROCEDURE sp_Vidu( @malop1 NVARCHAR(10), @malop2
                          NVARCHAR(10))
AS
DECLARE @tenlop1 NVARCHAR(30)
DECLARE @namnhaphoc1 INT
DECLARE @tenlop2 NVARCHAR(30)
DECLARE @namnhaphoc2 INT
SELECT @tenlop1=tenlop,@namnhaphoc1=namnhaphoc
FROM lop WHERE malop=@malop1
SELECT @tenlop2=tenlop, @namnhaphoc2=namnhaphoc
FROM lop WHERE malop=@malop2
PRINT @tenlop1+' nhap hoc nam '+str(@namnhaphoc1)
print @tenlop2+' nhap hoc nam '+str(@namnhaphoc2)
IF @namnhaphoc1=@namnhaphoc2
PRINT 'Hai lớp nhập học cùng năm
ELSE' PRINT 'Hai lớp nhập học khác năm’
```

1.5. Giá trị trả về của tham số trong thủ tục lưu trữ

Trong các ví dụ trước, nếu đối số truyền cho thủ tục khi có lời gọi đến thủ tục là biến, những thay đổi giá trị của biến trong thủ tục sẽ không được giữ lại khi kết thúc quá trình thực hiện thủ tục.

Xét câu lệnh sau đây

```
CREATE PROCEDURE sp_Conghaiso(@aINT,@b INT, @c INT)
AS SELECT @c=@a+@b
```

Nếu sau khi đã tạo thủ tục với câu lệnh trên, ta thực thi một tập các câu lệnh như sau:

```
DECLARE @tong INT
SELECT @tong=0
EXECUTE sp_Conghaiso 100,200,@tong
SELECT @tong
```

Câu lệnh "SELECT @tong" cuối cùng trong loạt các câu lệnh trên sẽ cho kết quả là: 0

Trong trường hợp cần phải giữ lại giá trị của đối số sau khi kết thúc thủ tục, ta phải khai báo tham số của thủ tục theo cú pháp như sau:

```
@tên_tham_số kiểu_dữ_liệu OUTPUT
@tên_tham_số kiểu_dữ_liệu OUT
```

hoặc và trong lời gọi thủ tục, sau đối số được truyền cho thủ tục, ta cũng phải chỉ định thêm từ khoá OUTPUT (hoặc OUT)

Ta định nghĩa lại thủ tục ở ví dụ trên như sau:

```
CREATE PROCEDURE sp_Conghaiso(@a INT, @b INT, @c INT
OUTPUT)
AS SELECT @c=@a+@b
```

và thực hiện lời gọi thủ tục trong một tập các câu lệnh như sau:

```
DECLARE @tong INT
SELECT @tong=0
```

```
EXECUTE sp_Conghaiso 100,200,@tong OUTPUT
SELECT @tong
```

thì câu lệnh "SELECT @tong" sẽ cho kết quả là: 300

1.6. Tham số với giá trị mặc định

Các tham số được khai báo trong thủ tục có thể nhận các giá trị mặc định. Giá trị mặc định sẽ được gán cho tham số trong trường hợp không truyền đối số cho tham số khi có lời gọi đến thủ tục.

Tham số với giá trị mặc định được khai báo theo cú pháp như sau:

```
@tên_tham_số kiểu_dữ_liệu = giá_trị_mặc_định
```

Trong câu lệnh dưới đây:

```
CREATE PROC sp_TestDefault( @tenlop NVARCHAR(30)=NULL,
                            @noisinh NVARCHAR(100)='Huế')
AS
BEGIN
IF @tenlop IS NULL
SELECT hodem,ten FROM sinhvien INNER JOIN lop ON
sinhvien.malop=lop.malop WHERE noisinh=@noisinh
ELSE
SELECT hodem,ten FROM sinhvien INNER JOIN lop ON
sinhvien.malop=lop.malop WHERE noisinh=@noisinh AND
tenlop=@tenlop

END
```

Thủ tục *sp_TestDefault*được định nghĩa với tham số @tenlop có giá trị mặc định là NULLvà tham số @noisinhcó giá trị mặc định là *Huế*. Với thủ tục được định nghĩa như trên, ta có thể thực hiện các lời gọi với các mục đích khác nhau như sau:

• Cho biết họ tên của các sinh viên sinh tại *Huế*:

```
sp_testdefault
```

• Cho biết họ tên của các sinh viên lớp *TinK24*sinh tại *Huế*:

```
sp_testdefault @tenlop='Tin K24'
```

• Cho biết họ tên của các sinh viên sinh tại *NghệAn*:

```
sp_testDefault @noisinh=N'Nghệ An'
```

• Cho biết họ tên của các sinh viên lớp *TinK26*sinh tại *ĐàNẵng*:

```
sp_testdefault @tenlop='Tin K26',@noisinh='Đà Nẵng'
```

Sửa đổi thủ tục

Khi một thủ tục đã được tạo ra, ta có thể tiến hành định nghĩa lại thủ tục đó bằng câu lệnh ALTER PROCEDURE có cú pháp như sau:

```
ALTER PROCEDURE tên_thủ_tục [(danh_sách_tham_số)] [WITH
RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION] AS
Các_câu_lệnh_Của_thủ_tục
```

Câu lệnh này sử dụng tương tự như câu lệnh CREATE PROCEDURE. Việc sửa đổi lại một thủ tục đã có không làm thay đổi đến các quyền đã cấp phát trên thủ tục cũng như không tác động đến các thủ tục khác hay trigger phụ thuộc vào thủ tục này.

1.7. Xoá thủ tục

Để xoá một thủ tục đã có, ta sử dụng câu lệnh DROP PROCEDURE với cú pháp như sau:

```
DROP PROCEDURE tên_thủ_tục
```

Khi xoá một thủ tục, tất cả các quyền đã cấp cho người sử dụng trên thủ tục đó cũng đồng thời bị xoá bỏ. Do đó, nếu tạo lại thủ tục, ta phải tiến hành cấp phát lại các quyền trên thủ tục đó.

Tài liệu tham khảo:

[1]. Giáo trình SQL Server – Đỗ Ngọc Sơn, Phan Văn Viên - Tài liệu lưu hành nội bộ của Trường Đại học Công nghiệp Hà Nội, 2015.

[2]. Giáo trình hệ quản trị cơ sở dữ liệu - Đỗ Ngọc Sơn; Phan Văn Viên; Nguyễn Phương Nga - NXB Khoa học Kỹ thuật

[3]. Bài tập Hệ quản trị Cơ sở dữ liệu – Phạm Văn Hà, Trần Thanh Hùng, Đỗ Ngọc Sơn, Nguyễn Thị Thanh Huyền – Trường Đại học Công nghiệp Hà Nội, 2020.