Hướng dẫn thực hành môn Hệ quản trị CSDL

Chủ đề 1
Stored-Procedure & Trigger & Function

Mục đích

- Xây dựng Stored procedure và trigger để thực hiện các chức năng của hệ thống.
- Sử dụng các lệnh Transact-SQL, Cursor.
- Bài tập thực hành:
  - File: Bai_tap_Chu_de_1_-_StoreProcedure_-_Trigger_-_Function.pdf
  - File: Dac_ta_CSDL_Quan_ly_thu_vien.pdf (các mục 4.1 → 4.13; 5.1 → 5.4).

Table of Contents

1. Stored-Procedure

1.1. Giới thiệu

Khi chúng ta tạo một ứng dụng với Microsoft SQL Server, ngôn ngữ lập trình Transact-SQL là ngôn ngữ chính giao tiếp giữa ứng dụng và database của SQL Server. Khi chúng ta tạo các chương trình bằng Transact-SQL, hai phương pháp chính có thể dùng để lưu trữ và thực thi cho các chương trình là:

- Chúng ta có thể lưu trữ các chương trình cục bộ và tạo các ứng dụng để gởi các lệnh đến SQL Server và xử lý các kết quả,
- Chúng ta có thể lưu trữ những chương trình như các stored procedure trong SQL Server và tạo ứng dụng để gọi thực thi các stored procedure và xử lý các kết quả.

Đặc tính của Stored-procedure trong SQL Server :

- ❖ Chấp nhận những tham số vào và trả về những giá trị được chứa trong các tham số ra để gọi những thủ tục hoặc xử lý theo lô.
- ❖ Chứa các lệnh của chương trình để thực hiện các xử lý trong database, bao gồm cả lệnh gọi các thủ tục khác thực thi.
- ❖ Trả về các trạng thái giá trị để gọi những thủ tục hoặc thực hiện các xử lý theo lô để cho biết việc thực hiện thành công hay thất bại, nếu thất bại thì lý do vì sao thất bại.

Ta có thể dùng Transact–SQL **EXCUTE** để thực thi các stored procedure. Stored procedure khác với các hàm xử lý là giá trị trả về của chúng không chứa trong tên và chúng không được sử dụng trực tiếp trong biểu thức.

Stored procedure có những thuận lợi so với các chương trình Transact-SQL lưu trữ cục bộ là:

- ❖ **Stored procedure cho phép điều chỉnh chương trình cho phù hợp:** Chúng ta có chỉ tạo stored procedure một lần và lưu trữ trong database một lần, trong chương trình chúng ta có thể gọi nó với số lần bất kỳ. Stored procedure có thể được chỉ rõ do một người nào đó tạo ra và sự thay đổi của chúng hoàn toàn độc lập với source code của chương trình.
- ❖ **Stored procedure cho phép thực thi nhanh hơn:** nếu sự xử lý yêu cầu một đoạn source code Transact – SQL khá lớn hoặc việc thực thi mang tính lặp đi lặp lại thì stored procedure thực hiện nhanh hơn việc thực hiện hàng loạt các lệnh Transact-SQL. Chúng được phân tích cú pháp và tối ưu hóa trong lần thực thi đầu tiên và một phiên bản dịch của chúng trong đó sẽ được lưu trong bộ nhớ để sử dụng cho lần sau, nghĩa là trong những lần thực hiện sau chúng không cần phải phân tích cú pháp và tối ưu lại, mà chúng sẽ sử dụng kết quả đã được biên dịch trong lần đầu tiên.
- ❖ **Stored procedure có thể làm giảm bớt vấn đề kẹt đường truyền mạng:** giả sử một xử lý mà có sử dụng hàng trăm lệnh của Transact-SQL và việc thực hiện thông qua từng dòng lệnh đơn, như vậy việc thực thông qua stored procedure sẽ tốt hơn, vì nếu không khi thực hiện chúng ta phải gởi hàng trăm lệnh đó lên mạng và điều này sẽ dẫn đến tình trạng kẹt mạng.
- ❖ **Stored procedure có thể sử dụng trong vấn đề bảo mật của máy:** vì người sử dụng có thể được phân cấp những quyền để sử dụng các stored procedure này, thậm chí họ không được phép thực thi trực tiếp những stored procedure này.

1.2. Định nghĩa

Một Stored procedure được định nghĩa gồm những thành phần chính sau:

- Tên của stored procedure
- Các tham số
- Thân của stored procedure: bao gồm các lệnh của Transact-SQL dùng để thực thi procedure.

Một stored procedure được tạo bằng lệnh **Create Procedure**, và có thể thay đổi bằng cách dùng lệnh **Alter Procedure**, và có thể xóa bằng cách dùng lệnh **Drop Procedure** trong lập lệnh của Transact – SQL

1.3. Cú pháp

1.3.1. Lệnh tạo Procedure

```
CREATE PROCEDURE  procedure_name
        {@parameter data_type  input/output }/*các biến tham số vào  ra*/
AS
Begin
        [khai báo các biến cho xử lý]
        {Các câu lệnh transact-sql}
End
```

- Ghi chú:
  - Trong SQL Server, có thể ghi tắt **một số** từ khóa mà tên có chiều dài hơn 4 ký tự. Ví dụ: có thể thay thế **Create Procedure** bằng **Create Proc**.
  - Tên hàm, tên biến trong SQL Server **không** phân biệt hoa thường.

1.3.2. Khai báo biến và gán giá trị cho biến, Ghi chú

```
/*Khai báo biến*/
DECLARE  @parameter_name data_type

/*Gán giá trị cho biến*/ SET
@parameter_name = value SELECT
@parameter_name = value

/*In thông báo ra màn hình*/
print N'Chuỗi thông báo unicode'

--Ghi chú 1, một dòng
/*
  Ghi chú 2
  Nhiều dòng
*/
```

1.3.3. Biên dịch và gọi thực thi một stored-procedure

- Biên dịch : Chọn toàn bộ mã lệnh Tạo stored-procedure → Nhấn F5
- Gọi thực thi một store-Procedure đã được biên dịch bằng lệnh exec:

```
EXECUTE procedure_name --Stored-proc không tham số
EXEC procedure_name Para1_value, Para2_value, … --Stored-proc có tham số
```

1.3.4. Lệnh cập nhật Procedure

```
ALTER PROCEDURE  procedure_name
        [ {@parameter data_type } ]
AS
Begin
        [khai báo các biến cho xử lý]
        {Các câu lệnh transact-sql}
End
```

1.3.5. Lệnh xóa Procedure

```
DROP PROCEDURE  procedure_name
```

1.4. Ví dụ

- Tạ o stored-procedure tính tổng của 2 số nguyên

```
--Tạo stored-procedure sp_tong
CREATE PROCEDURE sp_Tong
        @So1  int, @So2   int, @Tong  int out
AS
Begin
        SET @Tong = @So1 + @So2;
End

--Biên dịch stored-procedure → F5

--Kiểm tra
Declare @Sum int
Exec sp_Tong 1, -2, out @Sum
Select @Sum
```

- Tạo stored procedure liệt kê những thông tin của đầu sách, thông tin tựa sách và số lượng sách hiện chưa được mượn của một đầu sách cụ thể (ISBN).

```
CREATE PROCEDURE sp_ThongtinDausach
        @isbn int
AS
Begin
  SELECT tuasach, tacgia, ngonngu, bia, trangthai, count(*)
  FROM  dausach ds, tuasach ts, cuonsach cs
  WHERE
        ds.ma_tuasach = ts.ma_tuasach AND
        ds.isbn = cs.isbn AND
        ds.isbn = @isbn AND
        tinhtrang = yes
  GROUP BY tuasach, tacgia, ngonngu, bia, trangthai
End
```

2. Trigger

2.1. Giới thiệu

Trigger là một trường hợp đặc biệt của store procedure, nó sẽ có hiệu lực khi chúng ta thay đổi dữ liệu trên một bảng dữ liệu cụ thể, hoặc các xử lý làm thay đổi dữ liệu của các lệnh: insert, update, delete. Trigger có thể chứa các lệnh truy vấn từ các bảng khác hoặc bao gồm những lệnh SQL phức tạp.

Một số thuận lợi khi sử dụng trigger:

- **Trigger chạy một cách tự động:** chúng được kích hoạt ngay tức thì khi có sự thay đổi dữ liệu trên bảng dữ liệu.
- Trigger có thể thực hiện cascade khi việc thi hành có ảnh hưởng đến những bảng liên quan.
- Trigger có những hiệu lực ít bị hạn chế hơn so với ràng buộc giá trị nghĩa là có thể ràng buộc tham chiếu đến những cột của những bảng dữ liệu khác.
- Khi trigger được kích hoạt bởi 1 lệnh Transact-SQL insert để thêm một bộ mới vào bảng AAA thì bộ mới này được lưu tạm thời vào một bảng tạm có tên là **inserted** có cùng cấu trúc với bảng AAA. Khi kết thúc trigger này thì bộ dữ liệu mới thật sự lưu xuống CSDL.
- Tương tự đối với lệnh delete, các bộ dữ liệu bị xóa sẽ chuyển tạm vào bảng tạm **deleted**.

2.2. Cú pháp

2.2.1. Lệnh tạo Trigger

```
Create Trigger trigger_name on table_name
For     [insert,update,delete]
As
Begin
        {Khai báo các biến xử lý}
        {Các lệnh Transact-SQL}
End
```

2.2.2. Lệnh xóa Trigger

```
Drop Trigger trigger_Name
```

2.3. Ví dụ

Tạo trigger cho thao tác xóa một đầu sách trong bảng Muon.

```
CREATE TRIGGER tg_delMuon ON muon
FOR delete
AS
Begin
        DECLARE @isbn int, @ma_cuonsach smallint
        SELECT @isbn = isbn, @ma_cuonsach = ma_cuonsach
        FROM  deleted

        UPDATE cuonsach
        SET   tinhtrang = yes
        WHERE isbn = @isbn AND ma_cuonsach = @ma_cuonsach
End
```

3. Cursor

Cursor là một kiểu dữ liệu đặc biệt, được dùng để lưu trữ kết quả của câu lệnh **SELECT** trong quá trình lập trình.

3.1. Cú pháp

Lệnh khai báo biến cursor:

```
DECLARE cursor_name [INSENSITIVE] [SCROLL] CURSOR FOR Select_statement
```

Lệnh mở cursor:

```
OPEN cursor_name
```

Lấy dữ liệu từ trong cursor:

```
FETCH NEXT FROM cursor_name INTO @variable1, @variable2, …
```

Kiểm tra kết quả lấy dữ liệu từ cursor (kiểm tra ngay sau lệnh **FETCH NEXT**):

```
@@FETCH_STATUS = 0 : lấy dữ liệu thành công
@@FETCH_STATUS < 0 : không lấy được dữ liệu.
```

Đóng cursor:

```
CLOSE cursor_name
DEALLOCATE cursor_name
```

3.2. Ví dụ

Ví dụ 1 : Sử dụng cursor để duyệt dữ liệu trả về từ một câu select

```
--Khai báo biến
Declare @btt int, @btt2 int
declare @c cursor
set @c = cursor for Select tt,tt2 From bang1 Where [điều kiện]

--Mở cursor
open @c
fetch next from @c into @btt, @btt2

--Duyệt cursor
while @@fetch_status = 0
begin
```

```
        --Sử dụng 2 biến @btt, @btt2. Sau đó, gọi tiếp

        fetch next from @c into @btt, @btt2, ...
end

--Đóng cursor
close @c ;
deallocate @c ;
```

Ví dụ 2 : Sử dụng 2 cursor lồng nhau

```
declare @c cursor
set @c = cursor for select top 2 ma_docgia from DocGia

open @c

declare @madg varchar(66)
fetch next from @c into @madg

while @@fetch_status =0
begin
        print @madg ;

        --cursor @c2 dung binh thuong o day
        declare @c2 cursor ;
        set @c2 = cursor for select top 4 ma_tuaSach from TuaSach ;

        open @c2 ;
        declare @mats varchar(66) ;
        fetch next from @c2 into @mats ;

        while @@fetch_status = 0
        begin
              print ' ' + @mats ;
              fetch next from @c2 into @mats ;
         end

        close @c2 ;
        deallocate @c2 ;

        fetch next from @c into @madg ;
end

close @c
deallocate @c
```

4. Function

Trong SQL Server ta có thể viết hàm và lấy giá trị trả về. Các dạng hàm có thể viết như sau :

- Hàm trả về giá trị vô hướng (scalar value) : varchar, int, ….
- Hàm trả về giá trị là bảng tạm (inline table-valued) : table

4.1. Cú pháp

```
CREATE FUNTIONS function_name
( [@parameter_name parameter_data_type] )
RETURNS [return Data-type] /*Returns có 's' */
AS
Begin
	return [scalar value/select command]
End
```

4.2. Ví dụ

❖ Viết hàm tính tuổi của người có năm sinh là @ns :

```
--Xóa hàm nếu đã có
if object_id('fTuoi','FN') is not null
	drop function fTuoi
go

--Tạo hàm fTuoi
Create function fTuoi (@ns int)
Returns int
As
Begin
	return year(getdate()) - @ns
end
go

--Biên dịch hàm với F5
--Kiểm tra thử hàm
print dbo.fTuoi(1982) --phải có dbo.
```

❖ Viết hàm tạo bảng tạm từ một câu truy vấn :

```
--Xóa hàm nếu đã có
if object_id('fDSach','IF') is not null
	drop function fDSach
go

--Tạo hàm, giả sử trong CSDL ta đã có bảng T(namsinh int)
Create function fDSach (@ns int) --phải đặt tham số vào dấu ngoặc nhọn
Returns table
As
	Return (select * From T Where namsinh=@ns)
go
```

```
--Kiểm tra thử hàm
Select *
From fDSach(1982) --không cần dbo.
```

5. Bài tập

1. Làm các bài tập về Stored-procedure, Cursor, Trigger trong file Bai_tap_Chu_de_1_-_StoreProcedure_-_Trigger_-_Function.pdf
2. Làm các Store-procedure và trigger trong mục 4.1 → 4.13; 5.1 → 5.4 liên quan đến CSDL Quản lý thư viện trong file Dac_ta_CSDL_Quan_ly_thu_vien.pdf