CODEGYM

Raising the bar

Stored Procedure
Thủ tục lưu

Module: Java Web Back-end Development

Thảo luận bài cũ

- Hỏi và trao đổi về các khó khăn gặp phải trong bài "Các hàm thông dụng trong SQL"
- Tóm tắt lại các phần đã học từ bài "Các hàm thông dụng trong SQL"

Mục tiêu

- Trình bày được khái niệm chỉ mục (index)
- Tạo mới, sửa và xóa chỉ mục
- Sử dụng chỉ mục trong truy vấn SQL
- Trình bày được khái niệm khung nhìn (view)
- Tạo mới, sửa và xóa khung nhìn
- Trình bày được khái niệm thủ tục lưu (stored procedure)
- Tạo mới, sửa và xóa thủ tục lưu

Thủ tục lưu trữ - SPs

- Là tập hợp các câu lệnh transact-SQL được xem như một khối lệnh đơn nhằm thực hiện một tác vụ cụ thể.
- Hữu ích cho những tác vụ thực hiện lặp đi lặp lại
- Hỗ trợ các biến do người dùng khai báo, các điều kiện thực thi và các đặc trưng khác
- Ưu điểm:
  - Tăng tính bảo mật
  - Thực thi tiền biên dịch
  - Giảm thiểu lưu thông trong mô hình Client/Server
  - Khả năng sử dụng lại

Phân loại Thủ tục lưu

- Thủ tục lưu hệ thống – System SPs: được sử dụng để tương tác với các bảng hệ thống và thực hiện các tác vụ quản trị
- Thủ tục lưu mở rộng – Externed SPs: giúp SQL Server tương tác với hệ điều hành
- Thủ tục lưu do người dùng định nghĩa: còn gọi là Thủ tục lưu tùy chỉnh

Tạo thủ tục lưu

- Cú pháp:

```
CREATE  [PROC | PROCEDURE] procedure_name
[@parameter data_type[,]]
[ WITH RECOMPILE  |  ENCRYPTION ]
AS
<sql_statement>
```

- Ví dụ:

```
CREATE PROC Show_Customers
AS
SELECT * FROM Customer_Details
Go
Execute Show_Customers
```

Tạo thủ tục lưu

- Sử dụng mệnh đề "OUTPUT": Nhận dữ liệu trả về từ thủ tục lưu
- Yêu cầu sử dụng output cả khi định nghĩa và chạy thủ tục
- Ví dụ:

```
CREATE PROC Max_Salary
@max_sal int output
AS
SELECT @max_sal=MAX(Salary) FROM Employee_Details
```

➢

```
DECLARE @max_sal int
EXEC Max_Salary @max_sal output
PRINT @max_sal
```

Cú pháp điều khiển

- begin … end : đánh dấu khối lệnh

  begin
  {statements | statement_block}
  end

- if … else:

  if condition_expression
  {statements | statement_block}
  else
  {statements | statement_block}

Hướng dẫn tạo thủ tục lưu

- Khi một bảng tạm cục bộ được tạo bên trong một thủ tục lưu, bảng đó sẽ mất đi khi thủ tục lưu kết thúc.
- Thủ tục lưu có thể tham chiếu đến bảng, khung nhìn, hàm người dùng định nghĩa hay các thủ tục lưu khác.
- Khi một thủ tục lưu gọi thủ tục lưu khác, thủ tục được gọi có thể truy cập tất cả các đối tượng được tạo bởi thủ tục gọi.

Thực thi thủ tục lưu

- Câu lệnh EXECUTE được sử dụng để chạy các thủ tục do người dùng định nghĩa.
- Cú pháp:

  EXEC[UTE] procedure_name [parameters,]
- Ví dụ:

  EXECUTE Titles_1389

Xem thông tin thủ tục lưu

- Sử dụng sp_helptext

  Execute sp_helptext '<procedure_name>'

- Sử dụng OBJECT_DEFINITION

  SELECT OBJECT_DEFINITION(OBJECT_ID('<sp_name>')

- Sử dụng sys.sql_modules

  SELECT definition FROM sys.sql_modules WHERE object_id = OBJECT_ID('sp_name')

- Sử dụng sp_depends

Sửa thủ tục lưu

- Cú pháp:

```
ALTER [PROC | PROCEDURE] procedure_name
[@parameter data_type]
AS
<sql_statement>
```

- Ví dụ:

```
ALTER PROCEDURE Titles_Pub
@v_pubid char(4)
AS
DECLARE @v_return int
SELECT @v_return=COUNT(*)
FROM titles WHERE pub_id = @v_pubid
IF @v_return>0
  SELECT * FROM titles WHERE pub_id = @v_pubid
ELSE
    RETURN @v_return+1
```

Xóa thủ tục lưu

- Cú pháp:

  DROP PROCEDURE procedure_name

- Ví dụ:

  Drop Procedure Display_Customers

Biên dịch lại các thủ tục lưu

- Để phản ánh sự thay đổi tới các chỉ số.
- Có ba cách để biên dịch lại các thủ tục:
  - Sử dụng sp_recompile

    sp_recompile [@objectname =] 'object'
  - Chỉ rõ WITH RECOMPILE khi CREATE

    create proc *tên_thủ_tục* with recompile
  - Chỉ rõ WITH RECOMPILE khi EXECUTE

    Exec *tên_thủ_tục* with recompile

Làm việc với thủ tục lưu

- Thủ tục lưu lồng nhau
- Điều khiển thông báo lỗi
- Hàm “@@ERROR”

Thủ tục lưu lồng nhau

- Có thể gọi thủ tục lưu bên trong một thủ tục lưu khác
- Thủ tục lưu được gọi này lại có thể gọi đến một hoặc nhiều thủ tục lưu khác
- Ví dụ:

```
CREATE PROC NestProc
AS
Begin
Execute display_customers;
Execute city_Customers ‘New York’;
End
```

Điều khiển thông báo lỗi

- Sử dụng cú pháp "Try .... Catch"
- Cú pháp:

```
BEGIN TRY
        <sql_statement>
END TRY
BEGIN CATCH
        <sql_statement>
END CATCH
```

- Ví dụ:

```
CREATE PROC Error_SP as
Declare @result int
Select  'This will be executed'
Select @result = 'Hello'
Select 'This will not be executed'
go
BEGIN TRY
EXEC  Error_SP
END TRY
BEGIN CATCH
SELECT Error_message() as Error
END CATCH
```

Các hàm báo lỗi

- Error_line (): số dòng nơi lỗi xuất hiện
- Error_number(): mã số của lỗi
- Error_message(): văn bản thông báo lỗi
- Error_procedure(): tên thủ tục lưu nơi lỗi
- Error_sererity(): mức độ nghiêm trọng
- Error_state(): mã số tình trạng lỗi

Thông báo lỗi RAISEROR

- Trả về các mã hoặc lệnh RAISERROR có thể được dùng để đưa ra các lỗi của người dùng
- Trả về mã trong thủ tục lưu trữ là các giá trị nguyên
- Lệnh RAISERROR statement ghi các lỗi và gán các cấp độ nghiêm trọng của lỗi
- Ví dụ:

```
WHILE @v_ctr > 0
BEGIN
    SELECT @v_ctr * @v_ctr
    SELECT @v_ctr = @v_ctr – 1
    IF @v_ctr = 2
    BEGIN
    RAISERROR('Counter has fallen
    below 3', 1, 2)
    BREAK
END
END
```

Tóm tắt bài học

- Trình bày được khái niệm chỉ mục (index)
- Tạo mới, sửa và xóa chỉ mục
- Sử dụng chỉ mục trong truy vấn SQL
- Trình bày được khái niệm khung nhìn (view)
- Tạo mới, sửa và xóa khung nhìn
- Trình bày được khái niệm thủ tục lưu (stored procedure)
- Tạo mới, sửa và xóa thủ tục lưu

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: Database and CRUD

Raising the bar