BÀI 3: Thủ tục lưu trữ (Stored Procedure)

Ngày học | @October 9, 2023

3.1 Thủ tục là gì?

Thủ tục là đối tượng được tổ chức trong CSDL nhằm cung cấp khả năng module hóa các phép xử lý dữ liệu.

Trong thủ tục là các tập lệnh T-SQL được lập trình và khi thực thi thủ tục, các câu lệnh bên trong phần thân thủ tục được thực thi.

Thủ tục được “biên dịch” ở lần thực thi đầu tiên và chiến lược thực thi được lưu trữ lại để sử dụng cho các lần gọi sau. Chính vì vậy, việc thực thi các lệnh trong thủ tục thường nhanh hơn so với không sử dụng trong thủ tục.

Ưu điểm:

- Giúp đơn giản hóa các phép xử lý dữ liệu phức tạp (module hóa).
- Tăng hiệu năng khi xử lý dữ liệu (nhanh hơn).
- Tăng tính an toàn vào bảo mật đối với dữ liệu: thay vì cho phép người sử dụng thao tác trực tiếp trên các bảng, chúng ta có thể tạo ra các thủ tục và cấp quyền cho người sử dụng thông qua các thủ tục (được lập trình, được kiểm soát, kiểm tra lỗi,…).
- Giảm băng thông mạng.
- Linh hoạt với sự thay đổi của dữ liệu.

Sử dụng cũng có thể làm tăng độ phức tạp, độ khó khi triển khai các CSDL.

3.2 Tạo và sử dụng thủ tục

Để tạo thủ tục, sử dụng câu lệnh CREATE PROCEDURE hoặc CREATE PROC, có có pháp:

```
IF EXISTS (SELECT * FROM sys.objects WHERE name = 'Tên_thủ_tục')
DROP PROCEDURE Tên_thủ_tục
GO
CREATE PROCEDURE Tên_thủ_tục
      Danh_sách_tham_số_(nếu có)
AS
BEGIN
      SET NOCOUNT ON;
      Phần_thân_của_thủ_tục
END
GO
```

Chú ý:

- Tên thủ tục là phải duy nhất trong mỗi CSDL.
- Tên thủ tục nên sử dụng tiền tố để nhận biết (VD: proc_InsertData, sp_InsertData, usp_InsertData,…). Nên tránh dùng tiền tố sp_ vì đây là tiền tố sử dụng cho các thủ tục của hệ thống.
- Tham số của thủ tục được khai báo theo cú pháp

```
@Tên_tham_số Kiểu_dữ_liệu [ = Giá_trị_mặc_định]
```

Tên tham số phải bắt đầu bởi @

Các tham số khai báo phân cách nhau bởi dấu phẩy

- Câu lệnh CREATE PROCEDURE không được nằm chung trong một tập lệnh với các câu lệnh khác.

VD: Tạo thủ tục có 2 tham số @categoryName và @description có chức năng:

- Bổ dung thêm một loại hàng mới nếu @categoryName chưa tồn tại
- Ngược lại thì cập nhật Description cho loại hàng đã tồn tại

```
CREATE PROCEDURE proc_Category_InsertOrUpdate
    @categoryName nvarchar(255),
    @description nvarchar(255)
AS
BEGIN
    SET NOCOUNT ON; --Tắt chế độ đếm số dòng tác động bởi câu lệnh
    IF NOT EXISTS(SELECT * FROM Categories WHERE CategoryName = @categoryName)
        INSERT INTO Categories(CategoryName, Description)
        VALUES (@categoryName, @description)
    ELSE
      UPDATE Categories
      SET Description = @description
      WHERE CategoryName = @categoryName;
END
GO
```

(Thủ tục sau khi đã tạo thì nó tồn tại trong CSDL, được quản lý trong nhóm Programming → Store Procedures)

Một thủ tục đã tồn tại thì không thể tạo lại nó trừ khi chúng ta xóa nó đi.

- Để xóa thủ tục, sự dụng lệnh:

DROP PROCEDURE Tên_thủ_tục

- Để thay đổi 1 thủ tục (đã tồn tại) sử dụng lệnh:

ALTER PROCEDURE Tên_thủ_tục
    Danh_sách_tham_số
AS
    Phần_thân_của_thủ_tục
GO

Thông thường, khi tạo 1 thủ tục, chúng ta sẽ kiểm tra xem nó đã tồn tại hay chưa, nếu đã tồn tại thì xóa để tạo lại

VD:

```
IF EXISTS (SELECT * FROM sys.objects
           WHERE name = 'proc_Category_InsertOrUpdate')
    DROP PROCEDURE proc_Category_InsertOrUpdate
GO

CREATE PROCEDURE proc_Category_InsertOrUpdate
    @categoryName nvarchar(255),
    @description nvarchar(255)
AS
BEGIN
    SET NOCOUNT ON; --Tắt chế độ đếm số dòng tác động bởi câu lệnh
    IF NOT EXISTS(SELECT * FROM Categories WHERE CategoryName = @categoryName)
        INSERT INTO Categories(CategoryName, Description)
        VALUES (@categoryName, @description)
    ELSE
      UPDATE Categories
      SET Description = @description
      WHERE CategoryName = @categoryName;
END
GO
```

Sử dụng thủ tục (thực thi/gọi thủ tục)

EXECUTE Tên_thủ_tục Danh_sách_đối_số_truyền_cho_thủ_tục
hoặc
EXEC Tên_thủ_tục Danh_sách_đối_số

Trong đó:

> **- Danh sách đối số của thủ tục có thể viết theo 1 trong 2 cách**
> **+ Cách 1: Liệt kê giá trị các đối số, phân cách nhau bởi dấu phẩy. Thứ tự của các đối số phải giống với thứ tự các tham số khi định nghĩa thủ tục.**
> **+ Cách 2: Liệt kê tên tham số và giá trị truyền cho nó phân cách nhau bởi dấu phẩy (thứ tự không quan trọng)**

VD:

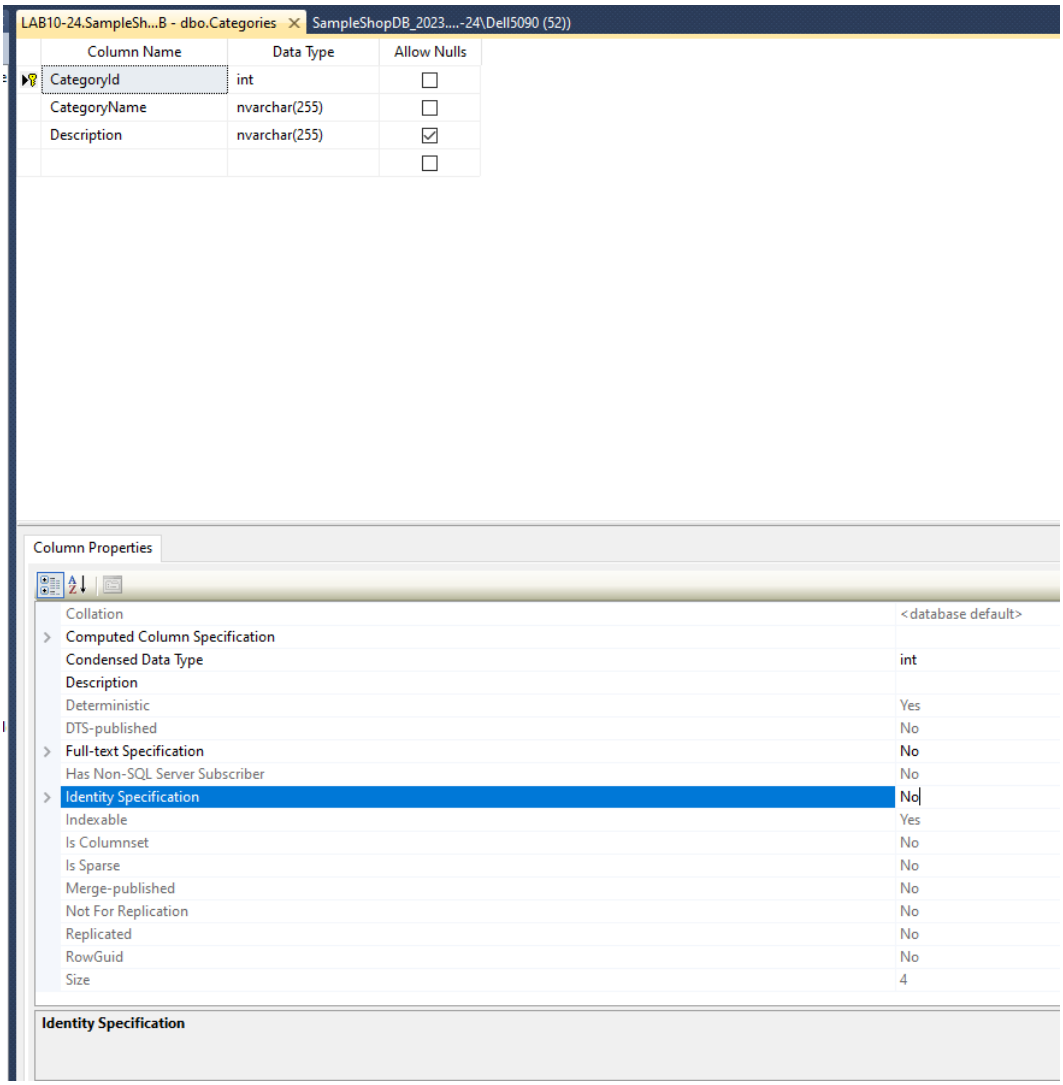
```
-- C1
EXECUTE proc_Category_InsertOrUpdate N'Thực phẩm', N'Cá, thịt, mắm,…'

-- C2
EXECUTE proc_Category_InsertOrUpdate
          @description = N'Các loại hàng thực phẩm',
          @categoryName = N'Thực phẩm';
```

> ## Lưu ý: Nên sử dụng cách viết thứ 2

Để chạy trực tiếp 1 thủ tục, có thể dùng cách sau:

> 💡 **Nhấp phải trên thủ tục (trong cửa sổ Object Explorer)**
> **Chọn Execute Stored Procedure**

Lưu ý:



Set cột CategoryID thuộc tính Identity thành YES

VD: Viết thủ tục proc_Order_GetDetails

@orderId int

Có chức năng hiển thị danh sách các mặt hàng được bán trong đơn hàng có mã là @orderId, thông tin cần hiển thị bao gồm: mã hàng, tên hàng, đơn vị tính, số lượng bán, giá bán và thành tiền (thành tiền = số lượng bán * giá bán)

```
IF EXISTS (SELECT * FROM sys.objects WHERE name = 'proc_Order_GetDetails')
  DROP PROCEDURE proc_Order_GetDetails
GO

CREATE PROCEDURE proc_Order_GetDetails
    @orderId int
```

```
AS
BEGIN
  SET nocount on;
  SELECT *
  FROM OrderDetails AS od JOIN Products as p
    ON p.ProductId = od.ProductId
  WHERE od.OrderId = @orderId;
END
GO
-- Test
EXECUTE proc_Order_GetDetails @orderID = 10250
```

3.3 Tham số của thủ tục

- Nếu một tham số mà không có giá trị mặc định thì bắt buộc phải được truyền giá trị khi sử dụng.
- Trong trường hợp tham số có giá trị mặc định thì có thể bỏ qua không truyền giá trị khi sử dụng.
- Tham số đầu ra (output): là tham số mà sự thay đổi về giá trị của nó trong thủ tục sẽ được giữ nguyên sau khi thoát khỏi thủ tục (tương tự như tham biến trong ngôn ngữ lập trình).
- Tham số đầu ra khi khai báo có thêm từ khóa **output**

@Tên_tham_số Kiểu_dữ_liệu output

- Khi sử dụng, nếu muốn 1 tham số là được dùng dưới dạng output thì:
  - Phải truyền bằng biến
  - Phải kèm theo từ khóa **output** khi truyền đối số (từ output trong SQL như & trong code) (truyền ở tạo thủ tục và truyền đối đối số đó khi thực thi)

```
CREATE PROCEDURE proc_Test
      @a int,
      @b int,
      @c int output
AS
    SET @a = @a * 2;
```

```
    SET @b = @b * 3;
    SET @c = @a + @b; -- Thay đổi giá trị của @c
GO

-- Case 1:
DECLARE @x int = 10,
        @y int = 20,
        @z int = 5000;

EXECUTE proc_Test @a = @x,
                  @b = @y,
                  @c = @z output;
select @x, @y, @z
```

VD: Viết thủ tục:

proc_GetRevenueByDates

@fromDate date,

@toDate date,

@totalRevenue money output

Có chức năng hiển thị doanh thu bán hàng từng ngày trong khoảng thời gian từ ngày @fromDate đến ngày @toDate, đồng thời cho biết tổng doanh thu trong khoảng thời gian trên thông qua tham số đầu ra @totalRevenue

```
IF EXISTS (SELECT * FROM sys.objects WHERE name = 'proc_GetRevenueByDates')
  DROP PROCEDURE proc_Order_GetDetails
GO
CREATE PROCEDURE proc_GetRevenueByDates
  @fromDate date,
  @toDate date,
  @totalRevenue money output
AS
BEGIN
  SET nocount on;
  WITH cte_Ngay as
  (
    SELECT @fromDate as SummaryDate
    UNION all
    SELECT DATEADD(DAY, 1, SummaryDate)
    FROM cte_Ngay
    WHERE SummaryDate < @toDate
  )
  , cte_ThongKe as
  (
    SELECT O.OrderDate, SUM(od.Quantity * od.SalePrice) as Revenue
```

```
    FROM Orders as o JOIN OrderDetails as od on o.OrderId = od.OrderId
    WHERE o.OrderDate between @fromDate and @toDate
    GROUP BY OrderDate
  )
  SELECT t1.SummaryDate, ISNULL(t2.Revenue, 0) as Revenue
  FROM cte_Ngay as t1
    LEFT JOIN cte_ThongKe as t2 on t1.SummaryDate = t2.OrderDate

  SELECT @totalRevenue = SUM(od.Quantity * od.SalePrice)
  FROM Orders AS o join OrderDetails as od on o.OrderId = od.OrderId
  WHERE o.OrderDate between @fromDate and @toDate;
END
GO

-- Test case
DECLARE @fromDate date = '2018/02/01',
    @toDate date = '2018/02/28',
    @totalRevenue money;
EXECUTE proc_GetRevenueByDates
    @fromDate = @fromDate,
    @toDate = @toDate,
    @totalRevenue = @totalRevenue output;
SELECT @totalRevenue AS totalRevenue
```

Tham số đầu vào kiểu bảng

Để sử dụng tham số đầu vào của thủ tục dưới dạng bảng, cần:

- Định nghĩa kiểu dữ liệu dạng bảng, sử dụng cú pháp:

CREATE TYPE Tên_kiểu AS TABLE
(
    Cấu_trúc_bảng
)

- Khai báo tham số đầu vào kiểu bảng cho thủ tục theo cú pháp:

@Tên_tham_số Kiểu_dữ_liệu_bảng READONLY
(READONLY là bắt buộc)

Tham số kiểu bảng khi truyền phải truyền bằng biến và nó có tính chất “chỉ đọc” khi sử dụng trong thủ tục (tức là chỉ được phép dùng lệnh SELECT đối với tham số này)

1 thủ tục khi sử dụng 1 kiểu dữ liệu dạng bảng ~ khóa kiểu dữ liệu đó lại

VD: Viết thủ tục proc_Order_Create có chức năng khởi tạo một đơn hàng và danh mục được bán hàng trong đơn hàng đó. **Đầu vào** của thủ tục bao gồm:

- Ngày lập đơn hàng
- Danh sách các mặt hàng được bán trong đơn hàng (mỗi mặt hàng bao gồm các thông tin là mã hàng, số lượng và giá bán)

Tham số **đầu ra** cho biết mã đơn hàng được tạo và tổng số tiền của đơn hàng là bao nhiêu? (CHÚ Ý: Cột OrderID của bảng Orders có tính chất IDENTITY)

```
CREATE TYPE TypeOrderDetails AS TABLE
(
  ProductId int primary key,
  Quantity int,
  SalePrice money
)
GO

IF EXISTS (SELECT * FROM sys.objects WHERE name = 'proc_GetRevenueByDates')
  DROP PROCEDURE proc_GetRevenueByDates;
GO

CREATE PROCEDURE proc_Order_Create
  @orderDate date,
  @orderDatails TypeOrderDetails readonly,
  @orderID int output,
  @sumOfPrice money output
AS
BEGIN
  SET nocount on;
  -- Tạo đơn hàng mới
  INSERT INTO Orders(OrderDate) values (@orderDate);

  SET @orderID = SCOPE_IDENTITY();
  -- SET @orderID = @@IDENTITY;: lấy giá trị indentity sinh ra ở lệnh gần nhất

  -- Bổ sung chi tiết đơn hàng
  INSERT INTO OrderDetails(OrderId, ProductId, Quantity, SalePrice)
  SELECT @orderID, ProductID, Quantity, SalePrice
  FROM @orderDatails;

  -- Tính tổng tiền
```

```
  SELECT @sumOfPrice = SUM(Quantity * SalePrice)
  FROM @orderDatails

END
GO

-- test
DECLARE @orderDetails TypeOrderDetails,
    @orderID int,
    @sumOfPrice money;
INSERT INTO @orderDetails VALUES(1, 20, 20.00),
                (2, 5, 25.00),
                (3, 2, 15.00)

EXECUTE proc_Order_Create
    @orderDate = '2023/10/09',
    @orderDatails = @orderDetails,
    @orderID = @orderID output,
    @sumOfPrice = @sumOfPrice output;

SELECT @orderID, @sumOfPrice

/* SELECT *
FROM OrderDetails as od JOIN Products as p ON od.ProductId = p.ProductId
WHERE */
```

BÀI TẬP: Viết lại tất cả các câu ở bài thực hành 2 bằng thủ tục

BT_09.10.rar