Nghiên cứu và xây dựng các Functions, Procedures trên môi trường MS SQL Server

Trình bày: Nguyễn Thị Sim

Time: 2019 October

Giới thiệu SQL Server

Định nghĩa SQL Serrver

- Là phần mềm được Microsoft phát triển dựa trên RDBMS.
- Là hệ quản trị cơ sở dữ liệu quan hệ đối tượng ORDBMS.
- Là một nền tảng độc lập.
- Phần mềm sử dụng cả giao diện dòng lệnh và giao diện GUI.

Mục đích sử dụng SQL Server

- Tạo cơ sở dữ liệu.
- Duy trì cơ sở dữ liệu.
- Phân tích dữ liệu bằng SSAS-SQL Server Analysis Services.
- Tạo báo cáo bằng SSRS-SQL Server Reporting Services.
- Thực hiện quá trình ELT(Extract-Transform-Load) bằng SSIS-SQL Server Intergration Services.

1. Function trong SQL Server

Function là một đối tượng trong cơ sở dữ liệu bao gồm một tập nhiều câu lệnh được nhóm lại với nhau và được tạo ra với mục đích sử dụng lại. Trong SQL Server, function được lưu trữ và có thể truyền các tham số vào cũng như trả về các giá trị. Các hàm sẽ giúp đơn giản hóa chương trình và có thể tái sử dụng nhiều lần.

Có 2 kiểu Function:

- Hàm trả về giá trị kiểu giá trị cụ thể (SCALAR VALUED)
- Hàm trả về giá trị kiểu TABLE(TABLE VALUED)

1.1. Tạo mới Function (Create Function)

Để tạo một function trong SQL Server, ta sử dụng cú pháp:

```
CREATE FUNCTION [schema_name.]function_name
( [ @parameter [ AS ] [type_schema_name.] datatype
 [ = default ] [ READONLY ]
 , @parameter [ AS ] [type_schema_name.] datatype
 [ = default ] [ READONLY ] ]
)

RETURNS return_datatype

[ WITH { ENCRYPTION
 | SCHEMABINDING
 | RETURNS NULL ON NULL INPUT
 | CALLED ON NULL INPUT
 | EXECUTE AS Clause ]

[ AS ]

BEGIN

[declaration_section]

executable_section

RETURN return_value

END;
```

Tham số:

- *schema_name:* Tên schema (lược đồ) sở hữu function.
- *function_name:* Tên gán cho function.
- *@parameter:* Một hay nhiều tham số được truyền vào hàm.
- *type_schema_name:* Kiểu dữ liệu của schema (nếu có).
- *Datatype:* Kiểu dữ liệu cho @parameter.

- *Default:* Giá trị mặc định gán cho @parameter.
- *READONLY:* @parameter không thể bị function ghi đè lên.
- *return_datatype:* Kiểu dữ liệu của giá trị trả về.
- *ENCRYPTION:* Mã nguồn (source) của function sẽ không được lưu trữ dưới dạng text trong hệ thống.
- *SCHEMABINDING:* Đảm bảo các đối tượng không bị chỉnh sửa gây ảnh hưởng đến function.
- *RETURNS NULL ON NULL INPUT*: Hàm sẽ trả về NULL nếu bất cứ parameter nào là NULL.
- *CALL ON NULL INPUT*: Hàm sẽ thực thi cho dù bao gồm tham số là NULL.
- *EXECUTE AS* clause: Xác định ngữ cảnh bảo mật để thực thi hàm.
- *return_value:* Giá trị được trả về.
- RETURNS NULL ON NULL INPUT: hàm sẽ trả về NULL nếu bất cứ parameter nào là NULL
- CALL ON NULL INPUT: hàm sẽ thực thi cho dù bao gồm tham số là NULL
- EXECUTE AS clause: xác định ngữ cảnh bảo mật để thực thi hàm

Áp dụng cho 2 kiểu giá trị trả về:

.Hàm trả về giá trị kiểu giá trị cụ thể:

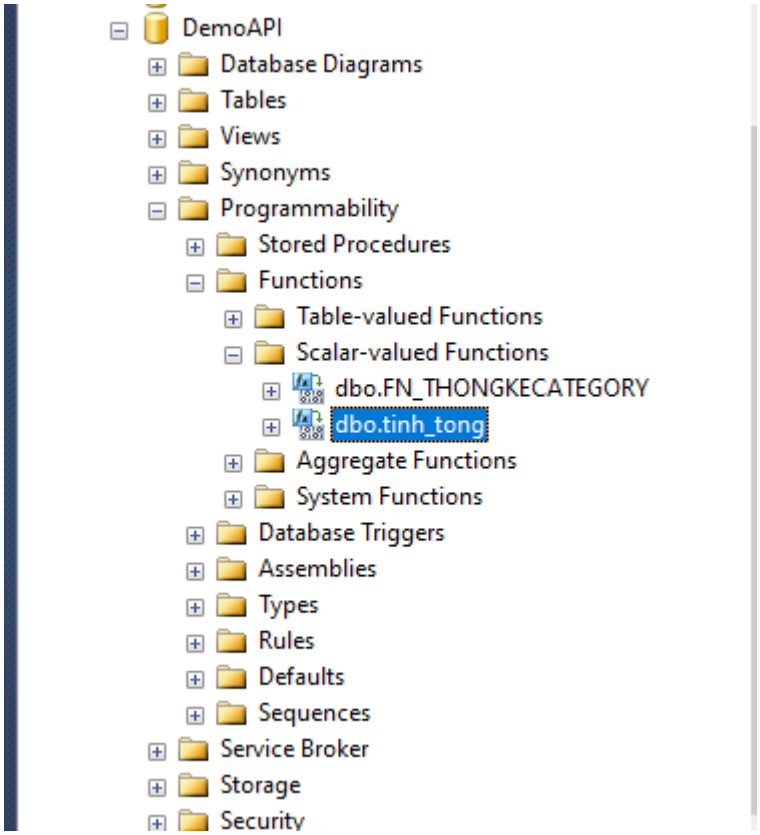
```
CREATE FUNCTION FN_TENHAM(@THAMBIEN1 KIEUDL(KT),,,,)
RETURNS KIEUDL TRA VE
AS
  BEGIN
    DECLARE @BIEN KIEUDL(KT)
    --XU LY TREN HAM
    RETURN @BIEN
  END
Gọi hàm: SELECT DBO.TENHAM(DOISO1,DOISO2, , ,)
```

Ví dụ: Đây là một Function có chức năng tính tổng hai số.

```
CREATE FUNCTION tinh_tong(
   @a INT,
   @b INT
)
```

```
RETURNS INT
AS
BEGIN
  RETURN @a * @b;
END;
```

Sau khi chạy khởi tạo hàm này thì chúng ta sẽ thấy nó trong SSMS bằng cách click vào **Programmability > Functions > Scalar-valued Functions**.



Để gọi hàm đã tạo ở trên ta gõ: SELECT dbo.tinh_tong(10, 20)

.Hàm trả về kiểu bảng: thay thể cho view, view không truyền được tham biến nên ta phải tạo Function trả về Table.

```
CREATE FUNCTION TENHAM(@THAMBIEN KIEUDL(KT).....)

RETURNS @TENBANG TABLE(

            TRUONG1 KIEUDL1(KT),

            TRUONG2 KIEUDL2(KT),

            ...

          )
```

```
AS
  BEGIN
    INSERT INTO @TENBANG
                SELECT TRUONG1,TRUONG2,...
    RETURN
  END
```

Gọi hàm: SELECT * FROM TENHAM(DOISO1,DOISO2, , ,)

1.2. Sửa Function trong SQL Server

Để sửa thay đổi nội dung của một Function ta thay CREATE bằng ALTER.

```
ALTER FUNCTION [schema_name.]function_name (parameter_list)
  RETURN data_type AS
  BEGIN
    statements
    RETURN value
  END
```

Ví dụ:

```
ALTER FUNCTION FN_THONGKE(@TITLE NVARCHAR(50))
RETURNS INT
AS
 BEGIN
   DECLARE @TONG INT
   SELECT @TONG=COUNT(*) FROM Courses INNER JOIN Categories ON
Courses.CategoryId=Categories.CategoryId
   WHERE Title=@TITLE and CourseName='TOIEC'
   RETURN @TONG
 END
```

1.3. Xóa bỏ Function (Drop Function)

Để xóa bỏ một function, ta có cú pháp sau:

```
FUNCTION [schema_name. ]function_name;
```

Tham số: Function_name: tên function chúng ta muốn xóa bỏ.

Ví dụ:

```
DROP FUNCTION FN_THONGKE
```

Thực hiện lệnh này là chúng ta đã xóa bỏ hàm FN_THONGKE khỏi database.

Lưu ý:

Mỗi function có thể sử dụng ở bất cứ đâu trong câu lệnh T-SQL và nằm trong phạm vi database.

Có thể có nhiều tham số, tuy nhiên chỉ trả về được một giá trị duy nhất, bắt buộc phải return.

Có thể sử dụng câu lệnh T-SQL bên trong function.

Function này có thể sử dụng function khác.

2. Stored Procedure trong SQL Server

Procedure là thủ tục nội tại không trả về giá trị và là một chương trình trong cơ sở dữ liệu gồm nhiều câu lệnh mà chúng ta lưu lại cho những lần sử dụng sau. Trong SQL Server, chúng ta có thể truyền các tham số vào procedure, tuy nó không trả về một giá trị cụ thể như function nhưng nó cho biết việc thực thi thành công hay thất bại.

Ưu điểm của Stored Procedure là:

- Động: 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.
- Nhanh hơn: Stored procedure có khả năng 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. Do đó stored procedure có khả năng thực thi nhanh hơn là việc xử lý một đoạn lệnh Transact – SQL lớn, lặp.

Định nghĩa về một stored procedure:

- Tên của stored procedure
- Các tham số
- Thân của stored procedure: bao gồm các lệnh

2.1. Lệnh tạo mới Procedure (Create procedute)

```
CREATE PROC SP_TENTHUTUC(@THAMBIEN1 KIEUDL1(KT),...)
AS
 BEGIN
   DECLARE @BIEN1 KIEUDL1(KT) --BIẾN CỤC BỘ
   ...
   XỬ LÝ TRONG THỦ TỤC
 END
```

Gọi thủ tục: EXEC **TENTHUTUC DOISO1**,...

Ví dụ: viết thủ tục nhập vào Category với các tham biến CategoryId, CategoryName, hãy kiểm tra xem CategoryName đã tồn tại trước đó hay chưa, nếu đã tồn tại đưa ra thông báo, nếu chưa tồn tại thì nhập vào bảng Category.

```
CREATE PROC SP_InputCategory(@CategoryId INT,@CategoryName
NVARCHAR(50))
 AS
   BEGIN
     IF(EXISTS(SELECT * FROM Categories WHERE
CategoryName=@CategoryName))
       PRINT 'Category Name'+@CategoryName+'DA TON TAI'
     ELSE
       INSERT INTO Categories VALUES(@CategoryId,@ CategoryName)
  END
```

Kiểm tra: SELECT * FROM **Categories**

EXEC **SP_ InputCategory** 6,'XYZ'

2.2. Lệnh cập nhật Stored Procedure trong SQL Server

```
ALTER PROC SP_TENTHUTUC(@THAMBIEN1 KIEUDL1(KT),...)
AS
 BEGIN
   DECLARE @BIEN1 KIEUDL1(KT) --BIẾN CỤC BỘ
   ...
   XỬ LÝ TRONG THỦ TỤC
 END
```

Hoặc click chuột phải vào stored cần sửa, sau đó chọn menu Modify thì nó sẽ hiển thị ra một trang query mới với cấu trúc của stored cũ.

Ví dụ:

```
ALTER PROC SP_InputCategory(@CategoryId INT,@CategoryName
NVARCHAR(50))
 AS
   BEGIN
     IF(EXISTS(SELECT * FROM Categories WHERE
CategoryName=@CategoryName))
       PRINT 'Category Name'+@CategoryName+' NAY DA TON TAI'
     ELSE
       INSERT INTO Categories VALUES(@CategoryId,@ CategoryName)
  END
```

2.3. Lệnh xóa bỏ Procedure(Drop Procedure) trong SQL Server

```
DROP PROCEDURE procedure_name;
```

Tham số: Procedure_name: tên procedure bạn muốn xóa bỏ.

Ví dụ: DROP PROCEDURE SP_InputCategory

3. Bài tập thực hành

Cho DB có 3 table: **Authors, Categories, Courses**

Authors([AuthorId],[Name],[Email],[Password],[DateCreated],[Coin])

Categories([CategoryId][Title][Active])

Courses([CourseId] ,[CategoryId],[AuthorId],[CourseName] ,[Price] ,[Discount],[CourseArtUrl],[Description],[Available],[DateCreated])

Đề bài:

1,Viết hàm thống kê xem mỗi Category có bao nhiêu Course với tên category(Title) được truyền vào.

2,Đưa ra danh sách bao gồm CourseId, CouseNamecủa các Course với Author Name XYZ nhập từ bàn phím.

3,Viết thủ tục nhập vào Category với các tham biến CategoryId,Title,Active. Hãy kiểm tra xem Title đã tồn tại trước đó hay chưa, nếu đã tồn tại đưa ra thông báo, nếu chưa tồn tại thì nhập vào bảng Category.

Đáp án:

1, Viết hàm thống kê xem mỗi Category có bao nhiêu Course với tên category(Title) được truyền vào.

```
CREATE FUNCTION FN_THONGKECATEGORY(@TITLE NVARCHAR(50))
RETURNS INT
AS
 BEGIN
   DECLARE @TONG INT
   SELECT @TONG=COUNT(*) FROM Courses INNER JOIN Categories ON
Courses.CategoryId=Categories.CategoryId
   WHERE Title=@TITLE
   RETURN @TONG
 END
Test:   select dbo.FN_THONGKECATEGORY('Techlonogy')
```

2,Đưa ra danh sách bao gồm CourseId, CouseName của các Course với Author Name XYZ nhập từ bàn phím.

```
CREATE FUNCTION DSCOURSE(@NAME NVARCHAR(50))
 RETURNS @COURSE TABLE(
            CourseId INT,
            CourseName NVARCHAR(50)
            )
AS
  BEGIN
    INSERT INTO @COURSE
                SELECT CourseId,CourseName
                FROM Authors INNER JOIN Courses
                ON Authors.AuthorId=Courses.AuthorId
                WHERE Authors.Name=@NAME
    RETURN
END
```

Test: `Select * from DSCOURSE('Name')`

3,Viết thủ tục nhập vào Category với các tham biến CategoryId,Title,Active. Hãy kiểm tra xem Title đã tồn tại trước đó hay chưa, nếu đã tồn tại đưa ra thông báo, nếu chưa tồn tại thì nhập vào bảng Category.

```
CREATE PROC SP_NHAPCATEGORY(@TITLE NVARCHAR(50),@ACTIVE BIT)
 AS
   BEGIN
     IF(EXISTS(SELECT * FROM Categories WHERE Title=@TITLE))
       PRINT 'TITLE '+@@TITLE+'DA TON TAI'
     ELSE
       INSERT INTO Categories VALUES(@TITLE,@ACTIVE)
  END
```

Test: `EXEC SP_NHAPCATEGORY 'XYZ',1`

Tổng kết

Trong bài nghiên cứu này, chúng ta đã nghiên cứu sử dụng Function và Stored Procedure trên môi trường MS SQL Server. Chúng ta đã biết cách tạo mới, thay đổi , xóa và sử dụng một Function cũng như một Stored Procedure trong một Database cùng với các ví dụ và bài tập thực tiễn.