

Function

NGUYEN HongPhuong

Email: phuongnh@soict.hust.edu.vn

Site: <http://users.soict.hust.edu.vn/phuongnh>

Face: <https://www.facebook.com/phuongnhbk>

Hanoi University of Science and Technology

Nội dung

- ❑ Hàm vô hướng trong SQL Server (scalar function)
- ❑ Cú pháp tạo function
- ❑ Chỉnh sửa scalar function
- ❑ Biến kiểu bảng
- ❑ Table function

Hàm vô hướng trong SQL Server

- ❑ Yêu cầu một hoặc nhiều tham số, trả về một giá trị đơn
- ❑ Giúp đơn giản hóa đoạn mã của lập trình viên. Ví dụ: câu truy vấn Select có phép tính phức tạp, thì có thể sử dụng hàm vô hướng đóng gói công thức này, và sử dụng nó trong mỗi lần truy vấn

Cú pháp tạo function

```
CREATE FUNCTION [schema_name.]function_name
( [@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;
```

Ý nghĩa

- ❑ `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.

Ý nghĩa

- ❑ 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.

Ví dụ 1

```
CREATE FUNCTION fNhanvien
(@nhanvien_id INT)
RETURNS VARCHAR(50)
AS
BEGIN
    DECLARE @nhanvien_name VARCHAR(50);
    IF @nhanvien_id < 10
    SET @nhanvien_name = 'Smith';
    ELSE
    SET @nhanvien_name = 'Lawrence';
    RETURN @nhanvien_name;
END;

SELECT dbo.fNhanvien(8);
```

Ví dụ 2: Với CSDL BikeStores

□ Với CSDL BikeStores

```
CREATE FUNCTION sales.fNetSale
(
    @quantity INT,
    @list_price DEC(10,2),
    @discount DEC(4,2)
)
RETURNS DEC(10,2)
AS
BEGIN
    RETURN @quantity * @list_price * (1 - @discount);
END;
```

```
SELECT sales.fNetSale(10,100,0.1) net_sale;
```


Ví dụ 2: Với CSDL BikeStores

```
SELECT order_id,  
       SUM(sales.fNetSale(quantity, list_price, discount)) net_amount  
FROM sales.order_items  
GROUP BY order_id  
ORDER BY net_amount DESC;
```

Chỉnh sửa scalar function

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

Chỉnh sửa scalar function

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

Xóa scalar function

❑ DROP FUNCTION

[schema_name.]function_name;

❑ DROP FUNCTION sales.fNetSale;

Một số điểm chính về scalar function

- ❑ Có thể sử dụng hầu hết mọi nơi trong các câu lệnh T-SQL
- ❑ Chấp nhận 1 hoặc nhiều tham số nhưng chỉ trả về 1 giá trị đơn, nên phải có 1 câu lệnh RETURN
- ❑ Có thể sử dụng logic như khối IF hoặc vòng lặp WHILE
- ❑ Không thể UPDATE dữ liệu. Không nên truy cập dữ liệu
- ❑ Có thể gọi function khác

Biến kiểu bảng trong SQL Server

- ❑ Biến kiểu bảng cho phép lưu trữ các bản ghi dữ liệu, tương tự như bảng tạm
- ❑ Cách khai báo biến kiểu bảng:

```
DECLARE @table_variable_name TABLE  
(  
    column_list  
);
```

- ❑ Phạm vi biến kiểu bảng
 - Không còn tồn tại sau khi kết thúc khối lệnh
 - Nếu định nghĩa biến kiểu bảng trong 1 SP hoặc Function, nó sẽ không tồn tại sau khi SP hoặc Function kết thúc

Biến kiểu bảng trong SQL Server

□ Ví dụ

```
DECLARE @product_table TABLE
(
    product_name VARCHAR(MAX) NOT NULL,
    brand_id INT NOT NULL,
    list_price DEC(11,2) NOT NULL
);
```

Chèn dữ liệu vào biến kiểu bảng

```
DECLARE @product_table TABLE (  
    product_name VARCHAR(MAX) NOT NULL,  
    brand_id INT NOT NULL,  
    list_price DEC(11,2) NOT NULL  
);
```

```
INSERT INTO @product_table  
SELECT product_name, brand_id, list_price  
FROM production.products  
WHERE category_id = 1;
```

```
SELECT * FROM @product_table;
```


Hạn chế của biến kiểu bảng

- ❑ Phải định nghĩa cấu trúc của biến kiểu bảng, không thể thay đổi cấu trúc của biến kiểu bảng sau khi đã khai báo giống như bảng thông thường hoặc bảng tạm
- ❑ Các biến kiểu bảng không chứa số liệu thống kê, nên không giúp ích gì cho trình tối ưu hóa truy vấn để đưa ra kế hoạch thực hiện truy vấn tốt. Chỉ nên sử dụng biến kiểu bảng để lưu trữ ít bản ghi.

-
- ❑ Không sử dụng biến kiểu bảng làm tham số đầu vào, đầu ra như các kiểu dữ liệu khác. Tuy nhiên, có thể trả về biến kiểu bảng từ function
 - ❑ Không thể tạo non-clustered index cho biến kiểu bảng. Từ SQL Server 2014 cho phép thêm non-clustered index như một phần của khai báo biến kiểu bảng

-
- ❑ Nếu đang sử dụng biến kiểu bảng với JOIN, cần đặt bí danh cho bảng
 - ❑ Ví dụ:

```
SELECT
    brand_name,
    product_name,
    list_price
FROM
    production.brands b
INNER JOIN @product_table pt
    ON b.brand_id = pt.brand_id;
```

Hiệu suất của biến kiểu bảng trong SQL Server

- ❑ Sử dụng biến kiểu bảng trong SP sẽ biên dịch lại ít hơn so với sử dụng bảng tạm
- ❑ Sử dụng ít tài nguyên hơn một bảng tạm với ít chi phí khóa và ghi nhật ký hơn.
- ❑ Các biến kiểu bảng thực hiện trong CSDL tempdb, không phải trong bộ nhớ (tương tự bảng tạm)

Sử dụng biến kiểu bảng trong user-defined function

```
CREATE FUNCTION udfSplit
(
    @string VARCHAR(MAX),
    @delimiter VARCHAR(50) = ' ')
RETURNS @parts TABLE
(
    idx INT IDENTITY PRIMARY KEY,
    val VARCHAR(MAX)
)
AS
BEGIN
    DECLARE @index INT = -1;
    WHILE (LEN(@string) > 0)
    BEGIN
        SET @index = CHARINDEX(@delimiter , @string);
        IF (@index = 0) AND (LEN(@string) > 0)
```

Sử dụng biến kiểu bảng trong user-defined function

```
BEGIN
  INSERT INTO @parts
  VALUES (@string);
  BREAK
END
IF (@index > 1)
BEGIN
  INSERT INTO @parts
  VALUES (LEFT(@string, @index - 1));
  SET @string = RIGHT(@string, (LEN(@string) - @index));
END
ELSE
SET @string = RIGHT(@string, (LEN(@string) - @index));
END
RETURN
END

SELECT * FROM udfSplit('foo,bar,baz',',');
```

Table function trong SQL Server

- Hàm bảng là hàm do người dùng định nghĩa, trả về kiểu dữ liệu bảng. Kiểu trả về của table function là một bảng, do đó, có thể sử dụng table function giống như sử dụng bảng.

Tạo và thực thi table function

```
CREATE FUNCTION udfProductInYear
```

```
(
```

```
    @model_year INT
```

```
)
```

```
RETURNS TABLE
```

```
AS
```

```
RETURN
```

```
    SELECT
```

```
        product_name,  
        model_year,  
        list_price
```

```
FROM
```

```
    production.products
```

```
WHERE
```

```
    model_year = @model_year;
```

```
SELECT *
```

```
FROM udfProductInYear(2017);
```

```
SELECT product_name, list_price
```

```
FROM udfProductInYear(2018);
```


Sửa đổi table function

```
ALTER FUNCTION udfProductInYear (  
    @start_year INT,  
    @end_year INT  
)  
RETURNS TABLE  
AS  
RETURN  
    SELECT  
        product_name,  
        model_year,  
        list_price  
    FROM  
        production.products  
    WHERE  
        model_year BETWEEN @start_year AND @end_year  
  
    SELECT  
        product_name,  
        model_year,  
        list_price  
    FROM  
        udfProductInYear(2017,2018)  
    ORDER BY  
        product_name;
```

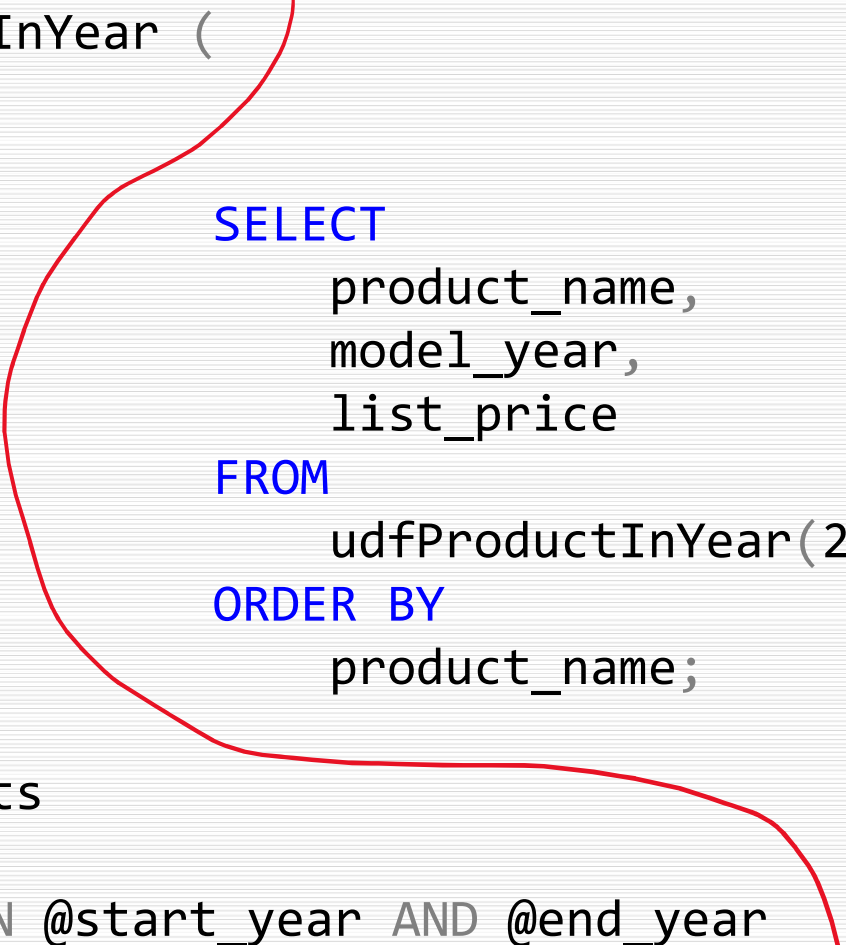


Table function đa câu lệnh

- ❑ Là hàm có nhiều câu lệnh và trả về giá trị kiểu bảng
- ❑ Hữu ích, vì có thể thực hiện nhiều truy vấn trong function và tổng hợp kết quả vào bảng được trả về
- ❑ Để định nghĩa table function, sử dụng biến kiểu bảng làm giá trị trả về. Bên trong function, thực hiện một/nhiều truy vấn chèn dữ liệu vào biến kiểu bảng này

Table function đa câu lệnh

□ Ví dụ

```
CREATE FUNCTION udfContacts()  
  RETURNS @contacts TABLE (  
    first_name VARCHAR(50),  
    last_name  VARCHAR(50),  
    email     VARCHAR(255),  
    phone     VARCHAR(25),  
    contact_type VARCHAR(20)  
  )  
AS  
BEGIN  
  INSERT INTO @contacts  
  SELECT first_name, last_name,  
         email, phone, 'Staff'  
  FROM sales.staffs;
```

```
INSERT INTO @contacts  
SELECT  
  first_name,  
  last_name,  
  email,  
  phone,  
  'Customer'  
FROM sales.customers;  
RETURN;  
END;  
  
SELECT *  
FROM udfContacts();
```

Khi nào nên sử dụng table function?

- ❑ Sử dụng table function như là một view có tham số (view động)
- ❑ So với stored procedure, table function linh hoạt hơn, bởi vì có thể sử dụng table function ở bất kỳ chỗ nào mà bảng được sử dụng.

Xóa function

□ Câu lệnh

```
DROP FUNCTION [ IF EXISTS ] [ schema_name. ] function_name;
```

□ Chú ý, câu lệnh xóa này sẽ thất bại:

- Nếu function được tham chiếu trong view hoặc function khác được tạo bằng tùy chọn WITH SCHEMABINDING
- Nếu có các ràng buộc CHECK, DEFAULT và các cột được tính toán liên quan đến function

❑ Xóa nhiều function

```
DROP FUNCTION [IF EXISTS]
  schema_name.function_name1,
  schema_name.function_name2,
  ...;
```

Ví dụ

```
CREATE FUNCTION sales.udf_get_discount_amount
(
    @quantity INT,
    @list_price DEC(10,2),
    @discount DEC(4,2)
)
RETURNS DEC(10,2)
AS
BEGIN
    RETURN @quantity * @list_price * @discount
END

DROP FUNCTION sales.udf_get_discount_amount;
```

❑ Xóa function có WITH SCHEMABINDING

```
CREATE FUNCTION sales.udf_get_discount_amount
(
    @quantity INT,
    @list_price DEC(10,2),
    @discount DEC(4,2)
)
RETURNS DEC(10,2)
WITH SCHEMABINDING
AS
BEGIN
    RETURN @quantity * @list_price * @discount
END
```

```
CREATE VIEW sales.discounts
WITH SCHEMABINDING
AS
SELECT
    order_id,
    SUM(sales.udf_get_discount_amount(
        quantity,
        list_price,
        discount
    )) AS discount_amount
FROM
    sales.order_items i
GROUP BY
    order_id;
```

❑ Thử xóa, sẽ có lỗi

```
DROP FUNCTION sales.udf_get_discount_amount;
```

-
- Muốn xóa, phải xóa view trước

```
DROP VIEW sales.discounts;
```

```
DROP FUNCTION sales.udf_get_discount_amount;
```

