Lập trình trong T-SQL

Cấu trúc lập trình

Giới thiệu

Bên cạnh các thành phần truy vấn theo chuẩn, T-SQL còn có các cấu trúc lập trình, bao gồm:

- DECLARE: khai báo biến.
- SET: gán giá trị cho biến.
- IF ... ELSE: câu rẽ nhánh
- WHILE: câu lặp

Nhóm lệnh

Một nhóm lệnh T-SQL là tập hợp một hay nhiều câu lệnh SQL nằm giữa BEGIN và END, một đoạn chương trình được xem là một đơn vị duy nhất khi thực thi

```
BEGIN
    <câu lệnh SQL>
END
```

Khối lệnh (BATCHES)

Một khối lệnh là tập hợp một hay nhiều câu lệnh SQL được SQL Server xử lý như một đơn vị duy nhất.

Một khối lệnh thường kết thúc bằng từ GO (tuy nhiên đây không phải là từ khóa của SQL).

Biến được khai báo trong một khối lệnh thì không thể sử dụng trong khối lệnh khác.

Khối lệnh (BATCHES)

Do khối lệnh được SQL Server xử lý như một đơn vị duy nhất, nên khi có lỗi cú pháp trong bất kỳ câu lệnh nào, khối lệnh sẽ không thực thi.

Tuy nhiên, nếu có lỗi khi thực thi câu lệnh, các câu lệnh khác vẫn có thể được thực thi bình thường.

Phân biệt nhóm lệnh và khối lệnh

Tuy cả hai đều là tập hợp các câu lệnh, từ khóa GO có tác dụng tách rời biến cũng như các câu lệnh thành nhóm riêng độc lập với nhau.

Khai báo biến đơn trị

Để khai báo biến trong T-SQL, ta có các cú pháp sau

```
DECLARE <tên biến> <kiểu dữ liệu>
```

Trong đó, <tên biến> bắt buộc phải bắt đầu bằng ký tự @

Gán dữ liệu cho biến đơn trị

Có thể gán giá trị cho biến bằng các cách sau

```
-- cách 1
DECLARE <tên biến> <kiểu dữ liệu>
SET <tên biến> = <giá trị> | <câu truy vấn>
-- cách 2
DECLARE <tên biến> <kiểu dữ liệu> = <giá trị> | <câu truy vấn>
-- cách 3
DECLARE <tên biến> <kiểu dữ liệu>
SELECT <tên biến> = ... FROM ...
```

Gán giá trị cho biến đơn trị

Lưu ý: Câu truy vấn chỉ có thể trả về một kết quả duy nhất.

Khai báo biến bảng

Tương tự như biến thông thường, biến bảng có thể được khai báo như sau

```
DECLARE <tên biến> TABLE (
    <định nghĩa bảng>
)
```

Các cấu trúc điều khiển

T-SQL cung cấp một số cấu trúc để điều khiển chương trình, bao gồm:

- IF ... ELSE ...
- WHILE ...

Cấu trúc điều khiển

Cú pháp IF .. ELSE ...

```
IF <điều kiện>
    BEGIN
        <câu lệnh SQL>
    END
ELSE
    BEGIN
        <câu lệnh SQL>
    END
```

Cấu trúc điều khiển

Cú pháp WHILE

```
WHILE <điều kiện>
    BEGIN
        <câu lệnh SQL>
    END
```

Trong câu lệnh WHILE, có thể sử dụng các từ khóa BREAK, CONTINUE.

Thủ tục (Stored Procedure)

Giới thiệu

Thủ tục là một đối tượng trong T-SQL. Thủ tục là một nhóm các câu lệnh SQL nhằm thực hiện một thao tác nào đó.

Thủ tục có thể thực hiện rất nhiều tác vụ:

- Trả về kết quả.
- Làm việc với dữ liệu.
- Bảo trì cơ sở dữ liệu.
- ...

Ưu điểm

Thủ tục có rất nhiều ưu điểm:

- Tái sử dụng
- Bảo mật
- Nâng cao chất lượng
- Nâng cao hiệu năng
- Giảm chi phí bảo trì

Phân loại

Có các loại thủ tục sau:

- Thủ tục do người dùng định nghĩa.
- Thủ tục tạm.
- Thủ tục hệ thống.

Cách gọi thủ tục

Thủ tục có thể được gọi như sau

```
[EXEC] <tên thủ tục> [tham số thủ tục]
[GO]
```

```
-- ví dụ
-- chạy thủ tục sp_rename để đổi tên bảng
[EXEC] sp_rename 'tbl', 'tbl_new'
```

Cách gọi thủ tục

Để truyền tham số, có thể dùng cách truyền theo vị trí hoặc truyền theo tên tham số

Cách viết thủ tục

Để tạo một thủ tục, ta dùng cú pháp sau

```
CREATE PROCEDURE <tên thủ tục> [(tham số)]
AS BEGIN
    <câu lệnh sql>
END
```

Lưu ý:

- Chỉ có thể tạo thủ tục ở đầu khối lệnh (batches).
- <tham số> có dạng <tên tham số> <kiểu dữ liệu>.

Cách viết thủ tục

```
-- Ví dụ, thủ tục không tham số
CREATE PROCEDURE hello AS
BEGIN
    DECLARE @now datetime2 = CURRENT_TIMESTAMP
    PRINT N'Bây giờ là ' + CONVERT(varchar(25), @now, 120)
END
```

Cách viết thủ tục

```
-- Ví dụ, thủ tục có tham số
CREATE PROCEDURE count_product (@min_price INT) AS
BEGIN
    SELECT COUNT(*)
    FROM Products
    WHERE Price > @min_price
END
```

Trả về tham số

Trong nhiều trường hợp, như paging, chúng ta cần lưu trữ giá trị trả về của thủ tục để xử lý tiếp. Khi đó, chúng ta dùng từ khóa OUTPUT để lấy giá trị trả về.

```
CREATE PROCEDURE <tên thủ tục>
    (<tham số muốn trả về> OUTPUT, ...)
AS BEGIN
    <câu lệnh sql>
END
```

Gọi thủ tục có trả về tham số

Để gọi thủ tục có trả về tham số, có thể gọi như sau

```
[EXEC] <tên thủ tục> <biến lưu trữ> = <tham số trả về> OUTPUT,
...
[GO]
```

Thủ tục trả về tham số

```
-- Ví dụ, thủ tục có trả về tham số
CREATE PROCEDURE get_id (@name NVARCHAR(10), @id INT OUTPUT) AS
BEGIN
    INSERT tbl VALUES (@name);
    SELECT @id = SCOPE_IDENTITY();
END
-- Để gọi thủ tục trên, ta có thể gọi như sau
DECLARE @inserted_id INT
EXEC get_id N'abc', @id = @inserted_id OUTPUT
PRINT @inserted_id
```

Xem định nghĩa thủ tục

Để xem định nghĩa của một thủ tục, ta có thể dùng

```
EXEC sp_helptext <tên thủ tục>
```

Chỉnh sửa thủ tục

Để chỉnh sửa một thủ tục có sẵn, ta dùng cú pháp

```
ALTER PROCEDURE <tên thủ tục> ...
```

```
CREATE OR ALTER PROCEDURE <tên thủ tục> ...
```

Xóa thủ tục

Để xóa thủ tục, ta dùng

```
DROP PROCEDURE <tên thủ tục>
```

Hàm (Function)

Giới thiệu

Hàm là một đối tượng trong T-SQL. Giống như thủ tục, hàm cũng là tập hợp các câu lệnh để hoàn thành một nhiệm vụ nào đó. Tuy nhiên, giữa hàm và thủ tục có nhiều điểm khác nhau.

Phân biệt hàm và thủ tục

Hàm	Thủ tục
• Không thể sử dụng DML • Bắt buộc phải trả về một giá trị • Có thể sử dụng trong câu truy vấn • Không thể bẫy lỗi (TRY ... CATCH) • Không thể gọi thủ tục trong hàm	• Có thể sử dụng DML • Có thể không trả về giá trị • Không thể sử dụng trong câu truy vấn • Có thể bẫy lỗi • Có thể gọi hàm trong thủ tục

Phân loại hàm

Có hai dạng hàm:

- Hàm trả về một giá trị.
- Hàm trả về một bảng.

Cách viết hàm

```
-- Hàm trả về một giá trị
CREATE FUNCTION <tên hàm> [(<tham số>)]
RETURNS <kiểu dữ liệu>
AS BEGIN
    <câu truy vấn>
    RETURN <kết quả>
END
```

Cách viết hàm

```
-- Hàm trả về một bảng, sử dụng một câu truy vấn
CREATE FUNCTION <tên hàm> [(<tham số>)]
RETURNS TABLE
AS RETURN
    <câu truy vấn>
```

Cách viết hàm

```
-- Hàm trả về một bảng, sử dụng nhiều câu truy vấn
CREATE FUNCTION <tên hàm> [(<tham số>)]
RETURNS <biến bảng> TABLE <(định nghĩa bảng)>
AS BEGIN
    <các câu truy vấn>
END
```

Sử dụng hàm

Như đã nói, hàm có thể được tại bất kỳ phần nào của câu truy vấn (SELECT, FROM, WHERE, ...) (miễn là kết quả của hàm phù hợp với thành phần đó).

Lưu ý: Khi gọi hàm được viết bằng CREATE FUNCTION, cần thêm **dbo.** vào trước tên hàm.

Các thao tác chỉnh sửa, xem và xóa

Tương tự như với thủ tục, chúng ta có

```
-- Xem
EXEC sp_helptext <tên hàm>
-- Chỉnh sửa
ALTER FUNCTION <tên hàm> ...
CREATE OR ALTER FUNCTION <tên hàm> ...
-- Xóa
DROP FUNCTION <tên hàm>
```