



Quản Trị Dữ Liệu Với Microsoft SQL Server

Chương: 13

Lập trình Với Transact-SQL



Mục tiêu

- Mô tả tổng quan lập trình Transact-SQL
- Mô tả các thành phần của lập trình Transact-SQL
- Mô tả các câu lệnh điều khiển luồng chương trình
- Mô tả các hàm Transact-SQL khác nhau
- Giải thích các thủ tục để tạo và sửa các hàm do người dùng định nghĩa (UDFs)
- Giải thích việc tạo các cửa sổ(windows) với OVER
- Mô tả về các hàm window(Describe window functions)

- Lập trình Transact-SQL là:
 - một ngôn ngữ thủ tục mở rộng cho SQL.
 - được mở rộng bằng cách thêm vào các chương trình con (subroutines) và lập trình cấu trúc tương tự như các ngôn ngữ bậc cao.
- Lập trình Transact-SQL cũng có các nguyên tắc(rules) và cú pháp điều khiển và cho phép các câu lệnh lập trình làm việc cùng nhau.
- Người dùng có thể điều khiển luồng chương trình bằng các câu lệnh điều kiện như `IF` và vòng lặp như `WHILE`.

Các thành phần ngôn ngữ Transact-SQL 1-2

Các thành phần ngôn ngữ lập trình Transact-SQL cho phép cho phép thực hiện nhiều thao tác không thể thực hiện bằng một câu lệnh đơn lẻ.

Người dùng có thể nhóm nhiều câu lệnh Transact-SQL với nhau bằng cách dùng một trong các cách sau:

Các lô (Batches)

- Là một tập hợp gồm một hay nhiều câu lệnh Transact-SQL được gửi đi như một khối từ ứng dụng tới server.

Thủ tục lưu (Stored Procedures)

- Là một tập hợp các Transact-SQL đã được định nghĩa và biên dịch từ trước trên server.

Triggers

- Là một dạng thủ tục lưu đặc biệt được thực thi khi người dùng thực hiện một sự kiện như các thao tác INSERT, DELETE, hoặc UPDATE trên bảng.

Kịch bản (Scripts)

- Là một dãy các câu lệnh Transact-SQL được lưu trữ trong một file được sử dụng là đầu vào cho trình soạn thảo code SSMS và tiện ích `sqlcmd`.

Các thành phần ngôn ngữ Transact-SQL 2-2

Các tính năng sau đây cho phép người dùng làm việc với các câu lệnh Transact-SQL:

Biến (Variables)

- Cho phép người dùng lưu trữ dữ liệu có thể được dùng làm đầu vào cho câu lệnh Transact-SQL.

Điều khiển luồng (Control-of-flow)

- Được sử dụng cho các cấu trúc điều kiện trong Transact-SQL.

Quản lý lỗi (Error Handling)

- Là kỹ thuật được sử dụng cho quản lý lỗi và cung cấp thông tin cho người dùng về lỗi đã xảy ra.

Lô(batch) trong Transact-SQL 1-5

Là một nhóm của một hoặc nhiều câu lệnh Transact-SQL được gửi tới server như là một khối(unit) từ ứng dụng để thực thi.

SQL Server biên dịch các câu lệnh SQL trong lô (batch) thành một khối có thể thực thi duy nhất, hay còn được gọi là một execution plan.

Trong execution plan, các câu lệnh SQL được thực hiện từng lệnh một.

Câu lệnh lô Transact-SQL nên được kết thúc bằng dấu chấm phẩy (semicolon).

Lỗi biên dịch như lỗi cú pháp làm hạn chế việc biên dịch execution plan.

Lô(batch) trong Transact-SQL 2-5

Lỗi thực thi(run-time error) như vi phạm ràng buộc (constraint) hoặc tràn số sẽ gây một trong các ảnh hưởng sau:

- Phần lớn các lỗi thực thi (run-time errors) sẽ làm dừng câu lệnh hiện tại và những câu lệnh tiếp theo trong lô (batch).
- Lỗi thực thi cụ thể như vi phạm ràng buộc không chỉ dừng thực hiện các câu lệnh hiện có mà còn cả các câu lệnh còn lại trong lô được thực hiện.

Các câu lệnh SQL thực hiện trước khi có lỗi thực thi xảy ra sẽ không bị ảnh hưởng.

Lô(batch) trong Transact-SQL 3-5

Một số nguyên tắc được áp dụng khi sử dụng lô (batch):

- Các câu lệnh `CREATE FUNCTION`, `CREATE DEFAULT`, `CREATE RULE`, `CREATE TRIGGER`, `CREATE PROCEDURE`, `CREATE VIEW`, và `CREATE SCHEMA` không thể sử dụng cùng các câu lệnh khác trong một lô (batch).
- Câu lệnh `CREATE SQL` bắt đầu lô (batch) và toàn bộ các lệnh khác bên trong lô sẽ được xem như là một phần định nghĩa câu lệnh `CREATE`.
- Không có các thay đổi được thực hiện trong bảng và các cột mới tham khảo cùng một lô.
- Nếu câu lệnh đầu tiên trong lô là lệnh `EXECUTE`, sau đó, thì từ khóa `EXECUTE` là không cần thiết.

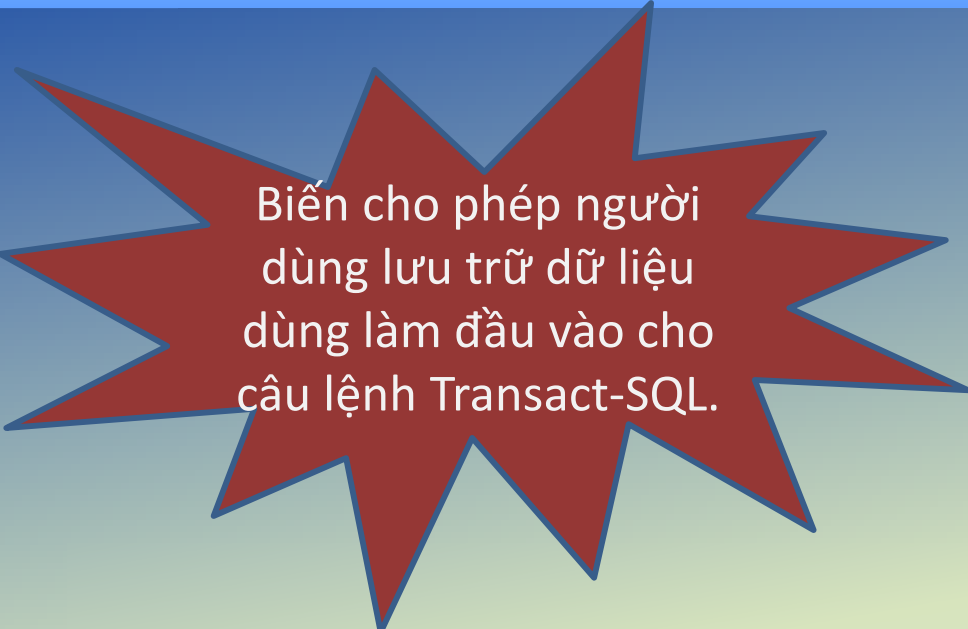


Lô(batch) trong Transact-SQL 4-5

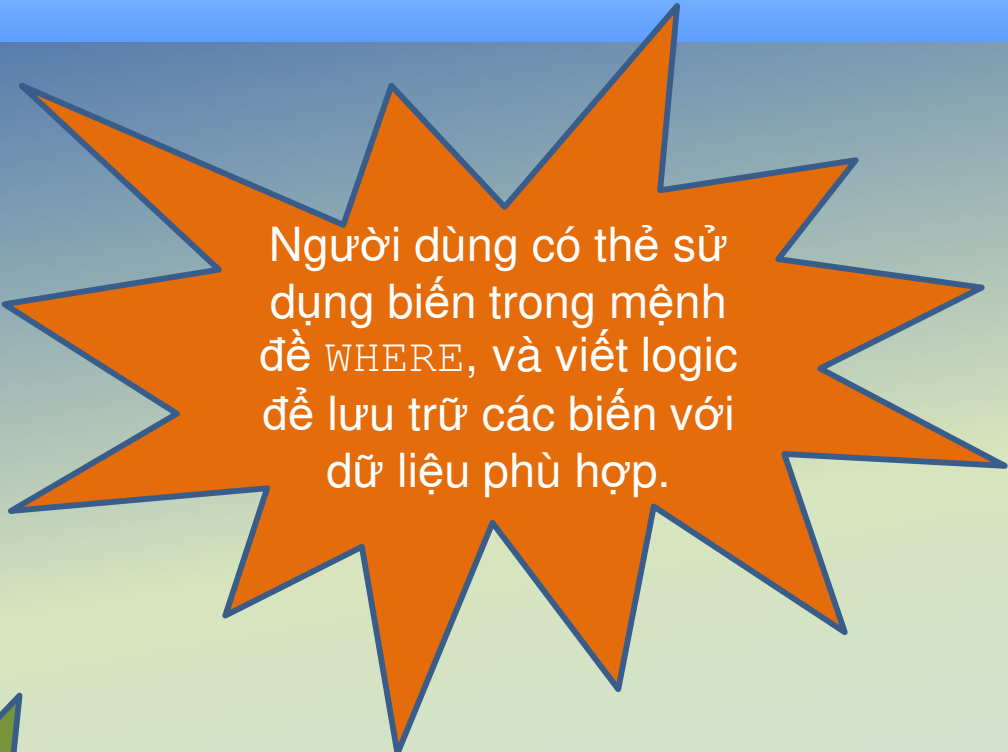
➤ Đoạn mã dưới đây cho thấy cách tạo một lô (batch):

```
BEGIN TRANSACTION
GO
USE AdventureWorks2012;
GO
CREATE TABLE Company
(   Id_Num int IDENTITY(100, 5),
    Company_Name nvarchar(100)
)
GO
INSERT Company (Company_Name) VALUES (N'A Bike Store')
INSERT Company (Company_Name) VALUES (N'Progressive Sports')
INSERT Company (Company_Name) VALUES (N'Modular Cycle Systems')
INSERT Company (Company_Name) VALUES (N'Advanced Bike Components')
INSERT Company (Company_Name) VALUES (N'Metropolitan Sports Supply')
INSERT Company (Company_Name) VALUES (N'Aerobic Exercise Company')
INSERT Company (Company_Name) VALUES (N'Associated Bikes')
INSERT Company (Company_Name) VALUES (N'Exemplary Cycles')
GO
SELECT Id_Num, Company_Name
FROM dbo. Company
ORDER BY Company_Name ASC;
GO
COMMIT;
GO
```

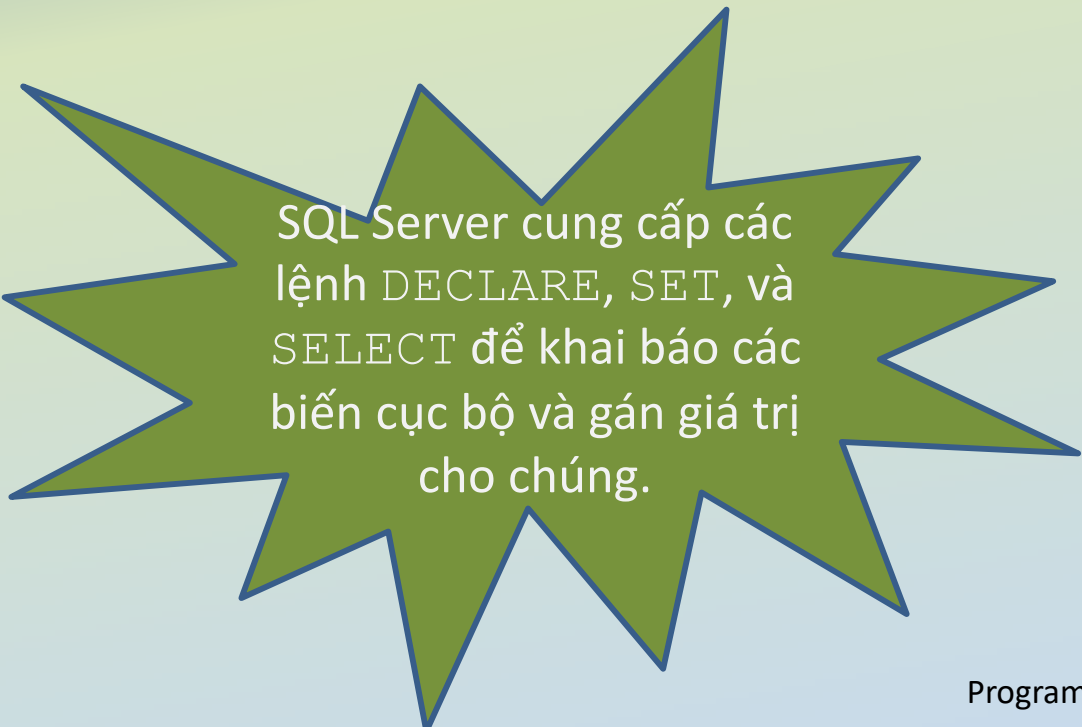
Biến trong Transact-SQL 1-8



Biến cho phép người dùng lưu trữ dữ liệu dùng làm đầu vào cho câu lệnh Transact-SQL.



Người dùng có thể sử dụng biến trong mệnh đề `WHERE`, và viết logic để lưu trữ các biến với dữ liệu phù hợp.



SQL Server cung cấp các lệnh `DECLARE`, `SET`, và `SELECT` để khai báo các biến cục bộ và gán giá trị cho chúng.

Biến trong Transact-SQL 2-8

DECLARE: Từ khóa được sử dụng để khai báo và khởi tạo giá trị ban đầu cho biến là NULL nếu không cung cấp giá trị cho nó. Các biến được gán giá trị bằng câu lệnh SELECT hoặc SET.

Cú pháp:

```
DECLARE { { @local_variable [AS] data_type } | [ = value ] }
```

trong đó,

@local_variable: chỉ ra tên của biến cục bộ được bắt đầu bằng kí hiệu @.

data_type: chỉ ra kiểu dữ liệu cho biến. Không sử dụng kiểu dữ liệu image, text, hoặc ntext.

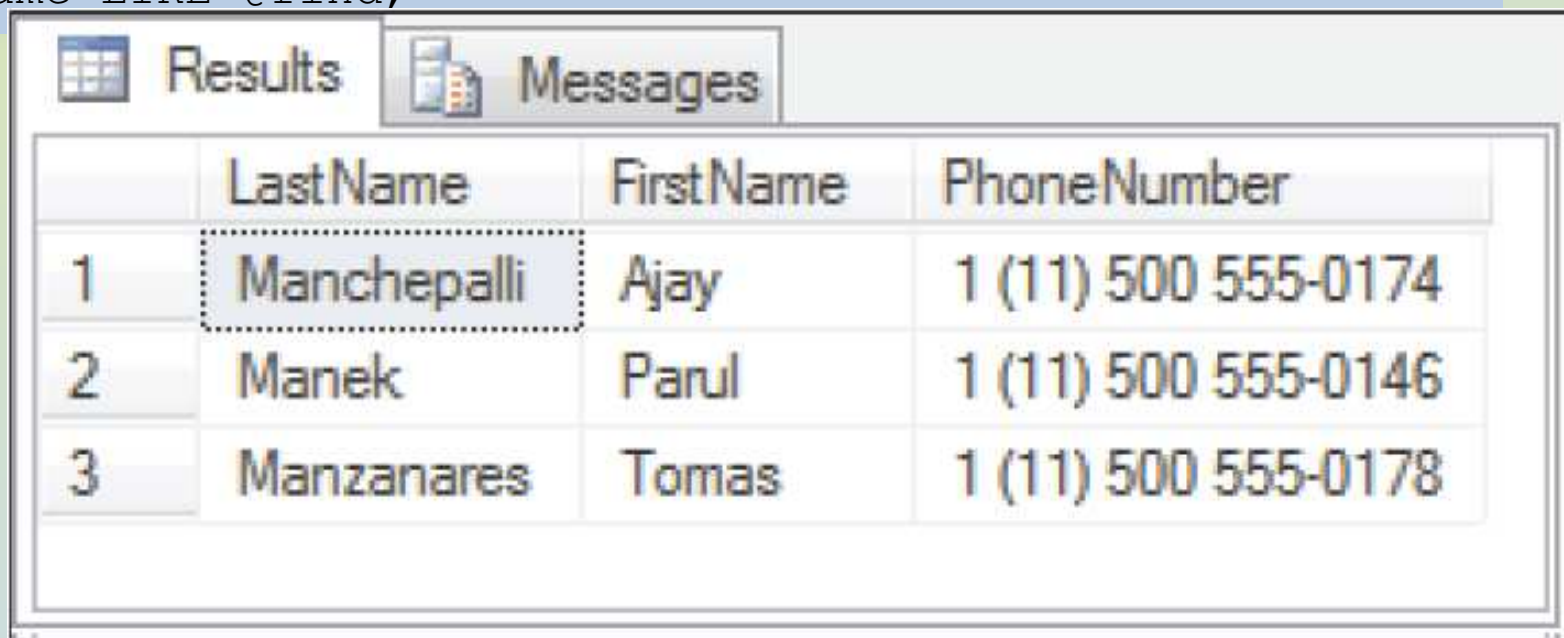
= value: gán giá trị value cho biến. Value có thể là một biểu thức hoặc hằng giá trị. Giá trị lên khớp với kiểu khai báo cho biến hoặc nên chuyển kiểu tương minh.

Biến trong Transact-SQL 3-8

- Đoạn code sau đây cho thấy việc sử dụng biến cục bộ để lấy thông tin liên lạc của các Person có last name bắt đầu là **Man**:

```
USE AdventureWorks2012;  
GO  
DECLARE @find varchar(30) = 'Man%';  
SELECT p.LastName, p.FirstName, ph.PhoneNumber  
FROM Person.Person AS p  
JOIN Person.PersonPhone AS ph ON p.BusinessEntityID =  
    ph.BusinessEntityID  
WHERE LastName LIKE @find;
```

Kết quả:



The screenshot shows the 'Results' tab in SQL Server Enterprise Manager. It displays a table with four columns: 'ID' (implied), 'LastName', 'FirstName', and 'PhoneNumber'. The table contains three rows of data, all with last names starting with 'Man'. The first row is highlighted with a dashed border.

	LastName	FirstName	PhoneNumber
1	Manchepalli	Ajay	1 (11) 500 555-0174
2	Manek	Parul	1 (11) 500 555-0146
3	Manzanares	Tomas	1 (11) 500 555-0178

Biến trong Transact-SQL 4-8

SET: câu lệnh dùng để gán giá trị cho biến được tạo bởi lệnh DECLARE.

Cú pháp:

```
SET
{ @local_variable = { expression}
}
|
{ @local_variable
{+= | -= | *= | /= | %= | &= | ^= | |= } expression
}
```

trong đó,

@local_variable: chỉ ra tên của biến được gán giá trị.

=: gán giá trị phía bên phải cho biến ở phía bên trái.

{= | += | -= | *= | /= | %= | &= | ^= | |= }: chỉ ra các toán tử kết hợp với phép gán.

expression: chỉ ra một biểu thức hợp lệ bất kỳ, thậm chí có thể là một truy vấn con tính toán trả về một giá trị.



Biến trong Transact-SQL 5-8

- Đoạn code dưới đây minh họa sử dụng SET để gán một giá trị chuỗi(string) cho một biến.

```
DECLARE @myvar char(20);  
SET @myvar = 'This is a test';
```

Biến trong Transact-SQL 6-8

SELECT: câu lệnh chỉ được sử dụng để gán một giá trị hoặc một biểu thức cho biến cục bộ đã được khai báo bằng lệnh DECLARE .

Cú pháp:

```
SELECT { @local_variable { = | += | -= | *= | /= | %= | &=
      | ^= | |= } expression } [ , ...n ] [ ; ]
```

trong đó,

@local_variable: chỉ ra tên của biến được gán giá trị.

=: gán giá trị phía bên phải cho biến ở phía bên trái.

{ = | += | -= | *= | /= | %= | &= | ^= | |= } : chỉ ra các toán tử kết hợp với phép gán.

expression: chỉ ra một biểu thức hợp lệ bất kỳ, thậm chí có thể là một truy vấn con tính toán trả về một giá trị.

Biến trong Transact-SQL 7-8

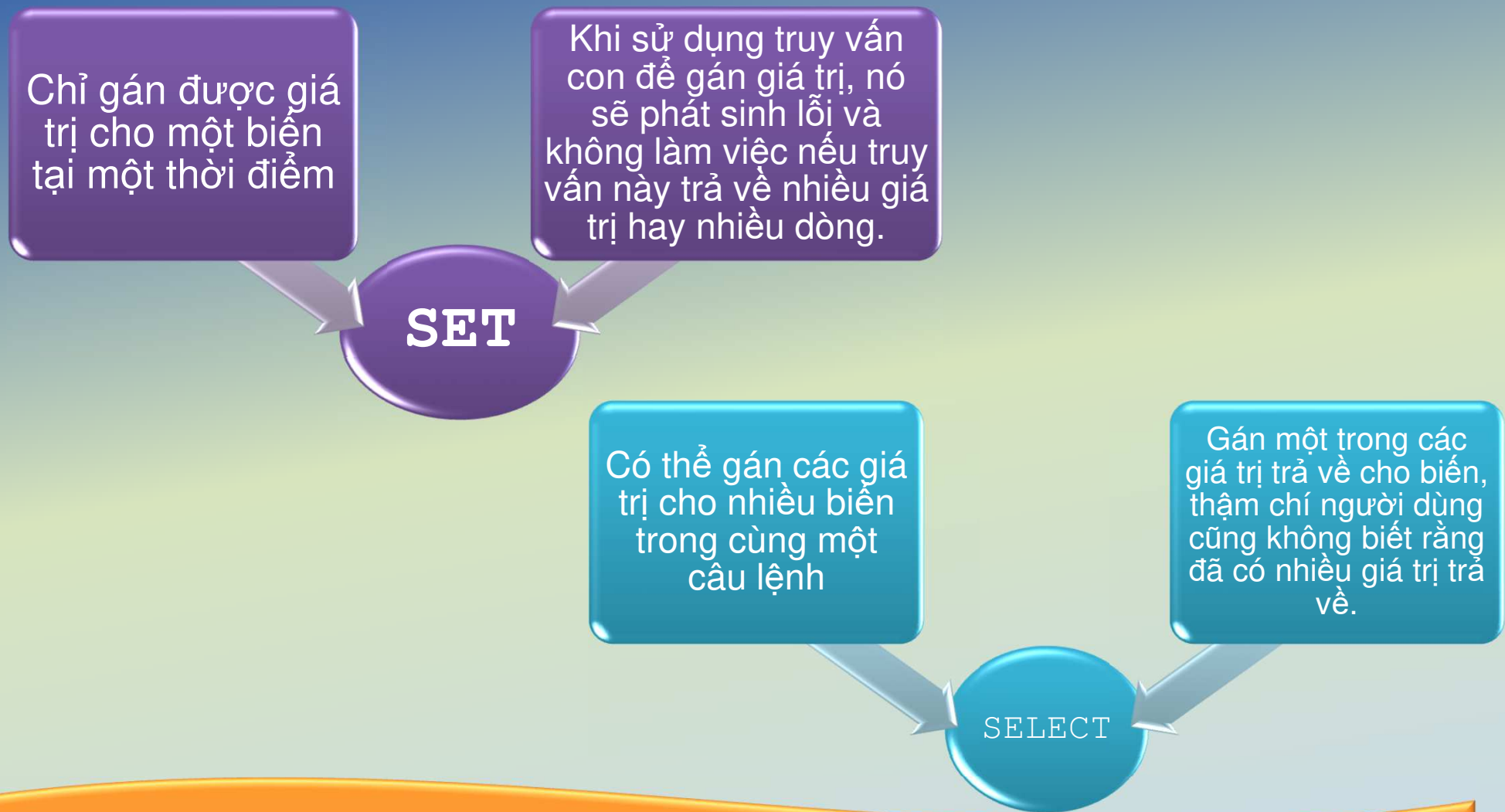
- Đoạn code dưới đây minh họa sử dụng câu lệnh `SELECT` để trả về một giá trị:

```
USE AdventureWorks2012 ;  
GO  
DECLARE @var1 nvarchar(30);  
SELECT @var1 = 'Unnamed Company';  
SELECT @var1 = Name  
FROM Sales.Store  
WHERE BusinessEntityID = 10;  
SELECT @var1 AS 'Company Name';
```

Kết quả:

	Company Name
1	Unnamed Company

Biến trong Transact-SQL 8-8



Để gán giá trị cho các biến, khuyên nên (recommended) sử dụng `SET @local_variable` thay cho `SELECT @local_variable`

Synonyms(đồng nghĩa/bí danh) 1-6

Một synonym (đồng nghĩa) là một đối tượng cơ sở dữ liệu phục vụ các mục đích sau:

- Cung cấp một cái tên thay thế(alias) cho một đối tượng nằm trong cơ sở dữ liệu khác, được xem như là đối tượng cơ sở, mà có thể tồn tại trên một máy chủ cục bộ hoặc từ xa.
- Cung cấp một lớp trừu tượng(layer of abstraction) để bảo vệ ứng dụng client khỏi sự thay đổi tên hoặc vị trí của các đối tượng cơ sở.

Một synonym thuộc về một lược đồ(schema), và giống như các đối tượng khác trong một schema, tên của một synonym phải là duy nhất

➤ Bạn có thể tạo các synonym cho các đối tượng cơ sở dữ liệu sau đây:

Database Objects
Extended stored procedure
SQL table-valued function
SQL stored procedure
Table(User-defined)
Replication-filter-procedure
SQL scalar function
SQL inline-tabled-valued function
View

Synonyms 2-6

➤ Synonyms và Schemas

Nếu người dùng có một schema mặc định mà người dùng không sở hữu và muốn tạo ra một synonym, bạn phải hội đủ điều kiện tên synonym với tên của một schema mà bạn sở hữu.

Ví dụ nếu bạn sở hữu một lược đồ **x**, nhưng **y** là lược đồ mặc định của bạn và bạn sử dụng câu lệnh CREATE SYNONYM bạn phải thêm tên tiền tố tên của synonym với lược đồ **x**, thay vì đặt tên cho các synonym bằng cách sử dụng một tên có một phần duy nhất.

➤ Gán quyền hạn trên các Synonym

Chỉ có các thành viên của roles `db_owner` hoặc `db_ddladmin` hoặc chủ sở hữu synonym là được phép gán các quyền hạn cho synonym.

Người dùng có thể cấm(deny), cấp(grant), hoặc thu hồi (revoke) tất cả hoặc bất kỳ quyền hạn nào trên synonym.

Synonyms 3-6

➤ Làm việc với Synonym

Có thể làm việc với synonym trong SQL Server 2012 bằng Transact-SQL hoặc SSMS.

Để tạo một synonym bằng SSMS, thực các bước sau đây:

- 1) Trong Object Explorer, bấm dấu + của csdl bạn muốn tạo một synonym mới
- 2) Bấm chuột phải trên folder **Synonyms** , sau đó click **New Synonym...**
- 3) Trong hộp thoại **New Synonym**, cung cấp thông tin sau:

Synonym name: là một tên mới cho đối tượng.

Synonym schema: là tên mới cho đối tượng schema.

Server name: là tên của server được kết nối đến.

Database name: là tên csdl để kết nối đối tượng.

Schema: là schema sở hữu đối tượng.

Synonyms 4-6

Để tạo synonym bằng Transact-SQL, hãy thực hiện theo các bước sau đây:

- 1) Kết nối đến Database Engine.
- 2) Click **New Query** trong thanh Standard.
- 3) Viết truy vấn để tạo synonym trong cửa sổ truy vấn.
- 4) Click **Execute** trên toolbar để hoàn thành việc tạo synonym.

Synonyms 5-6

Cú pháp:

```
CREATE SYNONYM [ schema_name_1. ] synonym_name FOR <object>
<object> :: =
{
[ server_name.[ database_name ] . [ schema_name_2 ] .|
  database_name . [ schema_name_2 ] .| schema_name_2. ]
  object_name
}
```

trong đó,

schema_name_1: chỉ ra schema mà synonym được tạo trong đó.

synonym_name: chỉ ra tên mới cho synonym.

server_name: chỉ ra tên server nơi mà đối tượng cơ sở đã được đặt ở đó(located).

database_name: chỉ ra tên csdl nơi mà đối tượng cơ sở đã được đặt ở đó(located).

schema_name_2: chỉ ra tên lược đồ của đối tượng cơ sở.

object_name: chỉ ra tên đối tượng cơ sở, được tham chiếu bởi synonym.



Synonyms 6-6

- Đoạn code dưới đây minh họa tạo synonym từ bảng hiện có:

```
USE tempdb;  
GO  
CREATE SYNONYM MyAddressType  
FOR AdventureWorks2012.Person.AddressType;  
GO
```

Các lệnh luồng chương trình 1-8

Sau đây là các loại câu lệnh điều khiển luồng chương trình được hỗ trợ trong Transact-SQL are:

➤ Ngôn ngữ điều khiển luồng Transact-SQL

Quyết định luồng thực thi của các câu lệnh Transact-SQL, các khối lệnh, các hàm người dùng định nghĩa và thủ tục lưu.

Các lệnh Transact-SQL được thực thi tuần tự theo thứ tự chúng xuất hiện.

Cho phép các câu lệnh được thực thi Allow statements to be executed in a particular order, to be related to each other, and made interdependent using constructs similar to programming languages.

Các lệnh luồng chương trình 2-8

- Bảng sau đây liệt kê một số từ khóa của ngôn ngữ điều khiển luồng Transact-SQL:

Control-Of-Flow Language Keywords
RETURN
THROW
TRY....CATCH
WAITFOR
WHILE
BEGIN....END
BREAK
CONTINUE
GOTO label
IF...ELSE

Các lệnh luồng chương trình 3-8

Câu lệnh BEGIN...END bao quanh một dãy các câu lệnh Transact-SQL với mục đích tạo một nhóm các lệnh Transact-SQL để thực thi.

Cú pháp:

```
BEGIN
{
  sql_statement | statement_block
}
END
```

trong đó,

{sql_statement | statement_block}: là một câu lệnh Transact-SQL hợp lệ bất kỳ được định nghĩa bằng một khối câu lệnh.

Các lệnh luồng chương trình 4-8

➤ Đoạn code sau đây cho thấy sử dụng các lệnh BEGIN và END :

```
USE AdventureWorks2012;
GO
BEGIN TRANSACTION;
GO
IF @@TRANCOUNT = 0
BEGIN
SELECT FirstName, MiddleName
FROM Person.Person WHERE LastName = 'Andy';
ROLLBACK TRANSACTION;
PRINT N'Rolling back the transaction two times would cause an error.';
END;
ROLLBACK TRANSACTION;
PRINT N'Rolled back the transaction.';
GO
```

Các lệnh luồng chương trình 5-8

Câu lệnh `IF...ELSE` áp đặt(enforces) một điều kiện về việc thực thi một câu lệnh Transact-SQL.

Câu lệnh Transact-SQL theo sau từ khóa `IF` và điều kiện chỉ được thực thi nếu điều kiện được thỏa mãn và trả về `TRUE`.

Từ khóa `ELSE` là câu lệnh Transact-SQL tùy chọn, chỉ được thực thi khi điều kiện của `IF` không thỏa mãn và trả về `FALSE`.

Cú pháp:

```
IF Boolean_expression
{ sql_statement | statement_block }
[ ELSE
{ sql_statement | statement_block } ]
```

trong đó,

`Boolean_expression`: chỉ ra một biểu thức trả về giá trị `TRUE` hoặc `FALSE`

Các lệnh luồng chương trình 6-8

{sql_statement | statement_block}: là câu lệnh Transact-SQL hợp lệ bất kỳ được định bằng việc sử dụng một khối lệnh.

- Đoạn code dưới đây cho thấy việc sử dụng câu lệnh IF...ELSE

```
USE AdventureWorks2012
GO
DECLARE @ListPrice money;
SET @ListPrice = (SELECT MAX(p.ListPrice)
    FROM Production.Product AS p
    JOIN Production.ProductSubcategory AS s
    ON p.ProductSubcategoryID = s.ProductSubcategoryID
    WHERE s.[Name] = 'Mountain Bikes');
    PRINT @ListPrice
IF @ListPrice <3000
PRINT 'All the products in this category can be purchased for an
    amount less than 3000'
ELSE
PRINT 'The prices for some products in this category exceed 3000'
```

Các lệnh luồng chương trình 7-8

Câu lệnh `WHILE` – chỉ ra một điều kiện cho việc thực hiện lặp đi lặp lại một khối lệnh.

Các câu lệnh được thực thi lặp đi lặp lại cho tới điều kiện chỉ ra là đúng.

Việc thực thi các câu lệnh trong vòng lặp `WHILE` có thể được điều khiển bằng việc dùng các từ khóa `BREAK` và `CONTINUE`.

Cú pháp:

```
WHILE Boolean_expression  
{ sql_statement | statement_block | BREAK | CONTINUE }
```

trong đó,

`Boolean_expression`: chỉ ra biểu thức trả về giá trị `TRUE` hoặc `FALSE`.

`{sql_statement| statement_block}`: là câu lệnh Transact-SQL hợp lệ bất kỳ được định bằng việc dùng một khối lệnh.

`BREAK`: thoát khỏi vòng lặp.

`CONTINUE`: chuyển sang lần lặp kế tiếp.

Các lệnh luồng chương trình 8-8

- Đoạn code sau đây cho thấy sử dụng lệnh WHILE:

```
DECLARE @flag int
SET @flag = 10
WHILE (@flag <=95)
    BEGIN
        IF @flag%2 =0
            PRINT @flag
        SET @flag = @flag + 1
        CONTINUE;
    END
GO
```

Các hàm Transact-SQL 1-5

Các hàm Transact-SQL được sử dụng phổ biến như sau:

➤ **Hàm xác định (deterministic) và không xác định (non-deterministic)**

- Hàm do người dùng định nghĩa có (possess) đầy đủ các tính chất (properties) định nghĩa khả năng của SQL Server Database Engine.
- Bộ máy csdl (Database engine) được sử dụng để đánh chỉ mục (index) kết quả của một hàm qua các cột tính toán (computed column) mà hàm gọi hoặc view có chỉ mục (indexed views) tham chiếu các hàm.
- Hàm xác định (deterministic) luôn trả về cùng một kết quả khi được gọi ở các thời điểm nhau với bộ các giá trị đầu vào như nhau.
- Hàm không xác định (nondeterministic) có thể trả về kết quả khác nhau khi chúng được gọi ở các thời điểm khác nhau với cùng bộ giá trị đầu vào.
- Các hàm dựng sẵn (built-in) là hàm xác định hay không xác định, nó phụ thuộc vào hàm được triển khai (implemented) bởi SQL Server.

Các hàm Transact-SQL 2-5

- Bảng dưới đây liệt kê một số hàm xác định, không xác định có sẵn:

Deterministic Built-in Functions	Non-Deterministic Built-in Functions
POWER	@@TOTAL_WRITE
ROUND	CURRENT_TIMESTAMP
RADIANS	GETDATE
EXP	GETUTCDATE
FLOOR	GET_TRANSMISSION_STATUS
SQUARE	NEWID
SQRT	NEWSEQUENTIALID
LOG	@@CONNECTIONS
YEAR	@@CPU_BUSY
ABS	@@DBTS
ASIN	@@IDLE
ACOS	@@IOBUSY
SIGN	@@PACK_RECEIVED

Các hàm Transact-SQL 3-5

- Bảng dưới đây liệt kê một số hàm luôn luôn là hàm không xác định nhưng bạn có thể sử dụng chúng trong view được có nếu chúng được đo kiểu xác định:

Function	Description
CONVERT	<p>Is deterministic only if one of these conditions exists:</p> <ul style="list-style-type: none"> ➔ Has an <code>sql_variant</code> source type. ➔ Has an <code>sql_variant</code> target type and source type is non-deterministic. ➔ Has its source or target type as <code>smalldatetime</code> or <code>datetime</code>, has the other source or target type as a character string, and has a non-deterministic style specified. The style parameter must be a constant to be deterministic.
CAST	Is deterministic only if it is used with <code>smalldatetime</code> , <code>sql_variant</code> , or <code>datetime</code> .
ISDATE	Is deterministic unless used with the <code>CONVERT</code> function, the <code>CONVERT</code> style parameter is specified, and style is not equal to 0, 100, 9, or 109.
CHECKSUM	Is deterministic, with the exception of <code>CHECKSUM(*)</code> .

Các hàm Transact-SQL 4-5

➤ Gọi thủ tục lưu mở rộng (Extended Stored Procedures) từ hàm

- Hàm gọi thủ tục lưu mở rộng là hàm không xác định, vì thủ tục mở rộng có thể có kết quả phụ (side effects) trên csdl.
- Trong khi thực thi thủ tục lưu mở rộng từ hàm do người dùng định nghĩa, người dùng không thể đảm bảo rằng nó sẽ trả về tập kết quả phù hợp.
- Therefore, the user-defined functions that create side effects on the database are not recommended.

➤ Hàm trả về giá trị kiểu vô hướng (Scalar-Valued Functions)

- Hàm Scalar-Valued (SVF) là hàm đơn trị. Hàm chỉ trả về một giá trị kiểu `int`, `bit`, hoặc `string`.
- Kiểu dữ liệu trả về và kiểu dữ liệu của tham số đầu vào của hàm đơn trị (SVF) có thể là kiểu dữ liệu bất kỳ ngoại trừ `text`, `ntext`, `image`, `cursor`, và `timestamp`.
- Hàm inline scalar là hàm chỉ có duy nhất một câu lệnh và không có phần thân.
- Hàm multi-statement scalar là hàm có phần thân được bao trong khối `BEGIN...END`.

Các hàm Transact-SQL 5-5

➤ Hàm Table-Valued (TVF)

- Là các hàm do người dùng định nghĩa (user-defined) trả về một bảng.
- Tương tự như hàm vô hướng inline (inline scalar function), một hàm table-valued inline có một câu lệnh đơn và không có thân hàm.

➤ Đoạn code sau cho thấy việc tạo một hàm table-valued:

```
USE AdventureWorks2012;
GO
IF OBJECT_ID (N'Sales.ufn_CustDates', N'IF') IS NOT NULL
    DROP FUNCTION Sales.ufn_CustDates;
GO
CREATE FUNCTION Sales.ufn_CustDates ()
RETURNS TABLE
AS
RETURN
(
    SELECT A.CustomerID, B.DueDate, B.ShipDate
    FROM Sales.Customer A LEFT OUTER JOIN Sales.SalesOrderHeader B
    ON A.CustomerID = B.CustomerID AND YEAR(B.DueDate)<2012
);
```

Chỉnh sửa hàm người dùng định nghĩa 1-3

Hạn chế và Giới hạn

- ALTER FUNCTION không cho phép người dùng thực hiện các hành động sau:
 - Sửa từ một hàm trả về một giá trị (scalar-valued function) sang hàm trả về (table-valued function).
 - Sửa hàm nội tuyến (inline function) sang hàm multi-statement.
 - Sửa một hàm Transact-SQL sang hàm CLR function.

Quyền hạn

- Cần có quyền ALTER trên lược đồ hoặc hàm.
- Nếu hàm chỉ ra là hàm người dùng định nghĩa, thì đòi hỏi có quyền EXECUTE trên kiểu này.

Chỉnh sửa hàm người dùng định nghĩa 2-3

Sửa đổi hàm người dùng định nghĩa bằng SSMS

- Người dùng cũng có thể thay đổi hàm người dùng định nghĩa bằng cách sử dụng SSMS.
- Để sửa đổi bằng SSMS, thực hiện các bước sau:
 - Click biểu tượng cộng (+) bên cạnh database có chứa hàm đang cần chỉnh sửa.
 - Click biểu tượng cộng (+) kế tiếp của folder Programmability.
 - Click biểu tượng cộng (+) kế tiếp của folder có chứa hàm đang cần chỉnh sửa.
 - Chuột phải trên hàm cần sửa, sau đó chọn Modify. Nội dung code của hàm sẽ xuất hiện trong cửa sổ soạn thảo truy vấn.
 - Trong cửa sổ soạn thảo truy vấn, thực hiện các thay đổi theo yêu cầu trong phần thân của câu lệnh `ALTER FUNCTION`
 - Click nút Execute trên thanh công cụ để thực thi lệnh `ALTER FUNCTION`.

Chỉnh sửa hàm người dùng định nghĩa 3-3

Chỉnh sửa hàm do người dùng định nghĩa bằng lệnh Transact-SQL

- Để chỉnh sửa hàm do người dùng định nghĩa bằng lệnh Transact-SQL, thực hiện các bước sau đây:
 - Trong Object Explorer, kết nối đến thể hiện Database Engine.
 - Click nút New Query trên thanh Standard.
 - Gõ code lệnh ALTER FUNCTION trong cửa sổ soạn thảo truy vấn(Query Editor).
 - Click nút Execute trên thanh công cụ để thực thi lệnh ALTER FUNCTION.

➤ Đoạn code dưới đây minh họa chỉnh sửa hàm bằng lệnh T-SQL:

```
USE [AdventureWorks2012]
GO
ALTER FUNCTION [dbo].[ufnGetAccountingEndDate]()
RETURNS [datetime]
AS
BEGIN
RETURN DATEADD(second, -2, CONVERT(datetime, '20040701', 112));
END;
```


Tạo Window với mệnh đề OVER

Các hàm window là các hàm T-SQL có sẵn có thể áp dụng cho các dòng đã được phân hoạch(partitioned) của tập kết quả – windows – để xếp thứ hạng(rank) hoặc tổng hợp dữ liệu trong mỗi phân hoạch.

SQL Server 2012 gồm có ba loại hàm window: Xếp thứ hạng(ranking), tổng hợp(aggregate), phân tích(analytic)

- Các hàm ranking trả về một giá trị thứ hạng cho mỗi dòng trong một phân hoạch(partition)
- Các hàm tổng hợp thực hiện tính toán dựa trên các giá trị của cột trong mỗi phân hoạch, như tính tổng hay tính trung bình
- Các hàm phân tích tính toán các giá trị tổng hợp dựa trên các giá trị trong một cột bên trong một phân hoạch. Đây là hàm mới có trong SQL Server 2012

Mệnh đề OVER sẽ xác định cách phân hoạch và sắp xếp tập kết quả như thế nào để áp dụng các hàm window.

Các thành phần Windowing 1-3

- Có ba thành phần cốt lõi (core) của việc tạo với mệnh đề OVER như sau:

Partitioning – là một tính năng giới hạn window của việc tính toán hiện tại chỉ tới các dòng đó từ tập kết quả có chứa cùng giá trị trong các cột partition như là trong dòng hiện hành.

- Đoạn code dưới đây minh họa sử dụng PARTITION BY và mệnh đề OVER với các hàm tính trung bình:

```
USE AdventureWorks2012;
Go
SELECT SalesOrderID, ProductID, OrderQty
, SUM(OrderQty) OVER (PARTITION BY SalesOrderID) AS Total
, MAX(OrderQty) OVER (PARTITION BY SalesOrderID) AS
    MaxOrderQty
FROM Sales.SalesOrderDetail
WHERE ProductId IN (776, 773);
GO
```

Các thành phần Windowing 2-3

Ordering – thành phần định nghĩa thứ tự cho việc tính toán trong phân vùng (partition). Trong SQL Server 2012, có sự hỗ trợ cho thành phần sắp xếp với các hàm tính trung bình.

- Đoạn code sau đây minh họa một ví dụ của thành phần sắp xếp:

```
SELECT CustomerID, StoreID,  
RANK() OVER(ORDER BY StoreID DESC) AS Rnk_All,  
RANK() OVER(PARTITION BY PersonID  
ORDER BY CustomerID DESC) AS Rnk_Cust  
FROM Sales.Customer;
```

Kết quả:

	CustomerID	StoreID	Rnk_All	Rnk_Cust
1	701	844	813	1
2	700	1030	633	2
3	699	842	815	3
4	698	640	1009	4
5	697	1032	631	5
6	696	840	817	6
7	695	638	1011	7
8	694	1034	629	8
9	693	838	819	9
10	692	802	855	10
11	691	1036	627	11

Các thành phần Windowing 3-3

Framing - là một tính năng cho phép bạn tiếp tục chỉ thêm việc phân chia các hàng trong một phân vùng cửa sổ(window partition). một khung(frame) giống như một cửa sổ di chuyển qua các dữ liệu bắt đầu và kết thúc tại các vị trí được chỉ ra và được xác định bằng cách sử dụng ROW hoặc mệnh đề con Range.

- Đoạn code dưới đây trình bày một truy vấn với **ProductInventory**, tính tổng số lượng cho mỗi sản phẩm (product) và vị trí(location):

```
SELECT ProductID, Shelf, Quantity,  
SUM(Quantity) OVER(PARTITION BY ProductID  
ORDER BY LocationID  
ROWS BETWEEN UNBOUNDED PRECEDING  
AND CURRENT ROW) AS RunQty  
FROM Production.ProductInventory;
```

Kết quả

	ProductID	Shelf	Quantity	RunQty
1	1	A	408	408
2	1	B	324	732
3	1	A	353	1085
4	2	A	427	427
5	2	B	318	745
6	2	A	364	1109
7	3	A	585	585
8	3	B	443	1028
9	3	A	324	1352
10	4	A	512	512
11	4	B	422	934

Các hàm Windows 1-9

- Sau đây là một số loại hàm window khác nhau:

Các hàm Ranking – trả về giá trị thứ hạng cho mỗi dòng trong một phân vùng. Dựa vào hàm được sử dụng, nhiều dòng sẽ có cùng giá trị thứ hạng như các dòng khác.

- Bảng liệt kê các hàm xếp thứ hạng khác nhau:

Ranking Functions	Description
NTILE	Spreads rows in an ordered partition into a given number of groups, beginning at 1. For each row, the function returns the number of the group to which the row belongs.
ROW NUMBER	Retrieves the sequential number of a row in a partition of a resultset, starting at 1 for the first row in each partition.
DENSE RANK	Returns the rank of rows within the partition of a resultset, without any gaps in the ranking. The rank of a row is one plus the number of distinct ranks that come before the row in question.

Các hàm Windows 2-9

- Đoạn code sau đây minh họa sử dụng các hàm xếp hạng (ranking) :

```
USE AdventureWorks2012;
GO
SELECT p.FirstName, p.LastName
,ROW_NUMBER() OVER (ORDER BY a.PostalCode) AS 'Row Number'
,NTILE(4) OVER (ORDER BY a.PostalCode) AS 'NTILE'
,s.SalesYTD, a.PostalCode
FROM Sales.SalesPerson AS s
INNER JOIN Person.Person AS p
ON s.BusinessEntityID = p.BusinessEntityID
INNER JOIN Person.Address AS a
ON a.AddressID = p.BusinessEntityID
WHERE TerritoryID IS NOT NULL
AND SalesYTD <> 0;
```

Các hàm Windows 3-9

Hàm OFFSET – Dưới đây là các loại hàm offset khác nhau:

- **SWITCHOFFSET** – trả về một giá trị DATETIMEOFFSET được thay đổi từ khoảng cách(offset) múi giờ(time zone) được lưu trữ tới múi giờ mới được chỉ ra.

Cú pháp:

```
SWITCHOFFSET ( DATETIMEOFFSET, time_zone )
```

trong đó,

DATETIMEOFFSET: là biểu thức được giải quyết tới một giá trị `datetimeoffset(n)`.

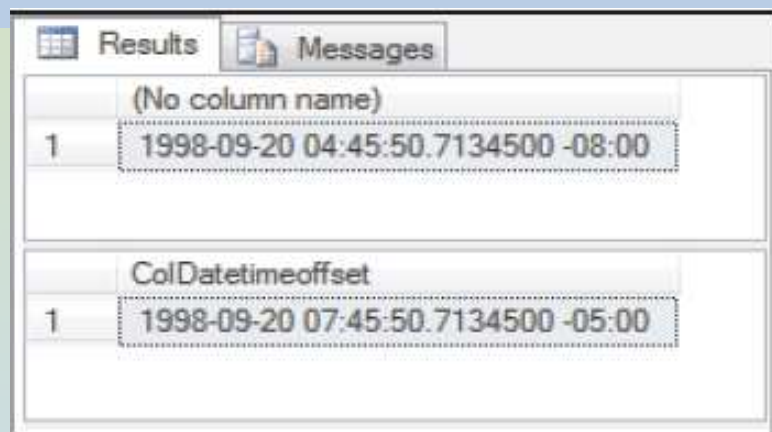
time_zone: chỉ ra chuỗi kí tự theo dạng `[+|-] T Z H : T Z M` hoặc một số nguyên được kí hiệu (của phút) biểu diễn cho khoảng cách múi giờ(time zone offset) và được giả định là ban ngày và điều chỉnh.

Các hàm Windows 4-9

- Đoạn code minh họa sử dụng hàm SWITCHOFFSET:

```
CREATE TABLE Test
(
    ColDatetimeoffset datetimeoffset
);
GO
INSERT INTO Test VALUES ('1998-09-20 7:45:50.71345 -5:00');
GO
SELECT SWITCHOFFSET (ColDatetimeoffset, '-08:00')
FROM Test;
GO
--Returns: 1998-09-20 04:45:50.7134500 -08:00
SELECT ColDatetimeoffset
FROM Test;
```

Kết quả:



The screenshot shows the SQL Server Results window with two tabs: 'Results' and 'Messages'. The 'Results' tab is active and displays two tables. The first table, titled '(No column name)', has one row with the value '1998-09-20 04:45:50.7134500 -08:00'. The second table, titled 'ColDatetimeoffset', has one row with the value '1998-09-20 07:45:50.7134500 -05:00'.

(No column name)	
1	1998-09-20 04:45:50.7134500 -08:00

ColDatetimeoffset	
1	1998-09-20 07:45:50.7134500 -05:00

Các hàm Windows 5-9

- DATETIMEOFFSETFROMPARTS – trả về giá trị `datetimeoffset` cho ngày và thời gian được chỉ ra cùng với độ chính xác(precision) và độ lệch (offset).

Cú pháp:

```
DATETIMEOFFSETFROMPARTS ( year, month, day, hour,  
minute, seconds, fractions, hour_offset,  
minute_offset, precision )
```

Trong đó,

`year`: chỉ ra một biểu thức nguyên cho năm.

`month`: chỉ ra một biểu thức nguyên cho tháng.

`day`: chỉ ra một biểu thức nguyên cho ngày.

`hour`: chỉ ra một biểu thức nguyên cho giờ.

`minute`: chỉ ra một biểu thức nguyên cho phút.

`seconds`: chỉ ra một biểu thức nguyên cho giây.

`fractions`: chỉ ra một biểu thức nguyên cho phần lẻ (fractions).

`hour_offset`: specifies the integer expression for the hour portion of the time zone offset.

`minute_offset`: specifies the integer expression for the minute portion of the time zone offset.



`precision`: specifies the integer literal precision of the `datetimeoffset` value to be returned.

Các hàm Windows 6-9

- Đoạn code minh họa sử dụng hàm DATETIMEOFFSETFROMPARTS:

```
SELECT DATETIMEOFFSETFROMPARTS ( 2010, 12, 31, 14, 23, 23, 0, 12, 0,  
7 ) AS Result;
```

Kết quả:

 Results	 Messages
Result	
1	2010-12-31 14:23:23.0000000 +12:00

Các hàm Windows 7-9

- SYSDATETIMEOFFSET – trả về giá trị `datetimeoffset (7)` có chứa ngày tháng và thời gian của máy tính mà thể hiện SQL Server đang chạy trên đó.

Cú pháp:

```
SYSDATETIMEOFFSET ( )
```

- Đoạn code dưới đây minh họa dùng hàm `SYSDATETIMEOFFSET ()` với các định dạng thời gian khác nhau:

```
SELECT SYSDATETIME ( ) AS SYSDATETIME
, SYSDATETIMEOFFSET ( ) AS SYSDATETIMEOFFSET
, SYSUTCDATETIME ( ) AS SYSUTCDATETIME
```

Cú pháp:

	SYSDATETIME	SYSDATETIMEOFFSET	SYSUTCDATETIME
1	2013-02-08 16:08:16.6565247	2013-02-08 16:08:16.6565247 +05:30	2013-02-08 10:38:16.6565247

Các hàm Windows 8-9

Các hàm phân tích (Analytic) - tính toán tổng giá trị dựa trên một nhóm các hàng. Analytic functions compute running totals, moving averages, or top-N results within a group.

➤ Following table lists the various analytic functions:

Function	Description
LEAD	Provides access to data from a subsequent row in the same resultset without using a self-join.
LAST_VALUE	Retrieves the last value in an ordered set of values.
LAG	Provides access to data from a previous row in the same resultset without using a self-join.
FIRST_VALUE	Retrieves the first value in an ordered set of values.
CUME_DIST	Computes the cumulative distribution of a value in a group of values.
PERCENTILE_CONT	Computes a percentile based on a continuous distribution of the column value in SQL.
PERCENTILE_DISC	Calculates a particular percentile for sorted values in an entire rowset or within distinct partitions of a rowset.

Các hàm Windows 9-9

- Đoạn code dưới đây minh họa sử dụng hàm LEAD () :

```
USE AdventureWorks2012;
GO
SELECT BusinessEntityID, YEAR(QuotaDate) AS QuotaYear, SalesQuota AS
    NewQuota,
    LEAD(SalesQuota, 1,0) OVER (ORDER BY YEAR(QuotaDate)) AS FutureQuota
FROM Sales.SalesPersonQuotaHistory
WHERE BusinessEntityID = 275 and YEAR(QuotaDate) IN ('2007','2008');
```

- Đoạn code dưới đây minh họa sử dụng hàm FIRST_VALUE () :

```
USE AdventureWorks2012;
GO
SELECT Name, ListPrice,
    FIRST_VALUE(Name) OVER (ORDER BY ListPrice ASC) AS
    LessExpensive
FROM Production.Product
WHERE ProductSubcategoryID = 37
```



Tóm tắt

- Transact-SQL cung cấp các thành phần lập trình cơ bản như biến, các thành phần điều khiển luồng, cấu trúc lặp và điều kiện.
- Một lô(batch) là một tập hợp của một hay nhiều câu lệnh Transact-SQL được gửi như là khối(unit) từ ứng dụng tới server.
- Biến cho phép người dùng lưu trữ dữ liệu và được dùng làm đầu vào của các câu lệnh Transact-SQL khác.
- Synonyms provide a way to have an alias for a database object that may exist on a remote or local server.
- Deterministic functions each time return the same result every time they are called with a definite set of input values and specify the same state of the database.
- Non-deterministic functions return different results every time they are called with specified set of input values even though the database that is accessed remains the same.
- A window function is a function that applies to a collection of rows.