Chương 6

Hàm – Function

Nội dung trình bày

1. Giới thiệu
2. Hàm hệ thống
3. Hàm tự định nghĩa

1. Giới thiệu

1. Giới thiệu

1.1 Giới thiệu Hàm (Function): là đối tượng cơ sở dữ liệu tương tự như **thủ tục lưu trữ** (Store procedure). Điểm khác biệt giữa **hàm** và **thủ tục lưu trữ** là **hàm trả về một giá trị thông qua tên hàm** còn **thủ tục lưu trữ** thì không. Điều này cho phép ta sử dụng **hàm** như là một thành phần của một biểu thức (chẳng hạn trong danh sách chọn của câu lệnh SELECT).

Ngoài những **hàm** *do hệ quản trị cơ sở dữ liệu cung cấp sẵn*, người sử dụng có thể định nghĩa thêm các hàm nhằm phục vụ cho mục đích riêng của mình

1. Giới thiệu

1.2 Định nghĩa Hàm (Function):

- Là một đối tượng trong cơ sở dữ liệu (CSDL) bao gồm một tập nhiều câu lệnh được nhóm lại với nhau và được tạo ra với mục đích sử dụng lại
- Được biên dịch sẵn và lưu trong CSDL nhằm mục đích thực hiện xử lý nào đó như tính toán phức tạp và trả về kết quả là giá trị nào đó.

Đặc điểm:

•Luôn trả về giá trị

•Gồm 2 loại: **Function hệ thống** và **Function do người dùng tự định nghĩa**

•Function người dùng tự định nghĩa gồm 2 loại:

- **Scalar-valued**: Trả về giá trị vô hướng của các kiểu dữ liệu T-SQL
- **Table-valued**: Trả về bảng, là kết quả của một hoặc nhiều lệnh

2. Hàm hệ thống

2. Hàm hệ thống

Sql Server cung cấp cho người dùng các hàm thường xuyên sử dụng trong quá trình làm việc.

2.1. Các hàm xử lý toán học:

•**COUNT**: Là hàm đếm số dòng kết quả câu query trả về, thường sử dụng chung với **GROUP BY**

```
SELECT COUNT(*) AS NUMBER FROM NhacTruong
```

	NUMBER
1	10

2. Hàm hệ thống

- **SUM**: Là hàm tính tổng giá trị của cột được truyền vào, thường sử dụng chung với **GROUP BY**

  Ví dụ: Tính tổng lương của nhân vien

```
SELECT SUM(Luong) AS LUONG FROM NhanVien
```

	LUONG
1	21900

- **AVG**: Là hàm tính giá trị trung bình cột được truyền vào, thường sử dụng chung với **GROUP BY**

  Ví dụ: Tính mức lương trung bình của nhân viên

```
SELECT AVG(Luong) AS LUONG FROM NhanVien
```

	LUONG TRUNG BINH
1	2190

2. Hàm hệ thống

- **MIN & MAX**: Min là hàm tìm giá trị nhỏ nhất trên cột truyền vào và Max là hàm tìm giá trị lớn nhất trên cột truyền vào, thường sử dụng chung với **GROUP BY**

Ví dụ: Tính mức lương trung bình của nhân viên

```
SELECT Max(Luong) AS 'LUONG CAO NHAT' FROM NhanVien;
SELECT Min(Luong) AS 'LUONG THAP NHAT' FROM NhanVien;
```

	LUONG CAO NHAT
1	3500

	LUONG THAP NHAT
1	1500

- **ROUND**: Trả về 1 số được làm tròn với số thập phân được chỉ định

Cú pháp: **ROUND**(column_name, decimals)

column_name	Tham số bắt buộc, là trường sẽ được làm tròn.
decimals	Tham số bắt buộc, chỉ ra số số lượng chữ số phần thập phân được trả về (được làm tròn).

```
SELECT COUNT(3.14, 0) as GiaTri;    -- 3
SELECT COUNT(3.14, 1) as GiaTri;    -- 3.1
SELECT COUNT(3.15, 1) as GiaTri;    -- 3.2
```

2. Hàm hệ thống

- **ABS**: Hàm trả về giá trị tuyệt đối.

  Cú pháp: **ABS**(Number)

```
SELECT Abs(-243.5) AS AbsNum;
--Kết quả: 243.5
```

- **CEILING**: Hàm trả về giá trị cận trên của số hoặc biểu thức, nghĩa là trả về số nguyên nhỏ nhất nhưng lớn hơn giá trị truyền vào.

  Cú pháp: **CEILING**(Number)

```
SELECT Ceiling(243.23) AS AbsNum;
--Kết quả: 244
```

- **FLOOR**: Hàm trả về giá trị cận dưới của số hoặc biểu thức, nghĩa là trả về số nguyên lớn nhất nhưng nhỏ hơn giá trị truyền vào.

  Cú pháp: **FLOOR**(Number)

```
SELECT Floor(243.23) AS AbsNum;
--Kết quả: 243
```

2. Hàm hệ thống

2.2. Các hàm xử lý chuỗi:

•**CONCAT**: Là hàm nối nhiều chuỗi lại với nhau

Cú pháp: CONCAT(*string1*, *string2*, *….*, *string_n*)

Ví dụ: SELECT CONCAT('LHU', '.com'); --KQ: LHU.com

•**LEFT**: Trích xuất 1 chuỗi con từ 1 chuỗi lớn, bắt đầu từ ký tự ngoài cùng bên trái

Cú pháp: LEFT(*string*, *num_of_chars*)

Ví dụ: SELECT LEFT('LH University', 2); --KQ: LH

•**RIGHT**: Trích xuất 1 chuỗi con từ 1 chuỗi lớn, bắt đầu từ ký tự ngoài cùng bên phải

Cú pháp: RIGHT(*string*, *num_of_chars*)

Ví dụ: SELECT RIGHT('LH University', 10); --KQ: University

2. Hàm hệ thống

- **LEN**: Là hàm trả về số lượng ký tự của chuỗi truyền vào

  Cú pháp: LEN(*string*)

  Ví dụ: SELECT LEN('LHU'); --KQ: 3

- **LTRIM**: Trả về chuỗi với các khoảng trắng ngoài cùng bên trái (nếu có) sẽ bị xóa bỏ

  Cú pháp: LTRIM(*string*)

  Ví dụ: SELECT LTRIM(‘ LH University’); --KQ: LH University

- **RTRIM**: Trả về chuỗi với các khoảng trắng ngoài cùng bên phải (nếu có) sẽ bị xóa bỏ

  Cú pháp: RTRIM(*string*)

  Ví dụ: SELECT RTRIM(‘LH University ’); --KQ: LH University

- **TRIM**: Trả về chuỗi với các khoảng trắng ở đầu và cuối sẽ bị xóa bỏ

  Cú pháp: TRIM(*string*)

  Ví dụ: SELECT TRIM(‘ LH University ’); --KQ: LH University

2. Hàm hệ thống

- **UPPER**: chuyển thành chuỗi in hoa

  Cú pháp: UPPER(*string*)

  Ví dụ: SELECT UPPER('Lhu'); --KQ: LHU

- **LOWER**: chuyển thành chuỗi in thường

  Cú pháp: LOWER(*string*)

  Ví dụ: SELECT LOWER('LHU'); --KQ: lhu

- **REPLACE**: sử dụng để thay thế tất cả các lần xuất hiện của chuỗi con a thành chuỗi con b mới trong một chuỗi cho trước.

  Cú pháp: REPLACE(*string, string_a, stringb*)

  Ví dụ: SELECT REPLACE('Lac Hong University', 'L', 'A'); --KQ: Aac Hong University

  SELECT REPLACE('Lac Hong University', 'Lac Hong', 'LHU');

  --KQ: LHU University

2. Hàm hệ thống

- **SUBSTRING**: Cho phép trích xuất 1 chuỗi con có độ dài chỉ định bắt đầu từ 1 vị trí nào đó trong chuỗi đầu vào

  Cú pháp: SUBSTRING(*string, start, length*)

  Ví dụ: SELECT SUBSTRING('LHU University', 1, 3); --KQ: LHU

  SELECT SUBSTRING('LHU University', 5, 6); --KQ: Univer

2. Hàm hệ thống

2.3. Các hàm chuyển kiểu dữ liệu:

- **CAST**: Chuyển đổi một biểu thức từ một kiểu dữ liệu này sang kiểu dữ liệu khác. Nếu chuyển đổi không thành công, CAST sẽ báo lỗi, ngược lại nó sẽ trả về giá trị chuyển đổi tương ứng

Cú pháp: CAST(*bieuthuc as kieudulieu [(dodai)]*)

Lưu ý:
Khi chuyển đổi kiểu dữ liệu *float* hay *numeric* sang số nguyên *int,* hàm cast sẽ cắt phần thập phân phía sau

```
SELECT CAST(14.85 AS int);
KQ: 14 (kết quả cắt phần thập phân phía sau)

SELECT CAST(14.85 AS float);
KQ: 14.85

SELECT CAST(15.6 AS varchar);
KQ: '15.6'

SELECT CAST(15.6 AS varchar(4));
KQ: '15.6'

SELECT CAST('15.6' AS float);
KQ: 15.6

SELECT CAST('2019-05-02' AS datetime);
KQ: '2019-05-02 00:00:00.000'
```

2. Hàm hệ thống

- **CONVERT**: Chuyển đổi một biểu thức nào đó sang một kiểu dữ liệu bất kỳ mong muốn nhưng có thể theo một định dạng nào đó (đặc biệt đối với kiểu dữ liệu ngày). Nếu chuyển đổi không thành công, CONVERT sẽ báo lỗi, ngược lại nó sẽ trả về giá trị chuyển đổi tương ứng

  Cú pháp: CONVERT(*kieudulieu [(dodai)], bieuthuc, dinhdang*)

Ví dụ

```
•SELECT CONVERT(int, 14.85);
    KQ: 14 (kết quả cắt phần thập phân phía sau)
•SELECT CONVERT(float, 14.85);
    KQ : 14.85
•SELECT CONVERT(varchar, 15.6);
    KQ : '15.6'
•SELECT CONVERT(varchar(4), 15.6);
    KQ : '15.6'
•SELECT CONVERT(float, '15.6');
    KQ : 15.6
•SELECT CONVERT(datetime, '2019-05-02');
    KQ : 2019-05-02 00:00:00.000
•SELECT CONVERT(datetime, '05/02/2019', 101);
    KQ : 2019-05-02 00:00:00.000
```

2. Hàm hệ thống

Định dạng năm (yy)	Định dạng năm (yyyy)	Hiển thị dữ liệu
0	100	mon dd yyyy hh:miAM/PM (Default)
1	101	mm/dd/yyyy (US standard)
2	102	yy.mm.dd (ANSI standard)
3	103	dd/mm/yy (British/French standard)
4	104	dd.mm.yy (German standard)
5	105	dd-mm-yy (Italian standard)
6	106	dd mon yy
7	107	Mon dd, yy
8	108	hh:mi:ss
9	109	mon dd yyyy hh:mi:ss:mmmAM/PM
10	110	mm-dd-yy (USA standard)
11	111	yy/mm/dd (Japan standard)
12	112	yymmdd (ISO standard)
13	113	dd mon yyyy hh:mi:ss:mmm (Europe standard - 24 hour clock)
14	114	hh:mi:ss:mmm (24 hour clock)
20	120	yyyy-mm-dd hh:mi:ss (ODBC canonical - 24 hour clock)
21	121	yyyy-mm-dd hh:mi:ss:mmm (ODBC canonical - 24 hour clock)
	126	yyyy-mm-ddThh:mi:ss:mmm (ISO8601 standard)
	127	yyyy-mm-ddThh:mi:ss:mmmZ (ISO8601 standard)
	130	dd mon yyyy hh:mi:ss:mmmAM/PM (Hijri standard)
	131	dd/mm/yy hh:mi:ss:mmmAM/PM (Hijri standard)

2. Hàm hệ thống

2.4. Các hàm xử lý ngày tháng:

- **GETDATE**: Trả về ngày tháng năm hiện tại của hệ thống

Cú pháp: **GETDATE**()

Ví dụ:

```
SELECT GETDATE();
KQ: '2019-02-25 18:11:00.160'
```

- **MONTH**: Trả về số nguyên là tháng trong năm (1->12) từ ngày tháng truyền vào

Cú pháp: **MONTH**(ngaythang)

```
SELECT MONTH ('2019/04/28');
KQ: 4

SELECT MONTH ('2019/03/31 10:05');
KQ: 3

SELECT MONTH ('2019/04/01 10:05:18.621');
KQ: 4
```

2. Hàm hệ thống

- **YEAR**: Trả về số nguyen là năm từ ngày tháng truyền vào

  Cú pháp: **YEAR**(ngaythang)

```
SELECT YEAR ('2019/04/28');
KQ: 2019

SELECT YEAR ('2020/03/31 10:05');
KQ: 2020

SELECT YEAR ('2021/04/01 10:05:18.621');
KQ: 2021
```

2. Hàm hệ thống

- **DATEDIFF**: Trả về chênh lệch giữa hai giá trị thời gian

Cú pháp: DATEDIFF(dangthoigian, thoigian1, thoigian2)

dangthoigian: dạng thời gian sử dụng để tính chênh lệch giữa thoigian1 và thoigian2. Có các giá trị như sau:

Giá trị	Giải thích
year, yyyy, yy	Năm
quarter, qq, q	Quý
month, mm, m	Tháng
dayofyear	Ngày trong năm
day, dy, y	Ngày
week, ww, wk	Tuần
weekday, dw, w	Ngày trong tuần
hour, hh	Giờ
minute, mi, n	Phút
second, ss, s	Giây
millisecond, ms	Milli giây

2. Hàm hệ thống

Ví dụ:

```
SELECT DATEDIFF(year, '2019/04/28', '2021/04/28');
KQ: 2

SELECT DATEDIFF(yyyy, '2019/04/28', '2021/04/28');
KQ: 2

SELECT DATEDIFF(yy, '2019/04/28', '2021/04/28');
KQ: 2

SELECT DATEDIFF(month, '2019/01/01', '2019/04/28');
Result: 3

SELECT DATEDIFF(day, '2019/01/01', '2019/04/28');
KQ : 117

SELECT DATEDIFF(hour, '2019/04/28 08:00', '2019/04/28 10:45');
KQ : 2

SELECT DATEDIFF(minute, '2019/04/28 08:00', '2019/04/28 10:45');
KQ : 165
```

2. Hàm hệ thống

- **DATEADD**: Trả về giá trị thời gian mới sau khi được cộng thêm 1 khoảng thời gian chỉ định

Cú pháp: DATEDIFF(dangthoigian, number, thoigian)
dangthoigian: dạng thời gian sử dụng để tính thêm vào. Có giá trị như sau:

Giá trị	Giải thích
year, yyyy, yy	Năm
quarter, qq, q	Quý
month, mm, m	Tháng
dayofyear	Ngày trong năm
day, dy, y	Ngày
week, ww, wk	Tuần
weekday, dw, w	Ngày trong tuần
hour, hh	Giờ
minute, mi, n	Phút
second, ss, s	Giây
millisecond, ms	Milli giây

2. Hàm hệ thống

Ví dụ:

```
SELECT DATEADD(year, 1, '2019/04/28');
KQ: '2020-04-28 00:00:00.000'

SELECT DATEADD(yyyy, 1, '2019/04/28');
KQ: '2020-04-28 00:00:00.000'

SELECT DATEADD(yy, 1, '2019/04/28');
KQ: '2020-04-28 00:00:00.000'

SELECT DATEADD(year, -1, '2019/04/28');
KQ: '2018-04-28 00:00:00.000'

SELECT DATEADD(month, 1, '2019/04/28');
KQ: '2019-05-28 00:00:00.000'

SELECT DATEADD(month, -1, '2019/04/28');
KQ: '2019-03-28 00:00:00.000'

SELECT DATEADD(day, 1, '2019/04/28');
KQ: '2019-04-29 00:00:00.000'

SELECT DATEADD(day, -1, '2019/04/28');
KQ: '2019-04-27 00:00:00.000'
```

3. Hàm tự định nghĩa

3. Hàm tự định nghĩa

3.1. Các lệnh điều khiển và vòng lặp

3.1.1. Lệnh IF ... ELSE

Câu lệnh IF...ELSE dùng để thực thi các lệnh có điều kiện, nếu lệnh đúng thì thực thi lệnh đó, nếu sai sẽ thực thi một lệnh khác.

```
IF <dieukien>
{...câu lệnh thực thi khi điều kiện là TRUE...}

[ ELSE
{...câu lệnh thực thi khi điều kiện là FALSE...} ]
```

3. Hàm tự định nghĩa

```
DECLARE @nhanvien_salary INT;
SET @nhanvien_salary = 15000000;
IF @nhanvien_salary < 10000000
   PRINT 'Chuyen vien';
GO
```

```
DECLARE @nhanvien_salary INT;
SET @nhanvien_salary = 15000000;
IF @nhanvien_salary > 10000000
   PRINT 'Truong phong';
ELSE
   PRINT 'Chuyen vien';
GO
```

```
DECLARE @nhanvien_salary INT;
SET @nhanvien_salary = 15000000;
IF @nhanvien_salary > 12000000
   PRINT 'Giam doc';
ELSE
BEGIN
   IF @nhanvien_salary > 10000000
      PRINT 'Truong phong’;
   ELSE
      PRINT 'Chuyen vien';
END;
GO
```

3. Hàm tự định nghĩa

3.1.2. Lệnh CASE

Câu lệnh **CASE** dùng để thiết lập điều kiện rẽ nhánh

Simple case

```
CASE bieuthuc_dauvao
        WHEN bieuthuc_1 THEN ketqua_1
        WHEN bieuthuc_2 THEN ketqua_2
        ...
        WHEN bieuthuc_n THEN ketqua_n
        ELSE ketqua_khac
END
```

Searched case

```
CASE WHEN dieukien_1 THEN ketqua_1
        WHEN dieukien_2 THEN ketqua_2
         ...
        WHEN dieukien_n THEN ketqua_n
        ELSE ketqua_khac
END
```

Simple CASE là so sánh một biểu thức với một bộ các biểu thức đơn giản để xác định kết quả

Searched CASE là đánh giá một bộ các biểu thức Boolean để xác định kết quả.

3. Hàm tự định nghĩa

```
SELECT Tenchuyenmuc, Code
            (CASE code WHEN 01 THEN 'Laptrinh.com’
                              WHEN 02 THEN 'Quantrimang.com’
                              WHEN 03 THEN 'Cuocsong.com’
                              ELSE 'Khoahoc-Quantrimang.com’
              END) AS domain
FROM chuyenmuc
ORDER BY Code
```

Tenchuyenmuc	Code	domain
SQL Server	01	Laptrinh.com
Linux	02	Quantrimang.com
Python	01	Laptrinh.com

3. Hàm tự định nghĩa

```
SELECT Tenchuyenmuc, Code,
            CASE WHEN code = 01 THEN 'Laptrinh-Quantrimang.com'
                    WHEN code = 02 THEN 'Congnghe-Quantrimang.com'
                    WHEN code = 03 THEN 'Cuocsong-Quantrimang.com'
                    ELSE 'Khoahoc-Quantrimang.com'
             END as domain
FROM chuyenmuc
```

Tenchuyenmuc	Code	domain
SQL Server	01	Laptrinh.com
Linux	02	Quantrimang.com
Python	01	Laptrinh.com

3. Hàm tự định nghĩa

3.1.3. Lệnh lặp WHILE

Được sử dụng nếu muốn chạy lặp đi lặp lại một đoạn mã khi điều kiện cho trước trả về giá trị là TRUE

```
WHILE <dieukien>
BEGIN
      ......câu lệnh thực thi khi điều kiện là TRUE.....
END;
```

Ví dụ:

```
DECLARE @Number INT = 1 ;
DECLARE @Total INT = 0 ;
WHILE @Number < = 10
BEGIN
      SET @Total = @Total + @Number;
      SET @Number = @Number + 1 ;
END
PRINT @Total;
GO
```

3. Hàm tự định nghĩa

3.2. Function (Hàm) tự định nghĩa

Có 2 loại Function:

1. Scalar Function: Là hàm do người dùng tự định nghĩa, có thể không truyền hoặc truyền vào nhiều tham số đầu vào, tuy nhiên chỉ trả ra 1 giá trị đơn.

2. Table Function: Là hàm do người dùng tự định nghĩa, và trả ra dữ liệu kiểu bảng. Do kiểu trả về là kiểu bảng nên có thể sử dụng table function giống như khi sử dụng bảng

3. Hàm tự định nghĩa

3.2.1. Scalar Function:

Cú pháp:

```
CREATE FUNCTION <function_name>
(
      parameter_list
)
RETURNS data_type
AS
BEGIN
      statements
      RETURN value
END
```

Lưu ý: kiểu dữ liệu value trả về buộc phải giống kiểu dữ liệu khai báo sau RETURNS

3. Hàm tự định nghĩa

Ví dụ: Viết hàm tính thành tiền theo số lượng và đơn giá truyền vào

```
CREATE FUNCTION TinhTien
(
    @quantity INT,
    @price DEC(10,2)
)
RETURNS DEC(10,2)
AS
BEGIN
    RETURN @quantity * @price;
END;
```

Sử dụng hàm:

```
SELECT  TinhTien(25, 12500);
KQ: 312500
```

3. Hàm tự định nghĩa

Lưu ý:

•Scalar function có thể định nghĩa truyền vào 1 hoặc nhiều tham số, nhưng chỉ trả ra 1 giá trị đơn. Do đó buộc phải có lệnh return

•Scalar function có thể sử dụng các lệnh rẽ nhánh IF...ELSE hoặc vòng lặp WHILE trong thân của nó

•Scalar function không thể update dữ liệu

•Scalar function có thể gọi các function khác

3. Hàm tự định nghĩa

3.2.1. Table Function:

Cú pháp:

```
CREATE FUNCTION <function_name>
(
      parameter_list
)
RETURNS TABLE
AS
BEGIN
      statements
      RETURN value
END
```

Trả ra dữ liệu của 1 bảng

3. Hàm tự định nghĩa

Ví dụ: Viết hàm trả về danh sách các nhân viên, với mã phòng ban là tham số được truyền vào

```
CREATE FUNCTION DsNhanVien
(
    @MaPB INT
)
RETURNS TABLE
AS
RETURN
    SELECT MaNhanVien, TenNhanVien, MaPB
    FROM    PhongBan
    WHERE MaPB = @MaPB;
```

Gọi hàm:

```
Select * From DsNhanVien(10);
```

Có thể chỉ định cột sẽ hiển thị

```
Select MaNhanVien, TenNhanVien From DsNhanVien(10);
```

3. Hàm tự định nghĩa

3.2.2. Table Function đa câu lệnh: Bảng trả về có giá trị từ nhiều bảng khác. Nghĩa là có thể thực hiện nhiều truy vấn trong function và tổng hợp kết quả vào bảng được trả về

Ví dụ: Tạo function hiển thị kết quả là danh sách kết hợp của nhân viên và khách hàng vào 1 bảng duy nhất

Gọi hàm:

```
Select * From DanhSachNV_KH();
```

Tạo hàm

```
CREATE FUNCTION   DanhSachNV_KH()
   RETURNS @contacts TABLE (
                                    HoTen VARCHAR(50),
                                    Email VARCHAR(255),
                                    DienThoai VARCHAR(25),
                                    Loai VARCHAR(20)
                                   )
AS
BEGIN
   INSERT INTO @contacts
   SELECT TenNV, Email, DienThoai, 'Staff'
   FROM    NhanVien;

   INSERT INTO @contacts
   SELECT TenKH, Email, DienThoai, 'Customer'
   FROM    KhachHang;
   RETURN;
END;
```

Câu hỏi ???