Lập trình trong SQL Server

THỦ TỤC VÀ HÀM

Biến trong SQL

- Biến là vùng nhớ trong bộ nhớ được đặt tên để chứa giá trị dữ liệu.
- Biến có thể phân thành 2 loại: Biến cục bộ và biến toàn bộ. Dữ liệu có thể truyền cho câu lệnh SQL bằng biến cục bộ.
- Local Variables (Biến cục bộ): Trong Transact-SQL, biến cục bộ được khai báo và sử dụng tạm thời khi thực thi câu lệnh SQL.

  Cú pháp:

  DECLARE
  {
  @local_variable_name [AS] data_type
  }

Biến trong SQL (tt)

- Gán giá trị cho biến: dùng SET hoặc SELECT

  SET @local_variable = value

  Hoặc

  SELECT @local_variable = value

- Xem giá trị hiện hành của biến

  PRINT @biến

- Đổi kiểu dữ liệu
  - CAST (@biến AS kiểu dữ liệu)
  - CONVERT: chuyển đổi dạng ngày tháng

Biến trong SQL (tt)

- Global Variables (Biến toàn cục):
  - Biến toàn cục là biến có sẵn và hệ thống quản lý, được đặt tên bắt đầu bởi hai ký hiệu @.
  - Ví dụ:
    - SELECT @@VERSION AS 'SQL Server version‘
  - Một số biến thường dùng
    - @@RowCount: tổng số bản ghi
    - @@Error: số mã lỗi của câu lệnh gần nhất
    - @@Fetch_Status: trạng thái việc đọc dữ liệu theo từng bản ghi (cursor)

Kiểu dữ liệu

Data Types in SQL Server			
tinyint	real	datetime	varbinary
smallint	char	smalldatetime	uniqueidentifier
int	nchar	image	numeric
bigint	varchar	money	timestamp
bit	nvarchar	smallmoney	sql_variant
decimal	text	xml	table
float	ntext	cursor	binary

Kiểu dữ liệu (tt)

Kiểu dữ liệu	Kích thước	Miền giá trị dữ liệu lưu trữ
> Các kiểu dữ liệu dạng số nguyên		
Int	4 bytes	từ-2,147,483,648đến +2,147,483,647
SmallInt	2 bytes	từ -32768 đến +32767
TinyInt	1 byte	từ 0 đến 255
Bit	1 byte	0, 1 hoặc Null
> Các kiểu dữ liệu dạng số thập phân		
Decimal, Numeric	17bytes	từ -10^38 đến +10^38
> Các kiểu dữ liệu dạng số thực		
Float	8 bytes	từ -1.79E+308 đến +1.79E+308
Real	4 bytes	từ -3.40E+38 đến +3.40E+38

Kiểu dữ liệu (tt)

> Các kiểu dữ liệu dạng chuỗi có độ dài cố định

Char	N bytes	từ 1 đến 8000 ký tự, mỗi ký tự là một byte

> Các kiểu dữ liệu dạng chuỗi có độ dài biến đổi

VarChar	N bytes	từ 1 đến 8000 ký tự, mỗi ký tự là 1 byte
Text	N bytes	từ 1 đến 2,147,483,647 ký tự, mỗi ký tự là 1 byte

> Các kiểu dữ liệu dạng chuỗi dùng font chữ Unicode

NChar	2*N bytes	từ 1 đến 4000 ký tự, mỗi ký tự là 2 bytes
NVarChar	2*N bytes	từ 1 đến 4000 ký tự, mỗi ký tự là 2 bytes
NText	2*N bytes	từ 1 đến 1,073,741,823 ký tự, mỗi ký tự là 2 bytes

Kiểu dữ liệu (tt)

> Các kiểu dữ liệu dạng tiền tệ

Money	8 bytes	từ -922,337,203,685,477.5808 đến +922,337,203,685,477.5807
SmallMoney	4 bytes	từ -214,748.3648 đến + 214,748.3647

> Các kiểu dữ liệu dạng ngày và giờ

DateTime	8 bytes	từ01/01/1753đến31/12/9999
SmallDateTime	4 bytes	từ01/01/1900đến06/06/2079

> Các kiểu dữ liệu dạng chuỗi nhị phân (Binary String)

Binary	N bytes	từ 1 đến 8000 bytes
VarBinary	N bytes	từ 1 đến 8000 bytes
Image	N bytes	từ 1 đến 2,147,483,647 bytes

Định nghĩa chú thích

- Microsoft SQL Server hỗ trợ hai loại chú thích sau:
  - -- chú thích một dòng
  - /* . . . */ chú thích nhiều dòng

Cấu trúc lệnh

- Cấu trúc lệnh IF

  if (điều_kiện)

  lệnh …. | khối_lệnh

  else

  lệnh …. | khối_lệnh

- Cấu trúc lệnh If Exists

  If exists (Câu lệnh Select)

  Lệnh…| khối_lệnh

  [Else

  Lệnh…| khối_lệnh]

- khối_lệnh := begin

  lệnh … | khối_lệnh

  end

Cấu trúc lệnh (tt)

- Cấu trúc lệnh WHILE

  while (điều_kiện)

  lệnh | khối_lệnh

- Lệnh ngắt vòng lặp

  break

  continue

Cấu trúc lệnh (tt)

- Biểu thức CASE

Case biểu thức

When giá trị 1 **Then** biểu thức 1

[**When** giá trị 2 **Then** biểu thức 2]

…

[**Else** biểu thức n]

End

Cấu trúc lệnh (tt)

- Biểu thức CASE

Case

When điều kiện 1 **Then** biểu thức 1

[**When** điều kiện 2 **Then** biểu thức 2]

…

[**Else** biểu thức n]

End

Cấu trúc lệnh (tt)

- Ví dụ Tính tổng số chẵn từ 1 -> 100

```
Declare @t int, @x int
Set @t = 0 ; Set @x = 1
   While (@x <= 100)
      begin
          if  ((@x % 2) = 0)
                set @t = @t + @x
                set @x = @x + 1
      end
Print @t
```

Con trỏ

- Khai báo biến:

  Declare Tên_Biến CURSOR

  [phạm vi] [di chuyển][trạng thái][xử lý]

  For câu lệnh Select

  [For update [OF danh sách cột]]

- Trong đó:
  - Câu lệnh select: không chứa các mệnh đề Into, Compute, Compute by
  - Danh sách cột: là danh sách các cột sẽ thay đổi được

Con trỏ (tt)

- Phạm vi :
  - Local :chỉ sử dụng trong phạm vi khai báo (mặc định)
  - Global :sử dụng chung cho cả kết nối
- Di chuyển :
  - ForWard_Only :chỉ di chuyển một hướng từ trước ra sau(mặc định)
  - Scroll : di chuyển tùy ý

Con trỏ (tt)

- Trạng thái
  - Static: dữ liệu trên Cursor không thay đổi mặc dù dữ liệu trong bảng nguồn thay đổi(mặc định)
  - Dynamic :dữ liệu trên Cursor sẽ thay đổi khi dữ liệu trong bảng nguồn thay đổi
  - KeySet :giống Dynamic nhưng chỉ thay đổi những dòng bị cập nhật
- Xử lý :
  - Read_Only :chỉ đọc (mặc định)
  - Scroll_Lock : đọc/ghi

Con trỏ (tt)

- Sử dụng

  open tên_biến_cursor

  ….

  close tên_biến_cursor

- Hủy cursor

  deallocate tên_biến_cursor

Con trỏ (tt)

- Di chuyển Cursor

  fetch *định_vị*

  from tên_biến_cursor

  into @tên_biến [,… n]

  định_vị := next | prior | last | first |

  absolute (giá_trị | biến)

  relative (giá_trị | biến)

Con trỏ (tt)

- NEXT: Di chuyển về sau
- PRIOR : Di chuyển về trước
- FIRST : Di chuyển về đầu
- LAST : Di chuyển về cuối
- ABSOLUTE n: nếu n>0 di chuyển đến bản ghi thứ |n| tính từ bản ghi đầu tiên, nếu n<0 : tính từ bản ghi cuối
- RELATIVE n :di chuyển đến bản ghi thứ n tính từ bản ghi hiện hành

Con trỏ

- Trạng thái Cursor

  @@fetch_status

  =0 : Đang trong dòng dữ liệu
  (lần đi kế tiếp thành công)

  ≠0 : Ngoài dòng dữ liệu
  (lần đi kế tiếp không thành công)

Con trỏ (tt)

- Chú ý: thứ tự các thao tác khi xử lý dữ liệu trên CurSor

1. Định nghĩa biến Cursor
2. Mở Cursor
3. Duyệt và xử lý dữ liệu trên Cursor
4. Đóng và giải phóng Cursor

Ví dụ

- Tính điểm trung bình chung (DTBC) của học sinh

```
Declare hs cursor for select mahs from DSHS
Open hs
Declare @mahs nvarchar(5), @dtb float
Fetch next from hs into @mahs
While (@@fetch_status = 0)
begin
select @dtb=round((toan*2+ly*2+hoa+ly)/6,2) from diem where
MAHS=@mahs
update dshs set dtbc=@dtb where MAHS=@mahs
 Fetch next from hs into @mahs
end
Close hs; Deallocate hs
```

```
DECLARE cs CURSOR FOR SELECT MAHS FROM DSHS
OPEN cs
DECLARE @mahs nvarchar(5), @dtb float, @toan float, @ly float, @van float, @hoa
float, @dtn float
FETCH NEXT FROM cs into @mahs
WHILE @@FETCH_STATUS=0
BEGIN
declare @xl nchar(25)
select @toan=toan, @van=van, @hoa=hoa, @ly=ly,
@dtb=round((toan*2+van*2+ly+hoa)/6,2) from DIEM where MAHS=@mahs
set @dtn=@toan
if @dtn>@van set @dtn=@van
if @dtn>@hoa set @dtn=@hoa
if @dtn>@ly set @dtn=@ly

IF (@dtb>=8 AND @dtn>=6.5) SET @xl=N'Giỏi'
ELSE IF (@dtb>=7 AND @dtn>=5) SET @xl=N'Khá'
ELSESET @xl=N'Trung bình'
Update dshs set XepLoai=@xl where MAHS=@mahs
FETCH NEXT FROM cs into @mahs
END
CLOSE cs
DEALLOCATE cs
```

Tránh sử dụng cursor như thế nào?

- Tại sao lại tránh sử dụng cursor?
- Tránh sử dụng bằng cách nào?
  - Sử dụng câu lệnh SQL chuẩn (chẳng hạn: Case)
  - Sử dụng lặp (while)
  - Sử dụng bảng tạm

Nội thủ tục (Stored procedure)

- Là các chương trình được lưu trữ trong CSDL. Được gọi thi hành khi có yêu cầu.
- Thường được thiết kế để thi hành các luật ràng buộc
- Ích lợi của SP:
  - Giảm thời gian biên dịch
  - Đơn giản trong bảo trì
  - Bảo mật
  - Giảm dung lượng truyền dữ liệu

Nội thủ tục (Stored procedure) (tt)

- Có 2 loại SP:
  - SP hệ thống (system sp)
  - SP người dùng (user sp)
- SP người dùng gồm 3 loại:
  - Trigger
  - User stored procedure
  - Defined function

Nội thủ tục (Stored procedure) (tt)

- SP được xây dựng từ các câu lệnh T-SQL và được lưu trú trên SQL server.
- Muốn thực hiện một SP, NSD chỉ cần thực hiện một lời gọi hàm.
- Khi SP được chạy lần đầu tiên nó sẽ được biên dịch qua 5 bước và sinh ra một mô hình truy vấn. Mô hình này sẽ được đặt trong một CSDL của SQL server, lần sau chạy lại thủ tục sẽ không phải dịch lại nữa.

Nội thủ tục (Stored procedure) (tt)

- Năm bước biên dịch thủ tục:
  - Thủ tục được phân tích ra thành nhiều phần
  - Kiểm tra sự tồn tại của các đối tượng (view, table, …) mà thủ tục tham chiếu tới.
  - Lưu trữ tên thủ tục vào bảng sysobject, lưu trữ các mã lệnh của thủ tục vào bảng syscomments.
  - Sinh ra mô hình truy vấn của thủ tục và lưu vào bảng sysprocedure
  - Khi SP được chạy lần đầu tiên, cây truy vấn sẽ được đọc và được tối ưu thành một kế hoạch thủ tục và chạy → tiết kiệm thời gian tái phân tích, biên dịch cây truy vấn mỗi khi chạy thủ tục.

Nội thủ tục (Stored procedure) (tt)

- Trong một phiên làm việc, nếu SP được thực hiện, nó sẽ được lưu trữ vào vùng nhớ đệm. Những lần sau nếu SP được gọi thực hiện lại thì nó sẽ được đọc trực tiếp ra từ vùng nhớ đệm → nâng cao hiệu suất chạy truy vấn.

Nội thủ tục (Stored procedure) (tt)

- Tạo sp:

  CREATE PROCEDURE procedurename [parameter1 datatype [length] [OUTPUT], parameter2...]

  AS

  BEGIN ... END

  Trong đó:

  + parameter1, parameter2, .. .: là các tham số
  + datatype: kiểu dữ liệu của tham số
  + output: tham số nhận giá trị trả về
  + trong khối BEGIN … END là các lệnh SQL.

Nội thủ tục (Stored procedure) (tt)

- Thực thi thủ tục

  exec tên_thủ_tục giá_trị | @biến [output] [,…n]

  – EXEC tên_thủ_tục giá_trị_1, giá_trị_2, …
  – EXEC tên_sp @p1 = giá_trị, @p2 = giá_trị, …
- Xóa thủ tục

  Drop procedure tên_thủ_tục
- Thay đổi thủ tục

  Alter procedure tên_thủ_tục

Nội thủ tục (Stored procedure) (tt)

- Ví dụ: sp có tham số
- /* tạo thủ tục có đầu vào là mã học sinh, đầu ra là điểm trung bình của học sinh đó*/

```
CREATE PROCEDURE DiemTrungBinh @mahs nvarchar(5), @dtb float output
AS
begin
select @dtb=round((toan*2+van*2+ly+hoa)/6, 2) from diem where MAHS=@mahs
End

Gọi thủ tục:
declare @tb float
exec DiemTrungBinh '00001', @tb output
print @tb
```

Nội thủ tục (Stored procedure) (tt)

- Ví dụ Viết thủ tục xóa các sinh viên theo thành phố

  sinhvien (masv char(5), tp char(5))

```
create procedure xoasinhvien
      @tp nchar(50)
as
begin
      delete from T1 where tp = @tp
end

exec  xoasinhvien  'HCM'
```

Nội thủ tục (Stored procedure) (tt)

- Ví dụ Viết thủ tục đếm xem có bao nhiêu sinh viên theo thành phố.

```
create procedure dem @tp char(5), @t int output  as
begin
      select @t = count(*) from sinhvien
      where tp = @tp
end

declare @tong int
exec  dem  'HCM' , @tong output
print @tong
```

Nội thủ tục (Stored procedure) (tt)

- **Ví dụ: sp có tham số đầu ra kiểu trỏ**

```
CREATE PROCEDURE sp5 @p cursor varying output
AS
begin
    set @p = cursor forward_only static for
    select * from T1
    open @p
end
CREATE PROCEDURE sp6
AS
begin
    declare @mycursor cursor, @myid int, @myname char(100)
    execute sp5 @p= @mycursor output
    fetch next from @mycursor into @myid, @myname
    while (@@FETCH_STATUS = 0)
    begin
       print convert(varchar(100),@myid) + '     ' + @myname
       fetch next from @mycursor into @myid, @myname
    end
    close @mycursor
    deallocate @mycursor
end
```

Nội thủ tục (Stored procedure) (tt)

- **Ví dụ: sp có giá trị trả về**

```
CREATE PROCEDURE  sp7
AS
begin
   declare @x char(100)
   select @x = ‘Hello world’
   return @x
End

declare @v char(100)
exec @v = sp7
print @v
```

Kiểm soát lỗi với TRY … CATCH

- Thực hiện các lệnh trong khối Try, nếu gặp lỗi sẽ chuyển qua xử lý bằng các lệnh trong khối Catch
- Cú pháp:

```
BEGIN TRY
    {các câu lệnh}
END TRY
BEGIN CATCH
    {các câu lệnh}
END CATCH
```

Kiểm soát lỗi cới TRY … CATCH (tt)

- Lưu ý:
  - Try và Catch phải cùng lô xử lý
  - Sau khối Try phải là khối Catch
  - Có thể lồng nhiều cấp

```
BEGIN TRY
    -- Generate some error.
    Declare @str varchar(20);
    Set @str = 'SQL SERVER!';
    print convert(datetime, @str);
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber
        ,ERROR_SEVERITY() AS ErrorSeverity
        ,ERROR_STATE() AS ErrorState
        ,ERROR_PROCEDURE() AS ErrorProcedure
        ,ERROR_LINE() AS ErrorLine
        ,ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
GO
```

Tạo một câu lệnh SQL phát sinh ra lỗi

Đọc thông tin về lỗi

Kiểm soát lỗi với TRY … CATCH (tt)

- Một số hàm cung cấp thông tin về lỗi vừa phát sinh:

STT	Tên hàm	Chức năng
1	Error_number()	Trả lại mã lỗi (dưới dạng số)
2	Error_severity()	Trả lại mức độ nghiêm trọng của lỗi
3	Error_state()	Trả lại trạng thái lỗi (dưới dạng số)
4	Error_procedure()	Trả lại tên thủ tục hoặc Trigger phát sinh lỗi
5	Error_line()	Trả lại vị trí dòng lệnh phát sinh lỗi
6	Error_message()	Trả lại thông báo lỗi dưới hình thức văn bản (text)

Kiểm soát lỗi cới TRY … CATCH (tt)

- Ví dụ kiểm soát lỗi TRY...CATCH với thủ tục:

```
ALTER PROCEDURE deleteA
    @A1 int
AS
BEGIN TRY
    DELETE FROM A WHERE A1 = @A1
END TRY
BEGIN CATCH
    PRINT 'ERROR on delete record: ' + CONVERT(varchar,@A1)
    RETURN 1001 -- return user-defined error code
END CATCH
GO
```

Sử dụng TRY ... CATCH để bắt và xử lý lỗi; Hai khối lệnh TRY và CATCH phải nằm liền kề nhau.

Hàm (function)

- Tạo lập hàm: Scalar Functions

```
CREATE FUNCTION [ schema_name. ] function_name
    ( [ { @parameter_name data_type [ = default ] [ READONLY ] } [ ,...n ] ] )
    RETURNS return_data_type
    [ WITH <function_option> [ ,...n ] ]
    [ AS ]
BEGIN
    function_body
    RETURN scalar_expression
END
```

Hàm (function)…

- Tạo lập hàm: Table-Valued Functions

```
CREATE FUNCTION [ schema_name. ] function_name
    ( [ { @parameter_name parameter_datatype [ = default ]
    [ READONLY ] } [ ,...n ] ]
    RETURNS TABLE
    [ WITH <function_option> [ ,...n] ]
    [ AS ]
    RETURN [ (] select_stmt[ )]
```

Hàm (function)…

```
CREATE FUNCTION [ schema_name. ] function_name
    ( [ { @parameter_name data_type [ = default ] [ READONLY ] } [ ,...n
    ] ] )
RETURNS @table_name TABLE
            (
             columns              data_type
            )
 AS
BEGIN
            function_body
RETURN
 END
```

Hàm (function)

- Thực thi hàm
  - = tên_hàm (giá_trị | @biến [,…n])
  - dùng select tên_hàm hoặc select…from tên_hàm
- Xóa hàm

  Drop function tên_hàm
- Thay đổi hàm

  Alter function tên_hàm

  ……

Hàm (function)

- Ví dụ

  Viết hàm sinh ra mã sinh viên tự động theo quy tắc

  - Mã sinh viên có dạng: BA0001

  'BA' : quy định (luôn có)

  0001 : là số

  VD:

  Hiện tại sinh viên có mã cao nhất là BA0024

  Thì sinh mã mới là BA0025

Hàm (function)

```
Create function sinhkhoa () returns char(6) As
Begin
    declare @max int
    select
        @max = max(cast(substring(masv,3,4) as int)) + 1
        from sinhvien
    declare @s char(8)
    set @s = '000' + rtrim(cast(@max as char(4)))
    set @s = 'BA' + right(@s,4)
    return @s
end
```

Hàm (function)

- Ví dụ với Table Function

```
create function laydssv (@malop char(5))
returns TABLE
as
    return (
        select masv,tensv from sinhvien
        where malop = @malop
        )

select * from laydssv('QT1')
```

Hàm (function)

- Ví dụ với Table Function

```
create function laydssv1 (@malop char(5))
        returns  @btam table(masv char(5),tensv char(20))
as
begin
        insert into @btam  select masv,tensv from sinhvien
                where malop = @malop
        return
end

select * from laydssv1('QT1')
```

HÀM CỦA NSD
(USER DEFINED FUNCTIONS-UDFs)

– UDFs giống như SP nhưng khác ở các điểm sau:

UDF	**SP**
- Giá trị các tham số không được truyền ra ngoài.	- Có thể đưa giá trị của tham số ra ngoài bằng thuộc tính OUTPUT
- Có thể trả về một giá trị vô hướng hoặc một bảng dữ liệu.	- Chỉ trả về kiểu DL giá trị kiểu vô hướng

Cảm ơn các em đã chú ý lắng nghe

Những em chưa chú ý vẫn được cảm ơn bình thường