Microsoft® SQL Server™ 2008

CHƯƠNG 5

PROGRAMMING IN TRANSACT_SQL

Giảng viên: Trần Thị Kim Chi

Microsoft® .net™

NỘI DUNG

1. Khái niệm
2. Biến - Variables
3. Gói lệnh - Batches
4. Transact-SQL Scripts
5. Các cấu trúc lệnh - Control-of-Flow Statements
6. Raiserror

1 Khái niệm lập trình trong SQL

- **Lập trình CSDL:** Giao tiếp với chương trình ứng dụng
  - Chương trình bao gồm: Biến (variable), câu lệnh SQL và cấu trúc điều khiển.
- Các khái niệm cơ bản:
  - Định danh (Identifiers)
  - Batch (tập các câu lệnh T-SQL liên tiếp kết thúc bằng lệnh GO)
  - Script

1 IDENTIFIERS_ĐỊNH DANH

- Tên của các đối tượng đều được gọi là định danh. Trong SQL Server, có các định danh như Server, Databases, object of Database as Table, View, Index, Constraint,…
- Qui tắc định danh
  - Tối đa 128 ký tự.
  - Bắt đầu là một ký tự từ A_Z
  - Bắt đầu là một ký hiệu @, # sẽ có một ý nghĩa khác.
  - Những định danh nào có dấu khoảng trắng ở giữa thì phải kẹp trong dấu [] hoặc “ “
  - Đặt định danh sao cho ngắn gọn, đầy đủ ý nghĩa, phân biệt giữa các đối tượng với nhau, không trùng lặp, không trùng với từ khóa của T-SQL.

2 Biến - Variables

- Biến là một đối tượng dùng để lưu trữ dữ liệu. Biến phải được khai báo trước khi dùng.
- Có 2 loại biến: cục bộ và toàn cục
- Biến cục bộ:
  - Được khai báo trong phần thân của một bó lệnh hay một thủ tục.
  - Phạm vi hoạt động của biến bắt đầu từ điểm mà nó được khai báo cho đến khi kết thúc một bó lệnh, một thủ tục hay một hàm mà nó được khai báo.
  - Tên của biến bắt đầu bằng @

2 Biến cục bộ-Local Variables

- **Local variable**
  - **Khai báo (Declare):**

DECLARE@ VariableName var_type

  - **Example**:

DECLARE @EmpIDVar int

DECLARE @CustID Char(5), @name varchar(50)

2 Biến cục bộ-Local Variables

- **Sử dụng biến cục bộ: Assign value for the variable**: When a variable is declared, its value

```
SET @VariableName = expression
or
SELECT{@VariableName=expression} [,...n]
```

– Example:

```
DECLARE  @temp_city  varchar(10)
SET  @temp_city  = 'london'
SELECT * FROM  Customers
WHERE   city = @temp_city
```

2 Biến cục bộ-Local Variables

- **Sử dụng biến cục bộ:**

Example :

```
DECLARE @manv int
SET @manv = 2
Go
SELECT * FROM Employees
        WHERE Emloyeeid = @manv
```

```
DECLARE @manv int, @country nvarchar(15)
SET @manv = 3
Set @country =‘Usa’
SELECT * FROM Employees
        WHERE Emloyeeid = @manv and country =@country
```

2 Biến cục bộ-Local Variables

- **Hiển thị giá trị của biến cục bộ**:

PRINTF @VariableName | expression
or
SELECT @VariableName | expression

– Example:

```
DECLARE  @temp_city  varchar(10)
SET  @temp_city  = 'london'
SELECT * FROM  Customers
WHERE   city = @temp_city
```

2 Biến cục bộ-Local Variables

Example 1 :

Write a script that declares an integer variable called @ID. Assign the value 70000 to the variable. Use the variable in a SELECT statement that returns all the SalesOrderID values from the Sales.SalesOrderHeader table that have a SalesOrderID greater than the value of the variable.

```
DECLARE @ID INTEGER = 70000;
SELECT SalesOrderID
FROM Sales.SalesOrderHeader
WHERE SalesOrderID > @ID;
```

2 Biến cục bộ-Local Variables

Example 2 :

Write a script that declares two integer variables called @MaxID and @MinID. Use the variables to print the highest and lowest SalesOrderID values from the Sales.SalesOrderHeader table.

```
DECLARE @MaxID INT, @MinID INT;
SELECT @MaxID = MAX(SalesOrderID),
@MinID = MIN(SalesOrderID)
FROM Sales.SalesOrderHeader;
PRINT 'Max: ' + CONVERT(VARCHAR(10),@MaxID);
PRINT 'Min: ' + CONVERT(VARCHAR(10), @MinID);
```

2 Biến cục bộ-Local Variables

Example 3 :

Write a script that declares three variables, one integer variable called @ID, an NVARCHAR(50) variable called @FirstName, and an NVARCHAR(50) variable called @LastName. Use a SELECT statement to set the value of the variables with the row from the Person.Person table with BusinessEntityID = 1. Print a statement in the "BusinessEntityID: FirstName LastName" format.

```
DECLARE @ID INT, @FirstName NVARCHAR(50), @LastName NVARCHAR(50);
SELECT @ID = BusinessEntityID, @FirstName = FirstName, @LastName = LastName
FROM Person.Person
WHERE BusinessEntityID = 1;
PRINT CONVERT(NVARCHAR(10),@ID) + ': ' + @FirstName + ' ' + @LastName;
```

2 Biến cục bộ-Local Variables

Example 4 :

Write a script that declares an integer variable called @SalesCount. Set the value of the variable to the total count of sales in the Sales.SalesOrderHeader table. Use the variable in a SELECT statement that shows the difference between the @SalesCount and the count of sales by customer.

```
DECLARE @SalesCount INT;
SELECT @SalesCount = COUNT(*)
FROM Sales.SalesOrderHeader;
SELECT @SalesCount - COUNT(CUSTOMERID) AS CustCountDiff
FROM Sales.SalesOrderHeader
```

2 Biến toàn cục-Global Variables

- Biến toàn cục trong SQL là một hàm hệ thống.
  - Giá trị trả về của hàm được hiển thị bởi câu lệnh **SELECT @@Variablename.**
  - Không gán giá trị cho biến toàn cục.
  - Biến toàn cục không có kiểu
  - Tên biến được bắt đầu với @@.

2 Biến toàn cục-Global Variables

- **Một số biến toàn cục thông dụng**
  - **@@SERVERNAME**: trả về tên của server
  - **@@ROWCOUNT**: số dòng chịu tác dụng của câu lệnh cuối cùng.
  - **@@ERROR**: trả về chỉ số index của lỗi
  - **@@IDENTITY**: trả về định danh

2 Biến toàn cục-Global Variables

Example

```
How many are transaction opening
If (@@Trancount>0)
Begin
        Raiserror (‘Take can not be executed within a trasaction’,10,1)
        Return
End
```

```
Update Person.Person set LastName = 'Brooke'
Where LastName ='Duffy'
If(@@rowcount !=0)
begin
        print 'There are rows were updated'

End
select *from Person.Person
```

2 Biến toàn cục - Global Variables

Example

```
Tra ve so Identitidey phat sinh sau cung
Create table hd (mahd int identity Primary key, ghichu
varchar(20))

Create table cthd(Mahd int, masp char(10), soluong int)

insert into hd Values ('Record 1')
insert into hd Values ('Record 2')

Declare @maso int
Set @maso = @@identity
insert into cthd Values (@maso,'sp001',5)
insert into cthd Values (@maso,'sp002',12)

Select * from hd
Select * from cthd
```

2 Lệnh thực thi

Execution of the SQL statement

EXEC [USE] ({*@string_variable*| [N] '*tsql_string*'} [+ ...*n*])

Example:

Ngày 01 tháng 08 năm 2015

DECLARE @vname varchar(20), @vtable varchar(20), @vdbase varchar(20)

SET @vname='Brooke'

SET @vtable='Person.Person'

SET @vdbase='AdventureWorks2008R2'

EXECUTE ('USE'+@vdbase + 'SELECT * FROM '+ @vtable + 'WHERE lastname='+@vname)

3 Gói lệnh - Batches

2. **Gói lệnh (Batch)**
   - Bao gồm các phát biểu T-SQL và kết thúc bằng lệnh GO.
   - Các lệnh trong gói lệnh sẽ được biên dịch và thực thi cùng một lúc.
   - Nếu một lệnh trong Batch bị lỗi thì batch cũng xem như lỗi
   - Các phát biểu Create bị ràng buộc trong một batch đơn.

```
Ex :    use northwind
        select * from Customers
GO
```

3. **Kịch bản (Script)**
   - Một kich bản là một tập của một hay nhiều bó lệnh được lưu lại thành một tập tin .SQL

3 Gói lệnh - Batches

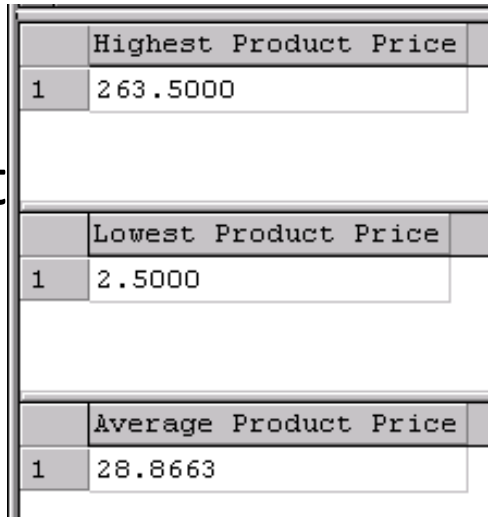
- Example:

```
use master
if exists(select * from sysdatabases where name like
      'sales') drop database sales
go
create database  sales
on
( name = sales_data, filename ='e:\sales_data.mdf', size
      = 1, maxsize = 5, filegrowth =1)
log on
( name = sales_log, filename ='e:\sales_log.ldf', size = 1,
      maxsize = 2, filegrowth =1)
```

3 Gói lệnh - Batches

- Example:

```
USE NorthWind
GO
SELECT MAX(Unitprice) AS ‘Highest Product Price’
FROM Products
SELECT MIN(Unitprice) AS ‘Lowest Product Price’
FROM Products
SELECT AVG(Unitprice) AS ‘Average Product Price’
FROM Products
GO
```



	Highest Product Price
1	263.5000

	Lowest Product Price
1	2.5000

	Average Product Price
1	28.8663

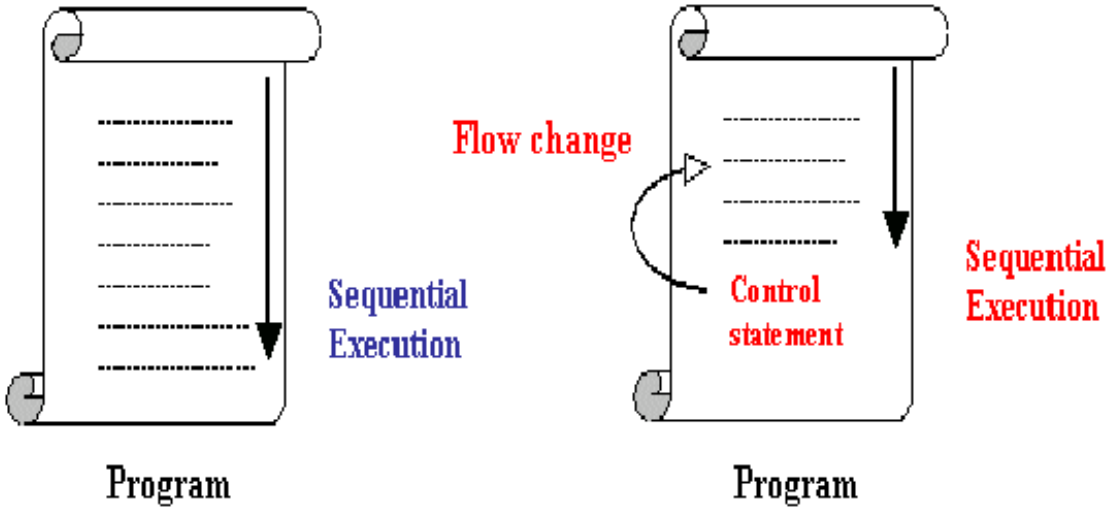
5 Control-of-Flow Statements

–BEGIN … END

–The IF … ELSE Statement

–The WHILE Statement

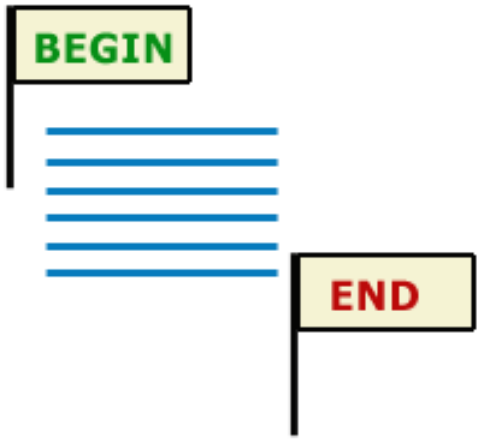
– The CASE Function



6 Control-of-Flow Statements

- **BEGIN …END**

```
BEGIN
    {sql_statement | statement_ block}
END
```



6 Control-of-Flow Statements



- **IF … ELSE**

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

Example :

```
If (select Count(*) From Customers where Country =‘Germany’)>0
        print ‘Co khach hang o Germany’
Else
        print ‘Khong co khach hang o Germany’
```

6 Control-of-Flow Statements



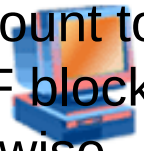
- **IF … ELSE**

```
if (select sum(soluong)
    from chitiethoadon
    where masach='s001')<20
    begin
        print N'Còn sách trong kho'
    end
else
    print N'Đã hết'
```

6 Control-of-Flow Statements

- **IF … ELSE**

Example :Write a batch that declares an integer variable called @Count to save the count of all the Sales.SalesOrderDetail records. Add an IF block that that prints “Over 100,000” if the value exceeds 100,000. Otherwise, print “100,000 or less.”



```
DECLARE @Count INT;
SELECT @Count = COUNT(*) FROM Sales.SalesOrderDetail;
IF @Count > 100000 BEGIN
        PRINT 'Over 100,000';
END
ELSE BEGIN
        PRINT '100,000 or less.';
END;
```

6 Control-of-Flow Statements

- **IF … ELSE**

Example 2 :Viết 1 đoạn batch, xuất cho biet nam nay la nam chan hay le

```
IF YEAR(GETDATE()) % 2 = 0 BEGIN
        PRINT 'The year is even.';
END
ELSE BEGIN
        PRINT 'The year is odd.';
END
```

6 Control-of-Flow Statements

- **IF … ELSE**

Example 3: Write a batch that uses IF EXISTS to check to see whether there is a row in the Sales.SalesOrderHeader table that has SalesOrderID = 1. Print “There is a SalesOrderID = 1” or “There is not a SalesOrderID = 1” depending on the result.

```
IF EXISTS(SELECT * FROM Sales.SalesOrderHeader
WHERE SalesOrderID = 1) BEGIN
        PRINT 'There is a SalesOrderID = 1';
END
ELSE BEGIN
        PRINT 'There is not a SalesOrderID = 1';
END;
```

6 Control-of-Flow Statements

- **CASE**
  - **Simple CASE function**

```
CASE input_expression
    WHEN when_expression THEN result_expression [ ...n ]
    [ELSE else_result_expression ]
END
```

6 Control-of-Flow Statements



– Searched CASE function

```
CASE
        WHEN Boolean_expression THEN
            result_expression [ ...n ]
        [ ELSE else_result_expression ]
END
```

6 Control-of-Flow Statements



- **Example 1 :Viết đoạn batch nhập 2 biến, xuất hiệu theo 2 trường hợp: nếu a>b hiệu là a-b, ngược lại là b-a**

```
Declare @a int, @b int, @Hieu int
Set @a = 15
Set @b =27
Set @hieu = Case
            When @a<@b then @b-@a
            When @a>@b then @a-@b
            else 0
        end
print 'hieu='+convert(varchar(20),@hieu)
```

6 Control-of-Flow Statements



- Example 2 :

```
Select ProductName, Unitprice,
                                    'Classification'=CASE
        when Unitprice<10 then 'Low price'
        When Unitprice Between 10 and 20 then
                                    'Moderately Price'
        when Unitprice>20 then 'Expensive'
        else 'Unknown'
    end
From Products
```

6 Control-of-Flow Statements



Example: Liệt kê danh sách các nhân viên trong Table HumanResources.Employee gồm BusinessEntityID, NationnalIDNumber, JobTitle. Trong đó dùng CASE để hiển thị “Even” khi BusinessEntityID là số chẵn hoặc “Odd” khi BusinessEntityID là số lẻ. Hướng dẫn: sử dụng phép toán phần tram để chia lấy phần dư.

```
SELECT BusinessEntityID,
CASE BusinessEntityID % 2 WHEN 0 THEN 'Even' ELSE
'Odd' END
FROM HumanResources.Employee;
```

6 Control-of-Flow Statements



Example 2: Liệt kê danh sách các thông tin chi tiết của các hóa đơn có trong bảng Sales.SalesOrderDetail gồm các thông tin SalesOrderID, OrderQty, trong đó OrderQty hiển thị giá trị “Under 10” OrderQTY nhỏ hơn 10, “10–19” nếu OrderQty từ 10-19, hoặc “20–29” nếu OrderQty từ 20-29, hoặc “30–39” nếu OrderQty từ 30-39, hoặc “40 and over” nếu OrderQty từ 40 trở lên.

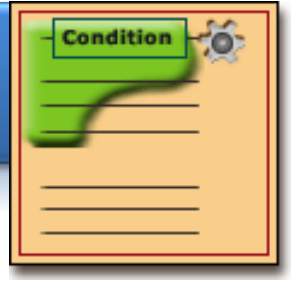
```
SELECT SalesOrderID, OrderQty,
CASE WHEN OrderQty BETWEEN 0 AND 9 THEN 'Under 10'
WHEN OrderQty BETWEEN 10 AND 19 THEN '10-19'
WHEN OrderQty BETWEEN 20 AND 29 THEN '20-29'
WHEN OrderQty BETWEEN 30 AND 39 THEN '30-39'
ELSE '40 and over' end AS range
FROM Sales.SalesOrderDetail;
```

6 Control-of-Flow Statements

- GOTO redirects the flow of program execution to a specified location (label)
- Example

```
Declare @a int, @b int, @Hieu int
Set @a = 39
Set @b =10
hieu_loop:
    if @a>@b
    begin
        Set @hieu =@A-@B
        print 'a='+convert(varchar(20),@a)
        print 'b='+convert(varchar(20),@b)
        print 'hieu='+convert(varchar(20),@hieu)
        Set @a =@hieu
        goto hieu_loop
        print 'a='+convert(varchar(20),@a)
        print 'b='+convert(varchar(20),@b)
        print 'hieu='+convert(varchar(20),@hieu)
    end
```

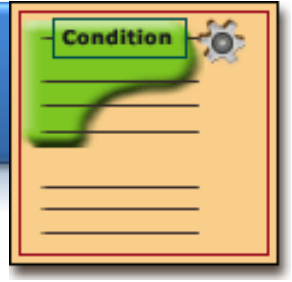
6 Control-of-Flow Statements



- **WHILE**

```
WHILE boolean_expression
        {sql_statement | statement_block}
        [BREAK]
        {sql_statement | statement_block}
        [CONTINUE]
```

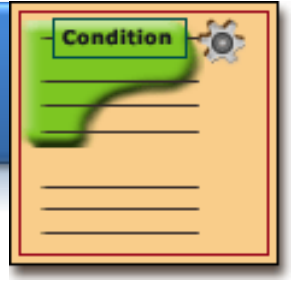
6 Control-of-Flow Statements



- **WHILE**

```
DECLARE @counter INT
SET @counter=0
WHILE (@counter<20)
BEGIN
    INSERT INTO nhomsach1
    VALUES ('N00'+CAST(@counter as char(2)),
            N'Sách_'+CAST(@counter as char(2)))
    SET @counter=@counter+1
END
```

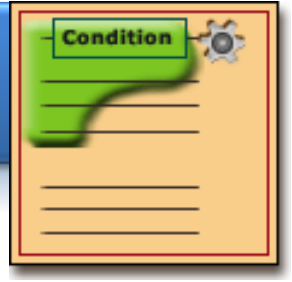
6 Control-of-Flow Statements



- **Example 1**: Write a script that contains a WHILE loop that prints out the letters *A* to *Z*. Use the function CHAR to change a number to a letter. Start the loop with the value 65.

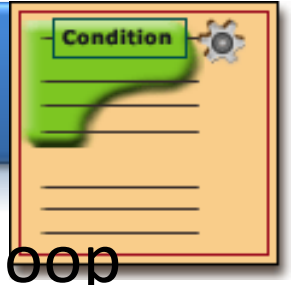
```
DECLARE @Letter CHAR(1);
SET @Letter = CHAR(65);
PRINT @Letter;
DECLARE @Count INT = 65;
WHILE @Count < 91 BEGIN
        PRINT CHAR(@Count);
        SET @Count += 1;
END;
```

6 Control-of-Flow Statements



- **Example 2**: Viết vòng lặp in ra bảng cửu chương của một số nào đó

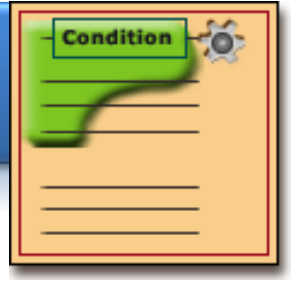
6 Control-of-Flow Statements



- **Example 4**: Write a script that contains a WHILE loop that counts up from 1 to 100. Print “Odd” or “Even” depending on the value of the counter.

```
DECLARE @Count INT = 1;
WHILE @Count <= 100
BEGIN
   IF @Count % 2 = 0
   BEGIN
            PRINT 'Even';
   END
   ELSE
   BEGIN
            PRINT 'Odd';
  END
SET @Count += 1;
```

6 Control-of-Flow Statements



Example 2:

```
WHILE (SELECT AVG(price) FROM titles) < $30
BEGIN
    UPDATE titles  SET price = price * 2
    SELECT MAX(price) FROM titles
    IF (SELECT MAX(price) FROM titles) > $50
        BREAK
    ELSE
        CONTINUE
END
PRINT 'Too much for the market to bear'
```

6 Control-of-Flow Statements

@@ERROR

- Write a statement that attempts to insert a duplicate row into the HumanResources.Department table. Use the @@ERROR function to display the error.

```
DECLARE @Error INT;
INSERT INTO
HumanResources.Department(DepartmentID,Name,Group
Name,ModifiedDate)
VALUES (1,'Engineering','Research and
Development',GETDATE());
DECLARE @Error char(30)
SET @Error = @@ERROR;
IF @Error > 0 BEGIN
        PRINT @Error;
END;
```

Control-of-Flow Statements

TRY… CATCH

- Provides error handling for T-SQL that is similar to the exception handling in the C# / Java
- Syntax:

```
BEGIN TRY
    { sql_statement | statement_block }
END TRY
BEGIN CATCH
    [ { sql_statement | statement_block } ]
END CATCH
```

6 Control-of-Flow Statements

TRY… CATCH

Example: Change the code you wrote in question 1 to use TRY…CATCH. Display the error number, message, and severity.

```
BEGIN TRY
        INSERT INTO
        HumanResources.Department(DepartmentID,Name,GroupName,Mo
        difiedDate)
        VALUES (1,'Engineering','Research and Development',GETDATE());
END TRY
BEGIN CATCH
        SELECT ERROR_NUMBER() AS ErrorNumber,ERROR_MESSAGE() AS
        ErrorMessage,
        ERROR_SEVERITY() AS ErrorSeverity;
END CATCH;
```

6 Control-of-Flow Statements

TRY… CATCH

- Example: Change the code you wrote in question 2 to raise a custom error message instead of the actual error message.

```
BEGIN TRY
    INSERT INTO
    HumanResources.Department(DepartmentID,Name,Grou
    pName,ModifiedDate) VALUES (1,'Engineering','Research
    and Development',GETDATE());
END TRY
BEGIN CATCH
    RAISERROR('You attempted to insert a duplicate!',16,1);
END CATCH;
```

6 Control-of-Flow Statements

RETURN

- Exits unconditionally from a query or procedure
- This will be discussed more detail in Stored Procedure section.
- Syntax

RETURN [integer_expression]

6 Control-of-Flow Statements

PRINT ‘any ACII Text’|@local_variable|
@@FUNTION| String_expr

RETURN [integer_expression]
integer_expression : return value

WAITFOR { DELAY '*time*' | TIME '*time*' }

6 Control-of-Flow Statements

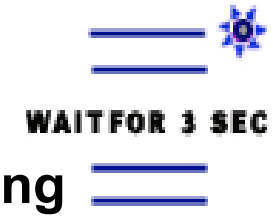
- WAITFOR: SQL Server tạm dừng một thời gian trước khi xử lý tiếp các phát biểu sau đó.
- Cú pháp :

  WAITFOR {DELAY 'time' |TIME 'time'}

  Time : hh:mm:ss

  Deplay 'time': hệ thống tạm dừng trong khoảng thời gian time

  TIME 'time': hệ thống tạm dừng trong khoảng thời gian time chỉ ra



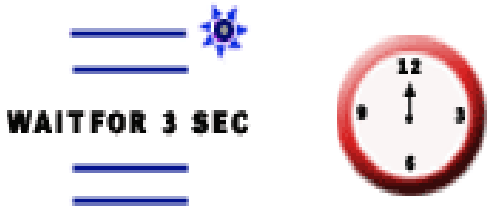
Ví dụ

```
WAITFOR DELAY '00:00:02'
SELECT EmployeeID FROM
    Northwind.dbo.Employees
```

6 Control-of-Flow Statements

- **Example**:

```
BEGIN
    WAITFOR TIME '22:20'
    EXECUTE update_all_stats
END
```



6 Control-of-Flow Statements

- **RAISERROR**

```
RAISERROR({msg_id | msg_str}
{ , severity , state }
[ , argument [ ,...n ] ] )
[ WITH option [ ,...n ] ]
```

6 Control-of-Flow Statements

- **RAISERROR**

```
RAISERROR({msg_id | msg_str}
{ , severity , state }
[ , argument [ ,...n ] ] )
[ WITH option [ ,...n ] ]
```

Msg_id : Là thông báo, nó được lưu trong bảng sysmessage.
Mã thông báo của người dùng phải bắt đầu từ trên 50000
Msg_str : Nội dung thông báo, tối đa 400 ký tự.
Để truyền tham số vào trong thông báo thì dùng dạng %<Loại ký tự>
Loại ký tự là d,I,o,x,X hay u

RAISERROR

Severity Levels: Mức lỗi của một thông báo lỗi cung cấp một sự biểu thị loại vấn đề mà SQL Server gặp phải.

- Mức lỗi **10** là lỗi về thông tin và biểu thị nguyên nhân do thông tin nhập vào.
- Mức lỗi từ **11 đến 16** thì thông thường là do các user.
- Mức từ **17 đến 25** do lỗi phần mềm hoặc phần cứng. Bạn nên báo cho nhà quản trị hệ thống bất cứ khi nào sự cố xảy ra. Nhà quả trị hệ thống phải giải quyết sự cố đó và theo dõi chúng thường xuyên. Khi mức lỗi 17,18,19 xảy ra, bạn có thể tiếp tục làm việc mặc dù bạn không thể thực thi lệnh đặc biệt.

Control-of-Flow Statements

RAISERROR

- *state*: Là một số nguyên tùy ý từ 1 đến 127 mô tả thông tin diễn giải về trạng thái lỗi.
- *Argument:* Là tham số dùng trong việc thay thế cho biến để định nghĩa thông báo lỗi hoặc thông báo tương ứng với mã lỗi msg_id. Có thể không hoặc có nhiều tham số. Tuy nhiên, không được quá 20. Mỗi tham số thay thế có thể là một biến local hoặc bất kỳ một trong các kiểu dữ liệu int, char, varchar, binary, varbinary. Các kiểu khác không được cung cấp.

6 Control-of-Flow Statements

- **RAISERROR**
  - Thêm một lỗi mới của người dùng định nghĩa

Syntax

```
Sp_AddMessage msg_id,
severity,'msg'[,'language'][,'with_log'][,'replace']
```

- **Msg_id:** Là thông báo, được lưu trong bảng sysmessage. Mã thông báo của người dùng phải bắt đầu từ trên 50000
- **Msg_str:** Nội dung thông báo, tối đa 400 ký tự.
- Để truyền tham số vào trong thông báo thì dùng dạng %<Loại ký tự>
- Loại ký tự là d,I,o,x,X hay u
- Lưu ý: số **float**, double, char không được hỗ trợ

Control-of-Flow Statements

RAISERROR

- Xóa một lỗi mới của người dùng định nghĩa

```
Sp_DropMessage msg_id
```

Các ký tự	Mô tả
d hoặc I	Biểu hiện là số nguyên (integer)
O	Octal không dấu
P	Con trỏ
S	Chuỗi
U	Số nguyên không dấu
x or X	Hexadecimal không dấu

RAISERROR

- ***msg_id***: là mã số của lỗi mới, là một số int, không được trùng các mã đã có sẵn, bắt đầu là 50001.
- *severity*: là mức lỗi của lỗi, là một số smallint. Mức hợp lệ là từ 1 đến 25. Chỉ có người quản trị CSDL mới có thể phát sinh thêm một thông báo lỗi mới từ 19 đến 25.
- **'*msg*':** là một chuỗi thông báo lỗi, tối đa 255 ký tự.
- **'*language*':** là ngôn ngữ của thông báo lỗi, không chỉ định thì mặc định là ngôn ngữ của phiên kết nối.

Control-of-Flow Statements

RAISERROR

- '***with_log***': thông báo lỗi có được gi nhận vào nhật ký của ứng dụng khi nó xảy ra hay không, mặc định là FALSE. Nếu là **true**, thì lỗi luôn luôn được ghi vào nhật ký ứng dụng. Chỉ có những thành viên thuộc **sysadmin** server role mới có thể sử dụng tham số này.
- '***replace***': nếu được chỉ định chuỗi **REPLACE**, thì thông báo lỗi đã tồn tại được ghi đè bởi chuỗi thông báo mới và mức lỗi mới. Tham số này phải chỉ định nếu *msg_id* đã có.
- Lưu ý: *nếu trả về 0 tức là thêm vào thành công, 1 thất bại.*

Control-of-Flow Statements

RAISERROR

- SP_ADDMESSAGE 50001,10,'KHONG TIM THAY MAU TIN %D TRONG %LS'
- SP_ADDMESSAGE 50002,16,'KHONG XOA DUOC %S VI %S CO TON TAI TRONG %LS'
- SP_ADDMESSAGE 50003,16,'MOT LOP CHI CO TOI DA %D HOC SINH'
- SP_ADDMESSAGE 50004,16,'DON GIA BAN PHAI LON HON DON GIA GOC'
- --XEM THONG BAO LOI VUA XAY DUNG(COI LAI SAI)
- SP_HELPTEXT 'SYSMESSAGE'
- SELECT * FROM SYSMESSAGE WHERE ERROR =50002

RAISERROR

```
--CAU 5 :XAY DUNG CAU THONG BAO LOI BANG RAISERROR
Use Northwind
RAISERROR (50001,10,1,4,'SANPHAM')
DECLARE @@MA INT
DECLARE @@TEN NVARCHAR
SET @@TEN ='SANPHAM'
SET @@MA =8
SELECT  productid FROM products WHERE
productid=@@MA
IF (@@ROWCOUNT=0)
  BEGIN
        RAISERROR (50001,10,1,@@MA,@@TEN)
  END
GO
```

THANKS YOU !