Chương 3
Lập trình T-SQL
Phần 1 – Batch & các cấu trúc điều khiển

Giáo trình & Tài liệu tham khảo:

1. Microsoft SQL Server 2008 R2 Unleashed, Ray Rankins, Paul Bertucci, Chris Gallelli, Alex T. Silverstein, 2011, Pearson Education, Inc

2. **MS SQL Server 2012 T-SQL fundamentals**, Tizik Ben-Gan

3. https://docs.microsoft.com/

Gv. Nguyễn Như Hoa

Giới thiệu

- T-SQL cung cấp khả năng lập trình
  - cho phép sử dụng biến, cấu trúc điều khiển, …
  - người dùng có thể tạo hàm, thủ tục
- Các hàm, thủ tục do user tạo ra được lưu trong database, giúp tăng khả năng dùng lại, tự động hóa tác vụ và cải thiện hiệu năng

Nội dung

- variables
- batches
- flow elements (if else, case, while)
- temporary tables
- routines such as user-defied functions, stored procedures, and triggers
- dynamic SQL

Biến

- **Biến** : variable
- Biến (local variables) có phạm vi sử dụng trong một đơn vị thực thi (batch, store procedure, function, trigger)
- Không có Biến toàn cục

Biến

local variables

- Khai báo biến và gán giá trị cho biến

```
DECLARE @i AS INT;
SET @i = 10;
```

Khai báo biến

Gán biến

hoặc

```
DECLARE @i AS INT = 10;
```

Ví dụ

```
declare @i int
set @i = 100
select @i
```

Xuất biến

Hoặc

Khai báo biến

Gán biến

```
declare @i int
select @i = 100
print @i
```

Xuất biến

Biến

local variables

- **Giá trị** gán cho biến có thể là scalar value hay scalar expression
- Ví dụ 1

```
DECLARE  @empname  AS  NVARCHAR(31);
SET @empname = ‘Ken Sachez’
SET @empname = ‘Ken’ + ‘ ’ + ‘Sachez’
SET @empname = (SELECT firstname + N' ' + lastname
                        FROM Person.Person
                        WHERE BusinessEntityID = 1) ;

SELECT @empname  AS  empname;

Go
```

Biến

local variables

- **Sử dụng lệnh SELECT**
  - gán biến
  - xuất/in giá trị của biến
- Ví dụ 2

```
DECLARE @firstname AS NVARCHAR(10), @lastname AS
                                        NVARCHAR(20);
SELECT @firstname = firstname,
       @lastname = lastname
FROM Person.Person
WHERE BusinessEntityID = 1;

SELECT @firstname AS firstname, @lastname AS lastname;
Go
```

Biến

local variables

- Dùng SET hay SELECT để gán biến => Nhận xét?
  - Lưu ý : khi SELECT trả về nhiều hơn 1 row
    - phép gán bằng SELECT không cảnh báo => giá trị row cuối cùng sẽ là giá trị của biến
    - Phép gán bằng SET sẽ báo lỗi trong trường hợp này

Biến
Lưu ý

- The names of some Transact-SQL system functions begin with two *at* signs (@@). Although in earlier versions of SQL Server, the @@functions are referred to as global variables, **@@functions aren't variables**, and they don't have the same behaviors as variables. The @@functions are system functions, and their syntax usage follows the rules for functions.

  https://docs.microsoft.com/#variables

- Ví dụ :
  - @@error
  - @@rowcount

Nội dung

- variables
- batches
- flow elements (if else, case, while)
- temporary tables
- routines such as user-defied functions, stored procedures, and triggers
- dynamic SQL

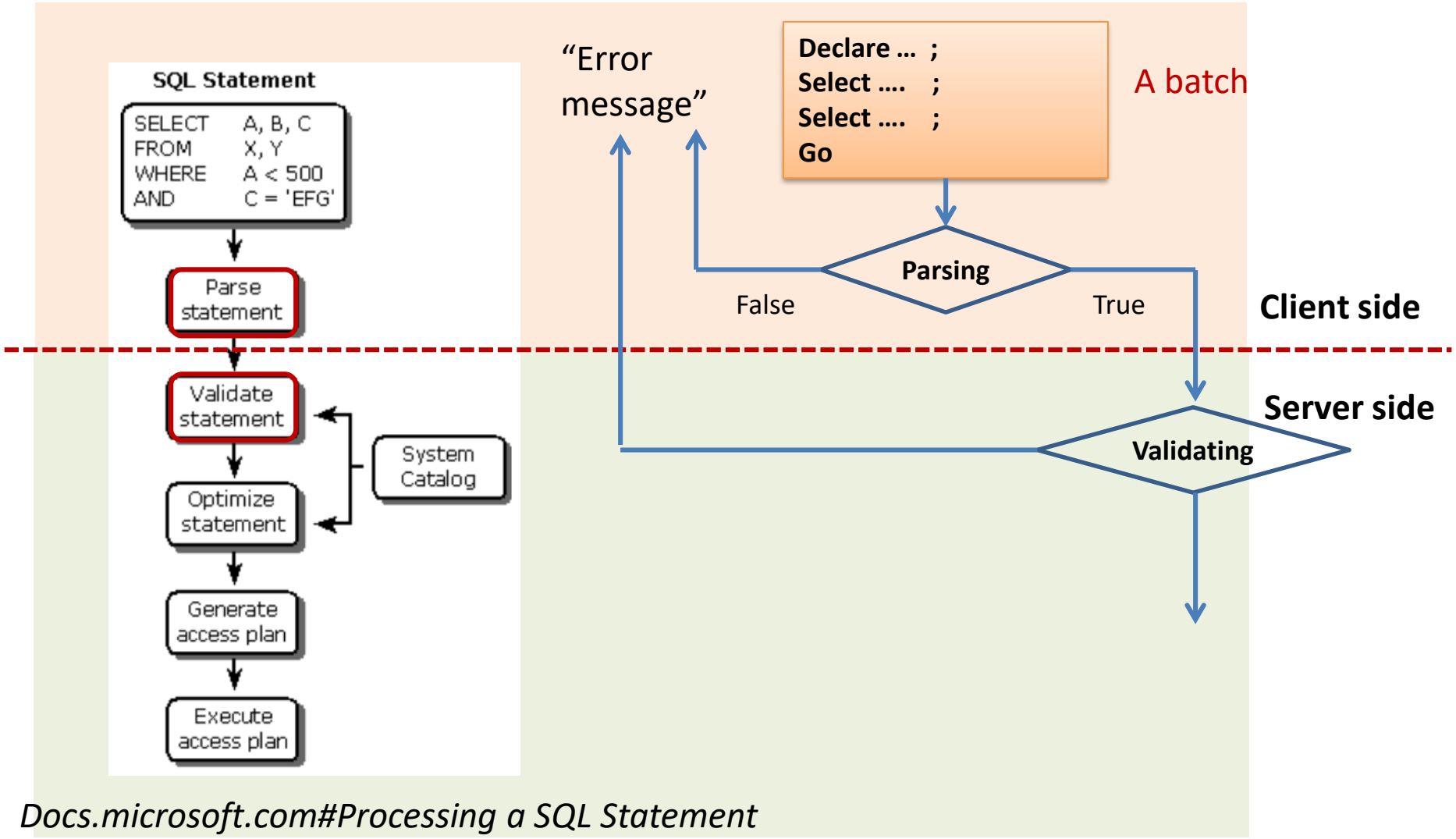
Batch

- **Một Batch** gồm một hay nhiều lệnh T-SQL, được gửi từ một Client App tới SQL Server và được thi hành như một đơn vị thực thi
- SSMS , SQLCMD sử dụng Go như một lệnh đánh dấu kết thúc batch
  - *có thể thay đổi thiết lập này*

- Nhiều batch có thể lưu vào file => **T-SQL scripts**

Batch

Nếu 1 lệnh trong batch sai cú pháp => ??



Docs.microsoft.com#Processing a SQL Statement

Batch

- Ưu điểm
  - This is often more efficient than submitting statements separately, because network traffic can often be reduced and the data source can sometimes optimize execution of a batch of SQL statements.

docs.microsoft.com#**Batches of SQL Statements**

Batch

Ví dụ 1:

```
DECLARE @i AS INT
SET @i = 10
PRINT @i
GO
SET @i = 100
PRINT 'new result ... '
PRINT @i
```

=> *Nhận xét :* Lỗi cú pháp

Batch

Ví dụ 2 :

```
-- 1 batch
ALTER TABLE Production.Product ADD   testcol   flag NULL;
UPDATE Production.Product
        SET  testcol= MakeFlag;
```

=>*Nhận xét* : lỗi run-time

```
-- 2 batch
ALTER TABLE Production.Product ADD   testcol   flag NULL;
GO
UPDATE Production.Product
        SET  testcol= MakeFlag;
```

Batch

Ví dụ 3:

```
PRINT 'First batch'
USE  AdventureWorks2008R2
GO
PRINT 'Second batch'
SELECT  BusinessEntityID  FROM  HumanResources.Employee
SELECT  BusinessEntityID  FOM  Person.Person
GO
PRINT 'Third batch'
SELECT BusinessEntityID   FROM  Sales.SalesPerson
```

=> Nhận xét : Khi có Lỗi cú pháp sẽ cancel toàn bộ batch

Batch

Ví dụ 4

```
IF OBJECT_ID('cities') IS NOT NULL
  DROP TABLE cities;
GO
CREATE TABLE cities (name VARCHAR(90) PRIMARY KEY)
PRINT 'Table created 1'

-- Table already exists (run-time error will be raised)
CREATE TABLE cities (name VARCHAR(90) PRIMARY KEY)
PRINT 'Table created 2'
GO
```

=> Nhận xét : Khi có lỗi run-time sẽ cancel từ lệnh bị lỗi đến cuối batch

http://www.sqlines.com/sql-server/sql_batch_go

Batch

- Các lệnh sau không thể kết hợp với lệnh khác trong cùng một batch : **CREATE VIEW**, **CREATE DEFAULT, CREATE RULE, CREATE FUNCTION, CREATE PROCEDURE, CREATE SCHEMA, CREATE TRIGGER**
- Ví dụ

```
IF OBJECT_ID('dbo.MyView', 'V') IS NOT NULL
    DROP VIEW Sales.MyView;

CREATE VIEW dbo.MyView
AS
SELECT YEAR(orderdate) as ordY, COUNT(*) AS numorders
FROM northwind.dbo.Orders
GROUP BY YEAR(orderdate);
GO
```

```
Msg 111, Level 15, State 1, Line 4
'CREATE VIEW' must be the first statement in a query batch.
```

Batch

- **Go n**
  - Chỉ định số lần thực thị batch
- Ví dụ

```
IF OBJECT_ID('dbo.T1', 'U') IS NOT NULL
      DROP TABLE dbo.T1;
CREATE TABLE dbo.T1 (col1 INT IDENTITY);
SET NOCOUNT ON;
GO

INSERT INTO dbo.T1 DEFAULT VALUES;
GO 100

SELECT *  FROM dbo.T1;
```

```
Beginning execution loop
Batch execution completed 100 times.
```

Nội dung

- variables
- batches
- flow elements (if else, case, while)
- temporary tables
- routines such as user-defied functions, stored procedures, and triggers
- dynamic SQL

Các cấu trúc điều khiển

- If else
- Case
- While

If else

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

- ***statement_block*** được đánh dấu bởi cặp BEGIN … END

If else
Ví dụ 1

```
DECLARE @Count INT;
SELECT @Count = COUNT(*) from sales.SalesOrderHeader
where CustomerID = 30115
IF @Count > 5
BEGIN
        PRINT   N‘Số lần mua hàng nhiều hơn 5 lần’
END
ELSE
BEGIN
        PRINT   N‘Số lần mua hàng ít hơn 6 lần’
END;
```

If else
Ví dụ 2

```
IF   (Select count(*) from sales.SalesOrderHeader
        where CustomerID = 30115)   > 5
        PRINT   N‘Số lần mua hàng nhiều hơn 5 lần’
ELSE
        PRINT   N‘Số lần mua hàng ít hơn 6 lần’
```

If else
Ví dụ 3

```
declare @id int
set @id = 1606
if exists (select * from Purchasing.PurchaseOrderHeader
                                    where VendorID = @id)
        print N'Có các  giao dich mua hàng với nhà cung cấp '+
        convert(varchar(5),@id)
 else
        print N'Không có các  giao dich mua hàng với nhà cung
        cấp '+ convert(varchar(5),@id)
```

Thay thể **if exists** bằng **if not exists** ?

Case

- Cú pháp 1

```
CASE expression
    WHEN value THEN result1
    WHEN value THEN result2
    ELSE else_result
END
```

Case

- Cú pháp 2

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

Case

- Ví dụ 1:

```
DECLARE @OrderID  INT
SET @OrderID = 43660
SELECT CASE (select OnlineOrderFlag
                        from sales.SalesOrderHeader
                        where SalesOrderID = @OrderID)
When 1  Then   N'Hóa đơn‘ + convert(varchar(5),@OrderID)
                        +  N'được đặt hàng online'
When 0  Then  N'Hóa đơn‘ + convert(varchar(5),@OrderID)
                        + N'được tạo trực tiếp'
END
```

Case

- Ví dụ 2 :

```
DECLARE @custID INT
SET @custID = 30115
SELECT CASE
    When (select COUNT(*) from sales.SalesOrderHeader
                          where CustomerID = @custID ) < 5
        Then  N'Số lần mua hàng nhỏ hơn 5'
    When (select COUNT(*) from sales.SalesOrderHeader
                          where CustomerID = @custID) < 10
        Then  N'Số lần mua hàng nhỏ hơn 10'
    else  N'Số lần mua hàng lớn hơn 10'
        END
```

Case

- Ví dụ 3:

```
Select SalesOrderID, CustomerID, TotalDue,
'period'= case
        when (ShipDate-OrderDate) <=7
                then   'in a week'
        when (ShipDate-OrderDate) between 8 and 14
                then    'in two weeks'
        else  '-'
        end
From sales.SalesOrderHeader
```

Case

- Sử dụng case expression, trong
  - lệnh update
  - Lệnh select thống kê
  - Sắp xếp
- Tham khảo https://www.mssqltips.com/sqlservertip/1455/using-the-case-expression-instead-of-dynamic-sql-in-sql-server/

while

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

While
Ví dụ 1

```
-- In danh sách các chữ cái A -> Z
DECLARE @Letter CHAR(1);
SET @Letter = CHAR(65);
PRINT @Letter;
DECLARE @Count INT = 66;
WHILE @Count < 91
BEGIN
    PRINT CHAR(@Count);
    SET @Count += 1;
END;
```

-- CHAR() function converts an **int** ASCII code to a character value.

While
Ví dụ 2

```
Declare @counter int
Set @Counter = 1
WHILE (@Counter <5)
    begin
        INSERT  INTO Production.ProductCategory(Name)
        VALUES  (N‘Cat ' + CAST(@counter as char(2)))

        SET @Counter=@counter + 1
    end
GO
Select * from Production.ProductCategory
```

While - Ví dụ 3

```
ALTER  TABLE  products  ADD  StockPlan int
GO
UPDATE  products  SET  StockPlan = UnitsInStock
GO
WHILE (SELECT AVG(StockPlan) FROM Products) < 100
BEGIN
      UPDATE  Products SET StockPlan  =  StockPlan* 2
      SELECT  MAX(StockPlan)  FROM Products
      IF  (SELECT  MAX(StockPlan)  FROM Products) > 200
          BREAK
      ELSE
          CONTINUE
END
SELECT  UnitsInStock , StockPlan FROM  Products
ALTER  TABLE  products  DROP   COLUMN  StockPlan
```

Kiểm soát lỗi

- Thông thường, lỗi phát sinh khi thực thi các T-SQL script -> hệ thống sẽ tự động xử lý và trả về thông báo lỗi

```
print 10/0
```

```
Msg 8134, Level 16, State 1, Line 1
Divide by zero error encountered.
```

- User có thể tự kiểm soát lỗi, dựa trên
  - Cấu trúc Try …. Catch
  - Sử dụng các hàm có sẵn cung cấp thông tin lỗi

Kiểm soát lỗi
Try … catch

- Cấu trúc bắt lỗi
  - Nếu không có lỗi xuất hiện trong Try block, các lệnh trong Catch block được bỏ qua
  - Nếu có lỗi xuất hiện trong Try block => điều khiển sẽ chuyển tới các câu lệnh trong Catch block

Try block

```
BEGIN TRY
    PRINT 10/2;
    PRINT 'No error';
END TRY
```

Catch block

```
BEGIN CATCH
    PRINT 'Error';
END CATCH;
```

```
BEGIN TRY
    PRINT 10/0;
    PRINT 'No error';
END TRY
```

```
BEGIN CATCH
    PRINT 'Error';
END CATCH;
```

Kiểm soát lỗi
Cung cấp thông tin lỗi

- SQL Server hỗ trợ tập các hàm cung cấp thông tin về lỗi
  - Error_number() : mã lỗi
  - Error_message() : thông báo lỗi
  - Error_severity() : mức độ lỗi
  - Error_state() : trạng thái lỗi
  - Error_line() : dòng code bị lỗi
  - Error_procedure() : đoạn code lỗi có nằm trong Proc không

Kiểm soát lỗi
Cung cấp thông tin lỗi

- **sys.messages** Là một system view cung cấp danh sách tất cả các error messages , gồm system và user-defined messages

  select * from sys.messages

- Ví dụ :

  select * from sys.messages

  where message_id = 2627

Kiểm soát lỗi
Cung cấp thông tin lỗi

- Severity level :
  - **Từ 0-> 18** : các lỗi ở những level này có thể được chỉ định bởi user thông thường
  - **Từ 19-> 25** : dùng với WITH LOG , được xác định bởi các user thuộc sysadmin role , tương ứng với các lỗi nghiêm trọng. Khi lỗi ở mức này xãy ra, the client connection is terminated after receiving the message, and the error is logged in the error and application logs.

Ví dụ

Ví dụ

```
BEGIN TRY
    PRINT 10/0;
    PRINT 'No error';
END TRY
BEGIN CATCH
  PRINT ' Error Number : ' +  CAST(ERROR_NUMBER() AS VARCHAR(10));
  PRINT ' Error Message : ' +  ERROR_MESSAGE();
  PRINT ' Error Severity: ' +  CAST(ERROR_SEVERITY() AS VARCHAR(10));
  PRINT ' Error State : ' +  CAST(ERROR_STATE() AS VARCHAR(10));
  PRINT ' Error Line : ' +  CAST(ERROR_LINE() AS VARCHAR(10));

END CATCH;
```

Ví dụ

Dùng Raiserror() đưa ra thông báo lỗi do user định nghĩa

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

```
DECLARE
   @ErrorMessage  NVARCHAR(4000),
   @ErrorSeverity INT,
   @ErrorState    INT;
BEGIN TRY
   PRINT 10/0;
   PRINT 'No error';
END TRY
BEGIN CATCH
   SELECT
      @ErrorMessage = ERROR_MESSAGE(),
      @ErrorSeverity = ERROR_SEVERITY(),
      @ErrorState = ERROR_STATE();

   -- return the error inside the CATCH block
   RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);
END CATCH;
```

Ví dụ

```
USE master;
GO
EXEC sp_addmessage 50001, 16,
   N'user defined error messages....';
GO
```

```
DECLARE
   @ErrorMessage  NVARCHAR(4000),
   @ErrorSeverity INT,
   @ErrorState    INT;
BEGIN TRY
   PRINT 10/0;
   PRINT 'No error';
END TRY
BEGIN CATCH
   SELECT
      @ErrorMessage = ERROR_MESSAGE(),
      @ErrorSeverity = ERROR_SEVERITY(),
      @ErrorState = ERROR_STATE();

   -- return the error inside the CATCH block
   RAISERROR(50001, @ErrorSeverity, @ErrorState);
END CATCH;
```

Ví dụ

- Raiserror cho phép tạo các dynamic message

```
RAISERROR (N'This is message %s %d.',          -- Message text
                    10,                        -- Severity
                    1,                         -- State
                    N'number',                 -- First argument.
                    5 );                       -- Second argument.
GO
```

=> The message text returned is: This is message number 5.

Nội dung

- variables
- batches
- flow elements (if else, case, while)
- temporary tables
- routines such as user-defied functions, stored procedures, and triggers
- dynamic SQL

Temporary table

- Tham khảo:
  - Microsoft SQL Server 2012 T-SQL Fundamentals, p.353

Dynamic SQL

- Tham khảo:
  - Microsoft SQL Server 2012 T-SQL Fundamentals, p.359

Tóm tắt

- variables
- batches
- flow elements (if else, case, while)
- temporary tables
- routines such as user-defied functions, stored procedures, and triggers
- dynamic SQL