


# LẬP TRÌNH T-SQL

# Nội dung

- ▶ Batch
- ▶ Stored Procedure
- ▶ User Defined Function

**Batch**

# BATCH

- ▶ Batch là một tập các câu lệnh T\_SQL được biên dịch đồng thời bởi SQL Server (chú ý được biên dịch đồng thời, nếu một lệnh có lỗi cú pháp thì cả batch không biên dịch được)
  - ▶ Các câu lệnh được gửi đồng thời đến server.
  - ▶ Câu lệnh GO đánh dấu kết thúc một batch.
- 

# BATCH

- ▶ Một số quy tắc với GO:
  - Nếu batch có bất kỳ một lỗi nào → toàn batch không được biên dịch
  - Các phát biểu CREATE (CREATE DATABASE, CREATE TABLE, CREATE VIEW) thường phải nằm riêng trong một batch.

# BATCH – Ví dụ 1

--Batch 1

```
select * from KHOA;  
select makhoa fom khoa;  
go
```

-- Batch 2

```
select tenkhoa from KHOA;  
go
```

-- Batch 1

```
select * from KHOA;  
select makhoa fom khoa;  
select tenkhoa from KHOA;  
go
```

# Batch – Ví dụ 2

```
IF OBJECT_ID('QLGV.MyView', 'V') IS NOT NULL DROP VIEW qlgv.MyView;  
CREATE VIEW MyView  
AS  
    SELECT YEAR(NGTLAP) AS NAMTL, COUNT(*) AS SLKHOA  
    FROM Khoa  
    GROUP BY YEAR(NGTLAP);  
GO
```

```
IF OBJECT_ID('QLGV.MyView', 'V') IS NOT NULL DROP VIEW qlgv.MyView;  
go  
CREATE VIEW MyView  
AS  
    SELECT YEAR(NGTLAP) AS NAMTL, COUNT(*) AS SLKHOA  
    FROM Khoa  
    GROUP BY YEAR(NGTLAP);  
GO
```

# BIẾN

- ▶ Khai báo biến:

DECLARE @<TenBien> <Kiểu> [(<kích thước>)]

- ▶ Biến là cục bộ trong một batch

- ▶ Gán giá trị cho biến bằng 2 cách:

**SET** @<TenBien> =<giá trị>;

**SELECT** @<TenBien> =<giá trị>;

- ▶ Ví dụ:

```
DECLARE @i AS INT;  
SET @i = 10;  
PRINT @i;           → Succeeds  
GO  
PRINT @i;           → Fails
```



# BIẾN

- ▶ Biến hệ thống có tên bắt đầu @@
- ▶ Biến hệ thống là biến toàn cục (Global Variables).
- ▶ Ví dụ:

```
PRINT @@SERVERNAME;  
SELECT @@MAX_CONNECTIONS AS 'Max Connections';
```

# CẤU TRÚC ĐIỀU KHIỂN

- ▶ Khởi **BEGIN...END**:
  - nhóm một số câu lệnh được thực thi với nhau trong các cấu trúc điều khiển.
- ▶ **IF** <biểu thức điều kiện>  
    {<câu lệnh SQL>|<khởi câu lệnh>}  
[**ELSE** [<biểu thức điều kiện>  
    {<câu lệnh SQL>|<khởi câu lệnh>}]]

# Ví dụ (cấu trúc IF ELSE)

```
DECLARE @kinhphi float;  
SELECT @kinhphi=budget  
FROM projects  
WHERE project_no='p1 ';  
IF @kinhphi>2000  
    PRINT N'Loại 1';  
ELSE  
    PRINT N'Loại 2';
```

# CASE WHEN

```
SELECT project_name,  
       CASE  
         WHEN budget > 0 AND budget < 100000 THEN 1  
         WHEN budget >= 100000 AND budget < 200000 THEN 2  
         WHEN budget >= 200000 AND budget < 300000 THEN 3  
       ELSE 4  
       END budget_weight  
FROM project
```

	project_name	budget_weight
1	Apollo	2
2	Gemini	1
3	Mercury	2

# CẤU TRÚC WHILE

**WHILE** <*biểu thức logic*>

[BEGIN

    <(các) câu lệnh>

    [BREAK]


    <(các) câu lệnh>

    [CONTINUE]

END]

# WHILE - Ví dụ

```
WHILE (SELECT AVG(BUDGET) FROM projects) < 4000
BEGIN
    UPDATE projects
    SET budget = budget * 2;
    IF (SELECT MAX(budget) FROM projects) > 1000
        BREAK
    ELSE
        CONTINUE
END
```



# Bài tập

1. Khai báo biến @n kiểu số nguyên
  - Gán giá trị 7 cho biến @n
  - Cho biết thông tin Project có budget lớn hơn giá trị @n
2. Cho biết kết quả của đoạn chương trình sau:

```
DECLARE @i AS INT = 0;  
WHILE @i < 10  
BEGIN  
    SET @i = @i + 1;  
    IF @i = 6 CONTINUE;  
    PRINT @i;  
END;
```


# Bài tập

3. Viết đoạn chương trình in danh sách các số lẻ nhỏ hơn 100
4. Sử dụng cấu trúc vòng lặp While, nhập 10 mẫu tin vào bảng employee với nội dung:
  - Emp\_no tăng từ 1 đến 10
  - Emp\_fname là 'nv1','nv2',...,'nv10'
  - Emp\_lname là 'Nguyen'
  - Dept\_no là 'd3'



# STORED PROCEDURE

# Giới thiệu

- ▶ SP được xây dựng từ các câu lệnh T-SQL và được lưu trữ trên SQL server.
  - ▶ Có thể chứa bất kỳ một câu lệnh SQL nào, kể cả lệnh gọi thực hiện một Stored Procedure.
  - ▶ Có thể nhận các tham số được truyền vào và phát sinh các tham số output.
  - ▶ Hiệu quả của việc sử dụng stored procedure cao hơn nhiều so với việc gửi từng câu SQL riêng lẻ tới Server yêu cầu xử lý.
  - ▶ Trong chương trình chỉ cần gọi thực hiện stored procedure.
  - ▶ Thay đổi một câu lệnh SQL bên trong một stored procedure dễ hơn thay đổi bên trong mã nguồn của chương trình.
- 

# Creation and Execution of Stored Procedures

```
CREATE PROC[EDURE] [schema_name.]proc_name  
[( {@param1} type1 [= default1] [OUTPUT])] {, ...}  
[WITH {RECOMPILE | ENCRYPTION | EXECUTE AS  
'user_name'}]  
AS batch
```

- ▶ **OUTPUT** : Tham biến
- ▶ **EXECUTE AS 'user\_name'** : Cho phép user được tham chiếu thủ tục.

# Ví dụ 1

```
USE sample;  
GO  
CREATE PROCEDURE increase_budget (@percent INT=5)  
AS  
    UPDATE project  
    SET budget = budget + budget* @percent/100;
```



- ▶ `USE sample;`
- ▶ `EXECUTE increase_budget 10;`

## Ví dụ 2

```
USE sample;  
GO  
CREATE PROCEDURE modify_empno (@old_no INTEGER,  
@new_no INTEGER)  
AS  
    UPDATE employee  
        SET emp_no = @new_no  
        WHERE emp_no = @old_no  
    UPDATE works_on  
        SET emp_no = @new_no  
        WHERE emp_no = @old_no
```

### Ví dụ 3: Tạo Procedure có tham số **output**

```
USE sample;  
GO  
CREATE PROCEDURE delete_emp @employee_no INT, @counter  
INT OUTPUT  
AS  
    SELECT @counter = COUNT(*)  
        FROM works_on  
        WHERE emp_no = @employee_no  
    DELETE FROM employee  
        WHERE emp_no = @employee_no  
    DELETE FROM works_on  
        WHERE emp_no = @employee_no
```

- ▶ DECLARE @quantity INT
- ▶ **EXECUTE** delete\_emp @employee\_no=28559, @counter=@quantity **OUTPUT**

# Changing the Structure of Stored Procedures

▶ **ALTER PROCEDURE** .....

Câu lệnh ALTER PROCEDURE thường được sử dụng để sửa đổi các lệnh Transact-SQL bên trong một thủ tục.

# Remove Stored Procedures

- ▶ **DROP PROCEDURE** .....



# Bài tập (sử dụng database QLGV đã tạo ở bài thực hành 1)

1. Tạo thủ tục KQ\_MH, cho biết kết quả thi môn học (MH) của học viên (SV).
  - Thông tin gồm: Mã học viên, tên học viên, mã môn học, tên môn học, lần thi, ngày thi, điểm, kết quả
  - Thủ tục nhận 2 tham số đầu vào là mã môn học (MH) và mã học viên (SV)
2. Tạo thủ tục GV\_MH, cho biết danh sách các môn học giáo viên (GV) đã dạy trong năm (NH).
  - Thông tin gồm: Mã GV, tên gv, mã môn học, tên môn học, số tín chỉ lý thuyết, số tín chỉ thực hành
  - Thủ tục nhận 2 tham số đầu vào là mã giáo viên (GV) và năm học (NH)
3. Tạo thủ tục MH\_TRUOC cho biết danh sách môn học tiên quyết của môn học (MH).
  - Thông tin gồm: mã môn học tiên quyết, tên môn học tiên quyết.
  - Thủ tục nhận 1 tham số đầu vào là mã môn học (MH).
4. Tạo thủ tục TOP\_N liệt kê danh sách n học viên có điểm thi cao nhất.
  - Thông tin gồm: Mã học viên, tên học viên, mã môn học, ngày thi, lần thi, điểm
  - Thủ tục nhận 1 tham số đầu vào là n.
5. Tạo thủ tục THONGKE, cho biết tổng số giáo viên (TS) và mức lương bình quân (BQ) của khoa (KH).
  - Trong đó mã khoa (KH) là tham số đầu vào, TS và BQ là tham số đầu ra

# User Defined Function

# Phân loại

- ▶ System Function
- ▶ User-defined Function


# System Function

- ▶ Hàm hệ thống
- ▶ Ví dụ:
  - Aggregate function: avg(), count(\*), sum(),max(),min()...
  - Other: getdate(), month(),year(),....

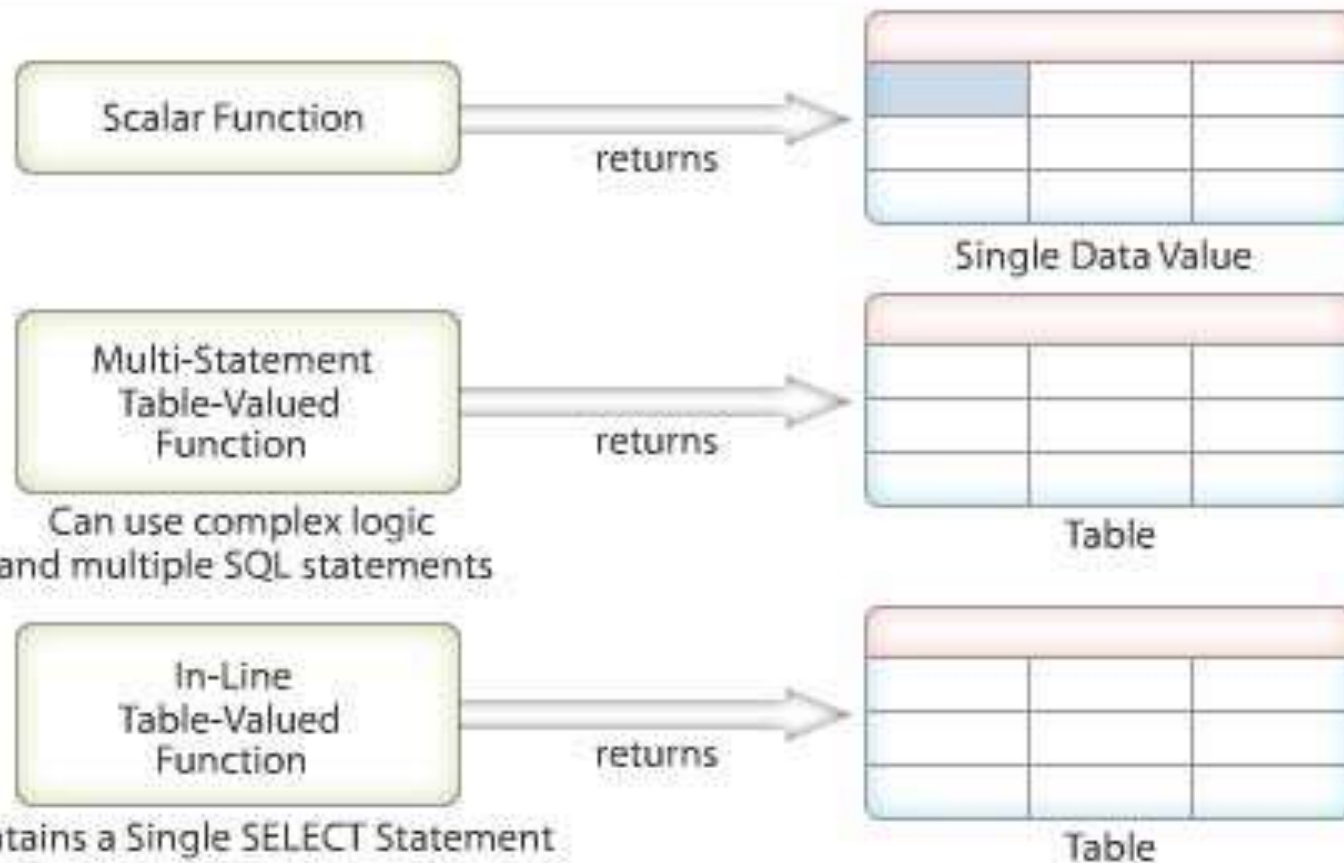
# User Defined Functions (UDF)

- ▶ Hàm tạo bởi người dùng
- ▶ UDF nhận vào không hoặc nhiều tham số và trả về giá trị đơn hoặc một bảng dữ liệu
- ▶ Có 3 loại UDF:
  - Scalar
  - Inline Table Valued
  - Multi statement Table Valued

# Phân loại UDF

- ▶ **Scalar Function**: kết quả trả về là một giá trị đơn.
  - ▶ **Multi-Statement Table-valued Functions**: kết quả trả về là một bảng được tạo thành từ nhiều câu lệnh SQL
  - ▶ **In-Line Table-valued Functions**: kết quả trả về là một bảng được tạo thành từ một câu SELECT
- 

# Phân loại UDF



# UDF Scalar syntax

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



Ví dụ:

Hàm trả về giá trị tổng hai số 4 và 6

```
create function tong2so()  
returns int  
as  
Begin  
    declare @so1 int, @so2 int  
    Set @so1=4  
    Set @so2=6  
    Return @so1 + @so2  
end
```

# Thực thi

- ▶ `select` `dbo.tong2so()` `as` `tong`
- ▶ `print` `'Tong 4 +6 = '` `+convert(char(10),dbo.tong2so())`

## Ví dụ: Tổng 2 số bất kỳ

```
Create function tong(@so1 int, @so2 int)
```

```
Returns int
```

```
as
```

```
Begin
```

```
Return @so1+@so2
```

```
end
```

# Thực thi

```
Declare @a int, @b int
```

```
Set @a = 4
```

```
Set @b = 6
```

```
Print 'Tong cua '+convert(char(5),@a) +' '+
```

```
convert(char(5),@b)+'='+convert(char(5),dbo.tong(@a,@b))
```

```
Select dbo.tong(@a,@b) as tong
```

## ■ Thay đổi định nghĩa một hàm

```
ALTER FUNCTION owner.function_name  
New function definition
```

## ■ Xóa một hàm

```
DROP FUNCTION owner.function_name
```

# Bài tập

- ▶ Viết hàm tính số lượng nhân viên của phòng ban (tham số đầu vào là mã phòng ban dept\_no)
- ▶ Viết hàm tính số lượng nhân viên tham gia vào dự án (tham số đầu vào là mã dự án project\_no)

# Inline Table valued Functions Syntax

```
CREATE FUNCTION [ owner_name. ] function_name  
([ { @parameter_name  
[AS] scalar_parameter_data_type [= default ] } [ ,...n ] ] )
```

## **RETURNS TABLE**

```
[WITH < function_option > [ [ , ] ...n ] ]  
[AS ]  
RETURN [ ( [ select-stmt ] ) ]
```

# Ví dụ: Cho biết số lượng nhân viên của từng phòng ban

```
create function tsnv()
```

```
returns table
```

```
as
```

```
Return (select dept_no, COUNT(*) as slnv  
        from employee  
        group by dept_no)
```

```
select * from tsnv()
```



# Bài tập

- ▶ Viết hàm tính số lượng dự án của từng nhân viên đã tham gia

# Multi-statement Table-valued Function

## ■ Cú pháp:

```
CREATE FUNCTION [ owner_name.] function_name  
([@ parameter_name parameter_data_type  
    [= default_value]} [...n])  
RETURNS @ return_variable TABLE  
( table_definition )  
[WITH option [...n]]  
AS  
    BEGIN  
        function_statements  
    RETURN  
END
```

Ví dụ: Đếm số lượng nhân viên của từng phòng ban


```
create function tsnv_dept()  
returns @thongke table (dep_no char(4),ts int)  
as begin  
    insert @thongke  
    select dept_no,COUNT(*) as slnv  
        from employee  
        group by dept_no  
  
    return  
end
```

```
select * from dbo.tsnv_dept()
```

# Bài tập

- ▶ Viết hàm cho biết số lượng nhân viên tham gia vào từng dự án

## **Bài tập (Sử dụng database QLGV đã tạo ở bài thực hành 1)**

1. Tạo hàm cho biết điểm trung bình (DTB) các môn thi của học viên (mã học viên là tham số của hàm).
  2. Tạo hàm cho biết sĩ số học viên của lớp .
  3. Tạo hàm cho biết số lượng môn học mà giáo viên đã dạy trong học kỳ (HK) của năm (NAM). Với mã giáo viên, mã học kỳ (HK) và năm (NAM) là tham số
  4. Tạo hàm cho biết danh sách giáo viên có hệ số lương cao nhất.
  5. Tạo hàm trả về kết quả là một bảng (Table) bằng hai cách: Inline Table-Valued Functions và Multistatement Table-Valued. Thông tin gồm: MAKHOA, TENKHOA, LuongTB
- 

# References

- ▶ Microsoft SQL Server T-SQL Fundamentals - ebook
- ▶ Beginners\_Guide\_to\_SQL\_Server\_2008 - ebook