



FPT POLYTECHNIC



THỰC HỌC – THỰC NGHIỆP



Conceive Design Implement Operate

QUẢN TRỊ CƠ SỞ DỮ LIỆU VỚI SQL SERVER

BÀI 4: ĐIỀU KIỆN & VÒNG LẶP

- ⦿ Điều kiện
- ⦿ Vòng lặp
- ⦿ Quản lý lỗi



Điều kiện

- ❖ Câu lệnh If....else
- ❖ Câu lệnh Case

Vòng lặp

- ❖ Câu lệnh While
- ❖ Break và Continue

Quản lý lỗi

- ❖ Try...Catch
- ❖ RAISERROR





PHẦN 1

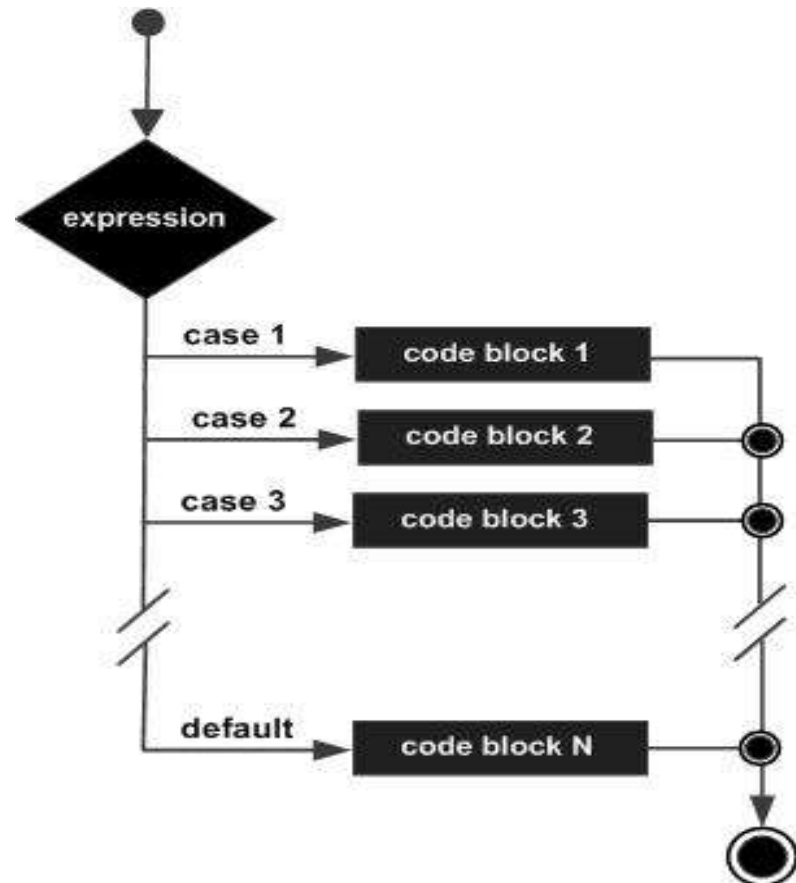
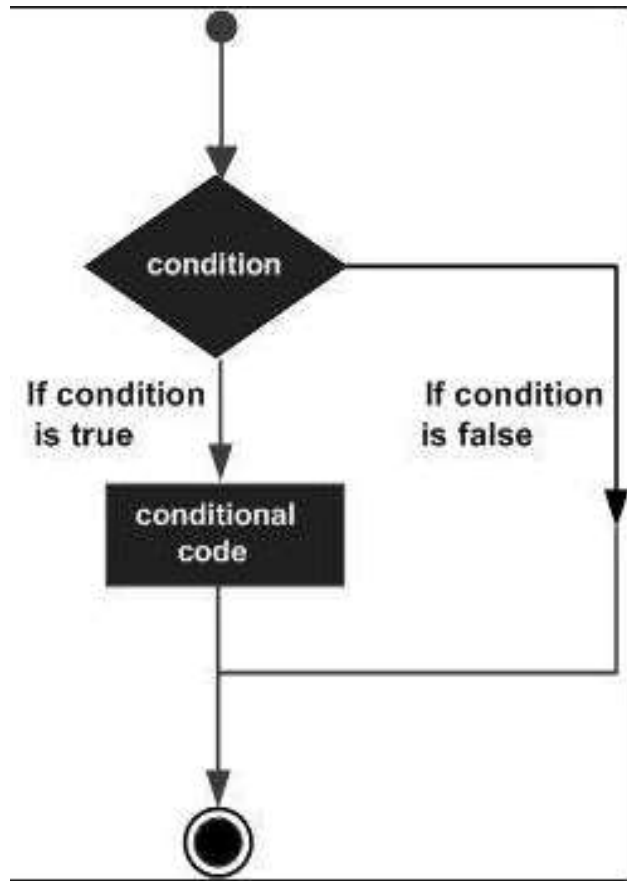
- ❑ Giả sử chúng ta viết chương trình xếp loại kết quả học tập dựa vào điểm trung bình khoá học theo tiêu chí sau:
- ❖ Nếu điểm trung bình (sau đây gọi là dtb) nhỏ hơn 5, xếp loại "Yếu"
 - ❖ Nếu dtb lớn hơn hoặc bằng 5 và nhỏ hơn 6.5, xếp loại "Trung bình"
 - ❖ Nếu dtb lớn hơn hoặc bằng 6.5 và nhỏ hơn 8, xếp loại "Khá"
- Nếu $dtb < 5$ thì "Yếu"
 - Ngược lại nếu $dtb < 6.5$ thì "Trung bình"
 - Ngược lại nếu $dtb < 8$ thì "Khá"

❑ Viết chương trình nhập vào số nguyên, hiển thị chức năng cho phép người dùng lựa chọn:

- ❖ Nhấn phím số 1: Thực hiện phép cộng
- ❖ Nhấn phím số 2: Thực hiện phép trừ
- ❖ Nhấn phím số 3: Thực hiện phép nhân
- ❖ Nhấn phím số 4: Thực hiện phép chia

- Nếu biến `pheptinh = 1` thì thực hiện phép tổng
- Nếu biến `pheptinh = 2` thì thực hiện phép trừ
- Nếu biến `pheptinh = 3` thì thực hiện phép nhân
- Nếu biến `pheptinh = 4` thì thực hiện phép chia

- ❑ Sử dụng đến câu lệnh if-else/case là là câu lệnh điều kiện được sử dụng khi cần đưa ra một quyết định nào đó



□ Câu lệnh IF - ELSE

- Cú pháp

```
IF <biểu thức điều kiện>  
    {<Câu lệnh> | BEGIN...END}  
[ELSE  
    {<Câu lệnh> | BEGIN...END}]
```

- Chú ý:

- Nếu thực thi hai hoặc nhiều câu lệnh trong mệnh đề IF hoặc ELSE. Bạn cần bao các câu lệnh này trong khối **BEGIN...END**

□ Câu lệnh IF - ELSE

```

IF 1 = 1
  BEGIN
    PRINT N'1 = 1 là đúng';
  END
ELSE
  BEGIN
    PRINT 'Sai';
  END;
  
```

Messages

1 = 1 là đúng

```

DECLARE @dbt float;
SET @dbt = 6.5;
IF @dbt < 5
  PRINT 'Yeu';
ELSE
  PRINT 'Trung binh';
GO
  
```

□ Câu lệnh IF - ELSE

```

IF(SELECT COUNT(*) FROM NHANVIEN WHERE LUONG > 30000)>0
BEGIN
    PRINT 'Danh sach nhan vien IT co luong > 30000'
    SELECT HONV,TENNV
    FROM NHANVIEN
    WHERE LUONG>3000
END
ELSE
    PRINT ' Khong co ai lam IT ma luong >30000'
  
```

	HONV	TENNV
1	Đinh	Như
2	Trần	Tâm
3	Nguyễn	Hùng
4	Nguyễn	Tùng
5	Phạm	Vinh
6	Bùi	Hành
7	Trần	Quang
8	Đinh	Tiên

```

DECLARE @dbt float;
SET @dbt = 6.5;
IF @dbt < 5
    PRINT 'Yeu';
ELSE
    BEGIN
        IF @dbt < 6.5
            PRINT 'Trung bình';
        ELSE
            PRINT 'Kha';
    END
GO
  
```

❑ Lệnh If Exists

```
IF EXISTS (Câu_lệnh_SELECT)
    Câu_lệnh1 | Khối_lệnh1
[ELSE
    Câu_lệnh2 | Khối_lệnh2]
```

```
IF EXISTS(SELECT * FROM NHANVIEN WHERE LUONG > 300000)
BEGIN
    PRINT 'Danh sach nhan vien IT co luong > 30000'
    SELECT HONV,TENNV
        FROM NHANVIEN
        WHERE LUONG>3000
END
ELSE
    PRINT ' Khong co ai lam IT ma luong >30000'
```

❑ Sử dụng IIF Function

Syntax :

IIF (expression, expr_true, expr_false)



```
IIF(expression ,expr_true, expr_false);
```

```
SELECT IIF( LUONG>30000 , 'Truong phong', 'NhanVien')
as ChucVu, TENNV, LUONG
FROM NHANVIEN
```

	ChucVu	TENNV	LUONG
1	Truong phong	Như	43000
2	NhanVien	Tâm	25000
3	Truong phong	Hùng	38000
4	Truong phong	Tùng	40000
5	Truong phong	Vinh	55000
6	NhanVien	Hành	25000
7	NhanVien	Quang	25000
8	NhanVien	Tiên	30000

❑ Hàm CASE trong SQL Server

- ❖ Hàm CASE kiểm định giá trị dựa trên danh sách điều kiện đưa ra, sau đó trả về một hoặc nhiều kết quả.
- ❖ CASE rất đa dạng, linh hoạt và rất hữu ích, ứng dụng trong nhiều trường hợp.
- ❖ CASE có 2 định dạng:
 - 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

❑ Hàm CASE trong SQL Server

❖ Simple CASE

Cú pháp hàm CASE đơn giản

```

CASE <biểu thức>
  WHEN <biểu thức 1> THEN <biểu thức kết quả 1>
  [WHEN <biểu thức 2> THEN <biểu thức kết quả 2>]...
  [ELSE <biểu thức kết quả mệnh đề else>]
END
  
```

```
-- Thêm tiền tố Mr hoặc Ms tùy vào phái là nam hay nữ
```

```

Select TenNV = case PHAI
  when 'nam' then 'Mr. '+[TENNV]
  when N'Nữ' then 'Ms. '+[TENNV]
end
from NHANVIEN
  
```

	TenNV
1	Ms. Như
2	Mr. Tâm
3	Mr. Hùng
4	Mr. Tùng
5	Ms. Vinh
6	Mr. Hành
7	Mr. Quang
8	Mr. Tiên

```

-- Thêm tiền tố Mr hoặc Ms tùy vào phái là nam hay nữ
Select TenNV = case PHAI
  when 'nam' then 'Mr. '+[TENNV]
  when N'Nữ' then 'Ms. '+[TENNV]
  else 'FreeSex. ' + [TENNV]--Sử dụng với ELSE
end
from NHANVIEN
  
```

❑ Hàm CASE trong SQL Server

❖ Searched CASE:

CASE

```
WHEN biểu_thức_điều-kiện_1 THEN biểu_thức_kết_quả_1
WHEN biểu_thức_điều-kiện_2 THEN biểu_thức_kết_quả_2
...
WHEN biểu_thức_điều-kiện_n THEN biểu_thức_kết_quả_n
ELSE biểu_thức_kết_quả
```

END

--Tạo thêm cột thuế dựa vào mức lương

```
Select TENNV, LUONG, Thuế = case
  When LUONG between 0 and 25000 then LUONG*0.1
  When LUONG between 25000 and 30000 then LUONG*0.12
  When LUONG between 30000 and 40000 then LUONG *0.15
  When LUONG between 40000 and 50000 then LUONG *0.2
  else LUONG*0.25 end
from NHANVIEN
```

	TENNV	LUONG	Thuế
1	Như	43000	8600
2	The	30000	3600
3	Tâm	25000	2500
4	Hùng	38000	5700
5	Tùng	40000	6000
6	Vinh	55000	13750
7	Hành	25000	2500
8	Quang	25000	2500
9	Tiên	30000	3600

❑ So sánh Simple CASE và Searched CASE

--Simple CASE

```
Select TenNV = case PHAI
when 'nam' then 'Mr. ' + [TENNV]
when N'Nữ' then 'Ms. ' + [TENNV]
else 'FreeSex. ' + [TENNV] --Sử dụng với ELSE
end
from NHANVIEN
```

--Searched CASE

```
Select TenNV = case
when Phai like 'nam' then 'Mr. ' + [TENNV]
when PHAI like N'Nữ' then 'Ms. ' + [TENNV]
else 'FreeSex. ' + [TENNV] --Sử dụng với ELSE
end
from NHANVIEN
```

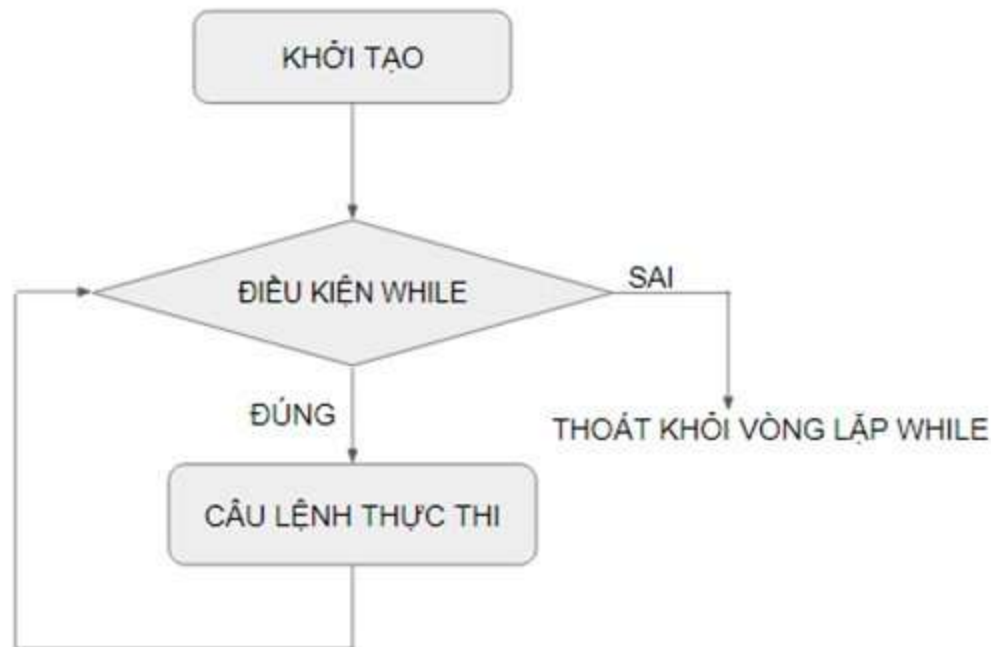



- ❖ Viết câu truy vấn đếm số lượng nhân viên trong từng phòng ban, nếu số lượng nhân viên nhỏ hơn 3 → hiển thị “Thiếu nhân viên”, ngược lại <5 hiển thị “Đủ Nhân Viên”, ngược lại hiển thị “Đông nhân viên”
- ❖ Viết câu truy vấn hiển thị TenNV và thêm cột thuế dựa vào mức lương: trong khoảng 0 and 25000 thì Thuế= LUONG*0.1, trong khoảng 25000 and 30000 thì LUONG*0.12, trong khoảng 30000 and 40000 thì LUONG *0.15, trong khoảng 40000 and 50000 thì LUONG *0.2, còn lại LUONG*0.25



PHẦN 2

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



□ Cú pháp:

WHILE <biểu thức điều kiện>
{ <câu lệnh> | **BEGIN...END** }
[**BREAK**]
[**CONTINUE**]

```
DECLARE @dem INT = 0;  
WHILE @dem < 5  
BEGIN  
    PRINT 'Quan trọng là phương pháp học';  
    SET @dem = @dem + 1;  
END;  
PRINT 'Học lập trình thì ra cũng dễ';  
GO
```

Messages

Quan trọng là phương pháp học
Quan trọng là phương pháp học
Quan trọng là phương pháp học
Quan trọng là phương pháp học
Quan trọng là phương pháp học
Học lập trình thì ra cũng dễ

❑ Lệnh Break (Ngắt điều khiển)

- ❖ Dùng để thoát khỏi vòng lặp
- ❖ Không có tham số và đối số nào nằm trong câu lệnh BREAK
- ❖ Nếu trong đoạn code có WHILE LOOP lồng nhau, BREAK sẽ chấm dứt vòng lặp WHILE gần nhất

```
DECLARE @Number INT = 1 ;  
DECLARE @Total INT = 0 ;  
WHILE @Number < = 10  
BEGIN  
    IF @NUMBER = 5  
        BREAK;  
    ELSE  
        SET @Total = @Total + @Number;  
        SET @Number = @Number + 1 ;  
END  
PRINT @Total;
```



❑ Lệnh Continue:

- ❖ Thực hiện bước lặp tiếp theo, bỏ qua các lệnh trong bước lặp hiện tại.
- ❖ Không có tham số và đối số nào nằm trong câu lệnh CONTINUE

```
--Không sử dụng Continue
DECLARE @Number INT = 1 ;
DECLARE @Total INT = 0 ;
WHILE @Number < = 10
BEGIN
  IF @NUMBER = 5
  BREAK;
  ELSE
  SET @Total = @Total + @Number;
  SET @Number = @Number + 1 ;
  SET @Number = @Number + 1 ;
END;
PRINT @Total;
GO
```

Messages

4

```
--Có sử dụng Continue
DECLARE @Number INT = 1 ;
DECLARE @Total INT = 0 ;
WHILE @Number < = 10
BEGIN
  IF @NUMBER = 5
  BREAK;
  ELSE
  SET @Total = @Total + @Number;
  SET @Number = @Number + 1 ;
  Continue
  SET @Number = @Number + 1 ;
END;
PRINT @Total;
GO
```

Messages

10



- ❖ Viết chương trình tính tổng các số chẵn từ 1 tới 10.
- ❖ Viết chương trình tính tổng các số chẵn từ 1 tới 10 nhưng bỏ số 4.

❑ Xử lý lỗi TRY...CATCH

- ❖ Thực hiện các lệnh trong khối TRY, nếu gặp lỗi sẽ chuyển qua xử lý bằng các lệnh trong khối CATCH

BEGIN TRY

{<câu lệnh SQL> | <Khối câu lệnh>}

END TRY

BEGIN CATCH

{<Câu lệnh SQL> | <Khối câu lệnh>}

END CATCH

- ❖ Các điểm cần lưu ý
 - TRY và CATCH phải cùng lô xử lý
 - Sau khối TRY phải là khối CATCH
 - Có thể lồng nhiều cấp

❑ Xử lý lỗi TRY...CATCH

❖ Một số hàm ERROR thường dùng

- `ERROR_NUMBER()` : Trả về mã số của lỗi
- `ERROR_MESSAGE()` Trả về chuỗi lỗi
- `ERROR_SEVERITY()` returns the error severity.
- `ERROR_STATE()` returns the error state number.
- `ERROR_LINE()` : Trả về dòng gây ra lỗi
- `ERROR_PROCEDURE()` Trả về tên thủ tục/ trigger gây ra lỗi

```

BEGIN TRY
    SELECT 1 + 'SQL';
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() as ErrorNumber,
        ERROR_MESSAGE() as ErrorMessage;
END CATCH
    
```

Results		Messages
(No column name)		
ErrorNumber	ErrorMessage	
1	245	Conversion failed when converting the varchar va...

❑ Xử lý lỗi TRY...CATCH

```
-- Bắt và xử lý lỗi chèn dữ liệu vào bảng PhongBan
BEGIN TRY
    INSERT PHONGBAN
    VALUES (799, 'ZK-799', '2008-07-01', '0197-05-22')
    -- Nếu lệnh chèn thực thi thành công in ra dòng bên dưới
    PRINT 'SUCCESS: Record was inserted.'
END TRY

-- Nếu có lỗi xảy ra khi chèn dữ liệu in ra dòng thông báo lỗi
BEGIN CATCH
    PRINT 'FAILURE: Record was not inserted.'
    PRINT 'Error ' + CONVERT(varchar, ERROR_NUMBER(), 1)
      + ': ' + ERROR_MESSAGE()
END CATCH
```

 Messages

(0 rows affected)

FAILURE: Record was not inserted.

Error 245: Conversion failed when converting the varchar value 'ZK-799' to data type int.

❑ Thủ tục RAISERROR

❖ Trả thông báo lỗi về cho ứng dụng


```
Raiserror(tbao_loi, muc_do, trang_thai [, cac_tham_so] )
```

Trong đó:

- tbao_loi :
 - mã thông báo lỗi do người dùng định nghĩa trước bằng sp_addmessage và được lưu trong sys.messages. Giá trị phải lớn hơn 50000.
 - chuỗi thông báo lỗi bất kỳ.
- muc_do:
 - Số có giá trị từ 0
 - 25 thể hiện mức độ nghiêm trọng của lỗi.
- trang_thai: Số từ 1-127 để xác định vị trí lỗi khi sử dụng cùng một tbao_loi tại nhiều điểm khác nhau
- cac_tham_so : Hỗ trợ cho các tbao_loi cần tham số


❑ Thủ tục RAISERROR

```
--Không dùng RAISERROR
BEGIN TRY
    DECLARE @result INT
    --Generate divide-by-zero error
    SET @result = 55/0
END TRY
BEGIN CATCH
    DECLARE
        @ErMessage NVARCHAR(2048),
        @ErSeverity INT,
        @ErState INT
    SELECT
        @ErMessage = ERROR_MESSAGE(),
        @ErSeverity = ERROR_SEVERITY(),
        @ErState = ERROR_STATE()
END CATCH
```

 Messages

Commands completed successfully.

```
--Sử dụng RAISERROR
BEGIN TRY
    DECLARE @result INT
    --Generate divide-by-zero error
    SET @result = 55/0
END TRY
BEGIN CATCH
    DECLARE
        @ErMessage NVARCHAR(2048),
        @ErSeverity INT,
        @ErState INT
    SELECT
        @ErMessage = ERROR_MESSAGE(),
        @ErSeverity = ERROR_SEVERITY(),
        @ErState = ERROR_STATE()
    RAISERROR (@ErMessage,
        @ErSeverity,
        @ErState )
END CATCH
```

 Messages

Msg 50000, Level 16, State 1, Line 46
Divide by zero error encountered.



❖ Demo các ví dụ trong phần Try..Catch và RAISERROR

☑ Điều kiện

- ❖ Câu lệnh If....else
- ❖ Câu lệnh Case

☑ Vòng lặp

- ❖ Câu lệnh While
- ❖ Break và Continue

☑ Quản lý lỗi

- ❖ Try...Catch
- ❖ RAISERROR



thank
you!