



ĐẠI HỌC ĐÀ NẴNG – ĐẠI HỌC KINH TẾ
KHOA THỐNG KÊ TIN HỌC

LẬP TRÌNH SQL

Giảng viên: Cao Thị Nhâm
nhamct@due.edu.vn



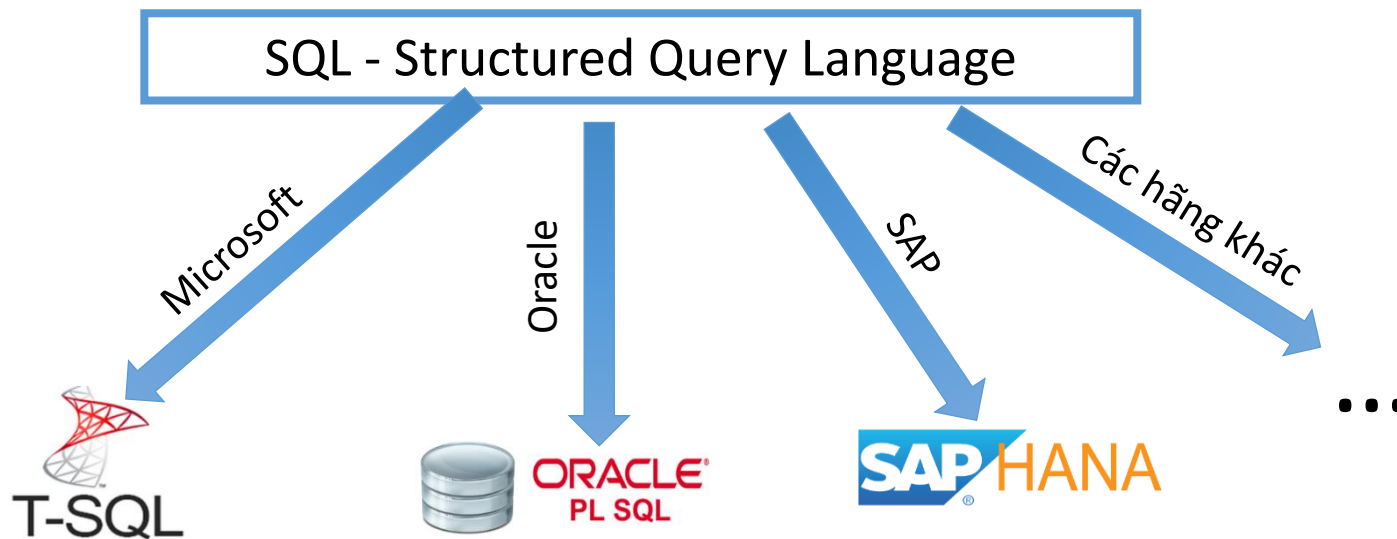


NỘI DUNG CHÍNH

- Giới thiệu về lập trình SQL
- Một số quy tắc lập trình
- Biến, toán tử và các lệnh điều khiển
- Thủ tục (Stored-procedures)
- Hàm (Functions)
- Trigger



MÔI TRƯỜNG LẬP TRÌNH SQL



(Tham khảo thêm tại: <https://en.wikipedia.org/wiki/SQL>)



QUY TẮC TRONG LẬP TRÌNH T-SQL

- Không phân biệt HOA thường khi gõ lệnh
- Khai báo ngữ cảnh sử dụng trước khi lập trình



Biến

- Các loại biến
 - Vô hướng
 - Bảng
- Khai báo biến

```
DECLARE @<variable name> <variable type>[= <value>][,  
        @<variable name> <variable type>[= <value>][,  
        @<variable name> <variable type>[= <value>]]]
```

```
DECLARE @<variable name> TABLE (  
    <column spec> [, . . .]  
)
```



Biến...

- Quy tắc đặt tên biến
 - Luôn bắt đầu bởi @ (biến toàn cục bắt đầu bởi @@)
 - Tên không quá 50 ký tự
 - Nên đặt theo kiểu camelCase
 - Không sử dụng các ký tự đặc biệt



Biến...

- Gán giá trị cho biến vô hướng
 - Gán khi khai báo (khởi tạo giá trị)
 - Dùng lệnh SET

```
SET @variable_name = value/expression
```

- Dùng lệnh SELECT

```
SELECT @var1 = col1/exp1, @var2 = col2/exp2,...  
FROM ...
```



Biến...

- So sánh SET và SELECT

SET	SELECT
1 lệnh SET, chỉ gán giá trị cho 1 biến	1 lệnh SELECT, gán cho nhiều biến
	Gán giá trị ngay trong query

```
declare @ac_type int, @ac_no nvarchar(50)

select @ac_type = ac_type, @ac_no = ac_no
from account
where ac_no = '1000000001'
```

```
declare @ac_type int
set @ac_type = (select ac_type from account where ac_no = '1000000001')
```




Biến...

- Gán giá trị cho biến TABLE

- Sử dụng các lệnh như với bảng thông thường

```
INSERT INTO @tên_biến VALUES(gt1, gt2, ...)
```

```
INSERT INTO @tên_biến SELECT ...FROM...
```

```
UPDATE @tên_biến SET...WHERE...
```

```
DELETE @tên_biến WHERE...
```

```
SELECT ...FROM @tên_biến...
```

```
DECLARE @customer TABLE (cust_id int,  
                           cust_name nvarchar(100),  
                           primary key(cust_id)  
                           )  
insert into @customer values(01, N'Cao Thị Nhâm')  
select * from @customer
```

	Results	Messages
	cust_id	cust_name
1	1	Cao Thị Nhâm



Một số biến tiện ích

Tên biến	Mục đích
@@ERROR	Mã lỗi của câu lệnh thực thi gần nhất
@@ROWCOUNT	Số lượng bản ghi chịu tác động của câu lệnh gần nhất
@@SERVERNAME	Tên server



Phép toán

- Giống các phép toán đã học



Cấu trúc điều khiển – Rẽ nhánh

- IF...ELSE

```
IF <Boolean Expression>  
    <SQL statement> | BEGIN <code series> END  
[ELSE  
    <SQL statement> | BEGIN <code series> END]
```

- CASE (nên dùng trong trường hợp có nhiều nhánh)


```
CASE <input expression>  
WHEN <when expression> THEN <result expression>  
[...n]  
[ELSE <result expression>]  
END
```

```
CASE  
WHEN <Boolean expression> THEN <result expression>  
[...n]  
[ELSE <result expression>]  
END
```

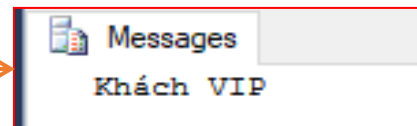


Cấu trúc điều khiển – Rẽ nhánh...

```
declare @ac_type int, @ret nvarchar(50)
select @ac_type = ac_type from account where ac_no = '1000000001'
if @ac_type = 1
begin
    print 'Khách VIP'
end
else if @ac_type = 0
begin
    print 'Khách thường'
end
else
begin
    print 'Không xác định'
end
```



	Ac_no	ac_balance	ac_type	cust_id
1	1000000001	88118000	1	000001





Cấu trúc điều khiển – Rẽ nhánh...

```
declare @ac_type int, @ret nvarchar(50)

select @ac_type = ac_type from account where ac_no = '1000000001'
set @ret = case when @ac_type = 1 then 'Khách VIP'
               when @ac_type = 0 then 'Khách thường'
               else 'Không xác định'
            end
print @ret
```

```
declare @ac_type int, @ret nvarchar(50)

select @ac_type = ac_type from account where ac_no = '1000000001'
set @ret = case @ac_type when 1 then 'Khách VIP'
                     when 0 then 'Khách thường'
                     else 'Không xác định'
            end
print @ret
```



Cấu trúc điều khiển – Lặp

- WHILE

```
WHILE <Boolean expression>
    <sql statement> |
[BEGIN
    <statement block>
    [BREAK]
    <sql statement> | <statement block>
    [CONTINUE]
END]
```

- Dừng:

- CONTINUE: để bỏ qua vòng lặp hiện tại
- BREAK: để ngắt lặp



Cấu trúc điều khiển – Lặp...

```
declare @count int, @name nvarchar(100)
set @count = 0
while @count < 10
begin
    select @name = cust_name from customer
    where cust_id = '00000' + cast(@count as varchar)

    print @name

    set @count = @count + 1
end
```

	Cust_id	Cust_name
1	000001	Hà Công Lục
2	000002	Trần Đức Quý
3	000003	Lê Quang Phong
4	000004	Trần Văn Thiện Thanh
5	000005	Nguyễn Đức Duy
6	000006	Nguyễn Quang Công Minh
7	000007	Trần Trương Thiện Nguyên
8	000008	Nguyễn Lê Minh Quân
9	000009	Nguyễn Văn Linh
10	000010	Đặng Nhật Phong



Messages
Hà Công Lục
Trần Đức Quý
Lê Quang Phong
Trần Văn Thiện Thanh
Nguyễn Đức Duy
Nguyễn Quang Công Minh
Trần Trương Thiện Nguyên
Nguyễn Lê Minh Quân
Nguyễn Văn Linh



Bắt lỗi (try...catch)

- Cú pháp

```
BEGIN TRY
    { <sql statement(s)> }
END TRY
BEGIN CATCH
    { <sql statement(s)> }
END CATCH [ ; ]
```

- Mã lỗi

Mã lỗi	Ý nghĩa
1 - 10	Informational only.
11 - 19	Relatively severe errors.
20 -25	Very severe

- Hàm bắt lỗi

- `ERROR_NUMBER()`, `ERROR_SEVERITY()`, `ERROR_STATE()`,
`ERROR_PROCEDURE()`, `ERROR_LINE()`, `ERROR_MESSAGE()`



Bắt lỗi...

```
declare @count int, @name nvarchar(100)
set @count = 0
while @count < 10
begin try
    select @name = cust_name from customer
    where cust_id = '00000' + cast(@count as varchar)

    print @name

    set @count = @count + 1
end try
begin catch
    print ERROR_MESSAGE();
end catch
```

Thủ tục (Stored-procedures)

- Thủ tục là gì?
- Định nghĩa, chỉnh sửa, xóa thủ tục
- Gọi thủ tục





Khái niệm thủ tục

- Thủ tục là tập hợp các lệnh SQL được lưu trữ trong CSDL để thực hiện một công việc nào đó.
- Đặc điểm
 - Mỗi thủ tục có một tên duy nhất
 - Có các tham số đầu vào và đầu ra
 - Có thể sử dụng lại nhiều lần
- Ưu điểm
 - Động
 - Nhanh hơn Transact-SQL
 - Giảm thiểu bandwidth
 - Bảo mật



Định nghĩa thủ tục

- Một thủ tục gồm 3 phần:
 - Tên
 - Tham số
 - Thân (nội dung của thủ tục)

```
CREATE PROCEDURE/PROC procedure_name [@param data_type [output],...]  
AS  
Begin  
    [Declare variables for processing]  
    {Transact-SQL statements}  
End
```



Định nghĩa thủ tục...

```
CREATE PROC spGetName (@cust_id varchar(6), @cust_name nvarchar(100) output)
as
begin
    select @cust_name = cust_name from customer where cust_id = @cust_id
end
```

```
CREATE PROC spGetAc (@cust_id varchar(6), @ret nvarchar(50) output)
as
begin
    declare @temp int = 0
    select @temp = count(*) from account where cust_id = @cust_id
    if @temp = 0
        set @ret = N'Không có tài khoản'
    else if @temp = 1
        set @ret = N'Có 1 tài khoản'
    else
        set @ret = N'Có nhiều hơn 1 tài khoản'
end
```



Chỉnh sửa thủ tục

- Sửa nội dung thủ tục → **ALTER PROC**

```
ALTER PROCEDURE/PROC procedure_name [@param data_type input/output]  
AS  
Begin  
    [Declare variables for processing]  
    {Transact-SQL statements}  
End
```

- Xóa thủ tục
`DROP PROC|PROCEDURE <spc name> [;]`



Gọi thủ tục

- Cú pháp
 - Không có tham số

```
EXECUTE/EXEC procedure_name
```

- Có tham số

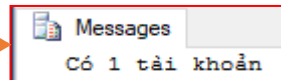
```
EXEC procedure_name Para1_value, Para2_value, ...
```




Gọi thủ tục

```
CREATE PROC spGetAc (@cust_id varchar(6), @ret nvarchar(50) output)
as
begin
    declare @temp int = 0
    select @temp = count(*) from account where cust_id = @cust_id
    if @temp = 0
        set @ret = N'Không có tài khoản'
    else if @temp = 1
        set @ret = N'Có 1 tài khoản'
    else
        set @ret = N'Có nhiều hơn 1 tài khoản'
end
```

```
DECLARE @a nvarchar(100)
exec spGetAc '000002', @a output
print @a
```



Messages
Có 1 tài khoản

Hàm (User defined functions)

- Hàm là gì?
- Định nghĩa, chỉnh sửa hàm
- Gọi hàm





Định nghĩa hàm

- Hàm là tập hợp các lệnh SQL được biên dịch, lưu trữ trong cơ sở dữ liệu và được sử dụng như một đơn vị độc lập
- So sánh thủ tục và hàm

THỦ TỤC	HÀM
Là tập hợp các lệnh SQL có chức năng thực hiện một công việc nào đó	
Giá trị trả về thông qua tham số	Trả về một giá trị cho tên hàm
Trả về nhiều giá trị	Chỉ trả về một giá trị (vô hướng/có hướng)
Có thể thực hiện các lệnh DML	Chỉ có thể dùng lệnh SELECT



Định nghĩa hàm

- Hàm vô hướng
 - Giá trị trả về vô hướng

```
CREATE FUNCTION function_Name ([@param data_type,...] )  
RETURNS data_type  
AS  
Begin  
    [Declare variables for processing]  
    {Transact-SQL statements}  
    RETURN ...  
End
```



Ví dụ hàm vô hướng

```
CREATE FUNCTION fGetComment (@cust_id varchar(10))
RETURNS nvarchar(100)
AS
BEGIN
    DECLARE @cust_ad nvarchar(100),
            @comment nvarchar(100)
    set @cust_ad = (select cust_ad from customer where cust_id = @cust_id)

    if @cust_ad like N'%Đà Nẵng'
        set @comment = N'Khách sống ở Đà Nẵng'
    else
        set @comment = N'Khách không sống ở Đà Nẵng'

    return @comment
END
```

```
select dbo.fGetComment('000001');
```

	Cust_id	Cust_name	Cust_phone	Cust_ad	Br_id
1	000001	Hà Công Lực	01283388103	NGUYỄN TIẾN DUÂN - THÔN 3 - XÃ DHÉYANG - EAHLEO - ĐẮKLẮK	VT009

(No column name)
1
Khách không sống ở Đà Nẵng



Định nghĩa hàm...

- Hàm trả về bảng

- Inline

```
CREATE FUNCTION function_Name ([@param data_type,...])  
RETURNS table  
AS  
RETURN query_statement
```

- Khai báo tường minh


```
CREATE FUNCTION function_Name ([@param data_type,...])  
RETURNS @var_name table ( col_name_1 data_type,  
                           col_name_2 data_type,...)  
  
AS  
BEGIN  
    [Declare variables for processing]  
    {Transact-SQL statements}  
    RETURN ...  
  
END
```



Ví dụ hàm có hướng

```
CREATE FUNCTION fGetAcList (@cust_id varchar(10))
RETURNS table
as
RETURN  select ac_no, ac_balance, case ac_type when 1 then 'VIP'
                                             when 0 then N'Thường'
                                             else N'Không xác định' end Ac_Type
        from account
        where cust_id = @cust_id
```

```
select * from dbo.fGetAcList('000001')
```



Results		Messages	
	ac_no	ac_balance	Ac_Type
1	1000000001	88118000	VIP
2	1000000036	381588000	VIP



Ví dụ hàm có hướng

```
CREATE FUNCTION fGetAcList1 (@cust_id varchar(10))
RETURNS @account table (ac_no varchar(10),
                        ac_balance numeric(12,0),
                        ac_type nvarchar(50))
as
BEGIN
    INSERT INTO @ACCOUNT
    select ac_no, ac_balance, case ac_type when 1 then 'VIP'
                                     when 0 then N'Thường'
                                     else N'Không xác định' end Ac_Type
    from account
    where cust_id = @cust_id

    RETURN
END
```

```
select * from dbo.fGetAcList1('000001')
```

	ac_no	ac_balance	ac_type
1	1000000001	88118000	VIP
2	1000000036	381588000	VIP

TRIGGER

- Trigger là gì?
- Đặc điểm
- Định nghĩa, chỉnh sửa trigger
- Sử dụng trigger





Đặt vấn đề

Account(ac_no, ac_type, ac_balance, cust_id)

Transactions(t_id, t_type, t_amount, t_date, t_time, ac_no)

Account		
ac_no	Cust_id	Ac_balance
16111	000001	1.000.000
16222	000002	3.000.000
16333	000003	2.000.000

transactions					
T_id	Ac_no	T_date	T_time	T_type	T_amount
1011301	16111	1:30	01/01/2013	1	1.000.000
1041301	16111	2:12	01/04/2013	0	2.000.000
1041302	16222	3:20	01/04/2013	0	2.000.000
1041303	16333	00:11	01/04/2013	1	3.000.000
1041305	16222	5:20	01/04/2013	0	4.000.000

R1: “Số tiền tối thiểu mỗi lần rút là 50.000”

R2: “Số tiền tối đa mỗi lần rút: nhỏ hơn số tiền trong tài khoản”

R3: “Sau khi rút tiền, cập nhật lại số dư tài khoản:

$SoDuTK = SoDuTK - SoTienGD$ ”

Trigger



Khái niệm

- Trigger là một loại stored procedure được dùng để thực hiện một số xử lý một cách **tự động** khi có một sự kiện nào đó xảy ra đối với CSDL
- Một số xử lý cần tới trigger
 - Kiểm tra ràng buộc dữ liệu (phức tạp, liên quan tới nhiều bảng,...)
 - Tính toán nghiệp vụ
 - Lưu vết hoạt động



Đặc điểm

- Được thực hiện tự động khi có sự thay đổi nào đó liên quan tới CSDL
- Không thể gọi tường minh như một stored procedure thông thường
- Không trả về dữ liệu cho người dùng



Cú pháp tạo trigger

```
CREATE TRIGGER Tên_Trigger  
ON Tên_Table  
AFTER (FOR) | INSTEAD OF INSERT, DELETE, UPDATE  
AS  
BEGIN  
    Các_lệnh_của_Trigger  
END
```



Cú pháp tạo trigger

- Sự kiện kích hoạt trigger

- INSERT
- UPDATE
- DELETE

- Loại trigger

- AFTER (FOR)
- INSTEAD OF



AFTER trigger

- Được gọi sau khi đã thực hiện thành công thao tác insert/update/delete
- Có thể quay lui thao tác thực hiện bằng lệnh **ROLLBACK TRANSACTION**

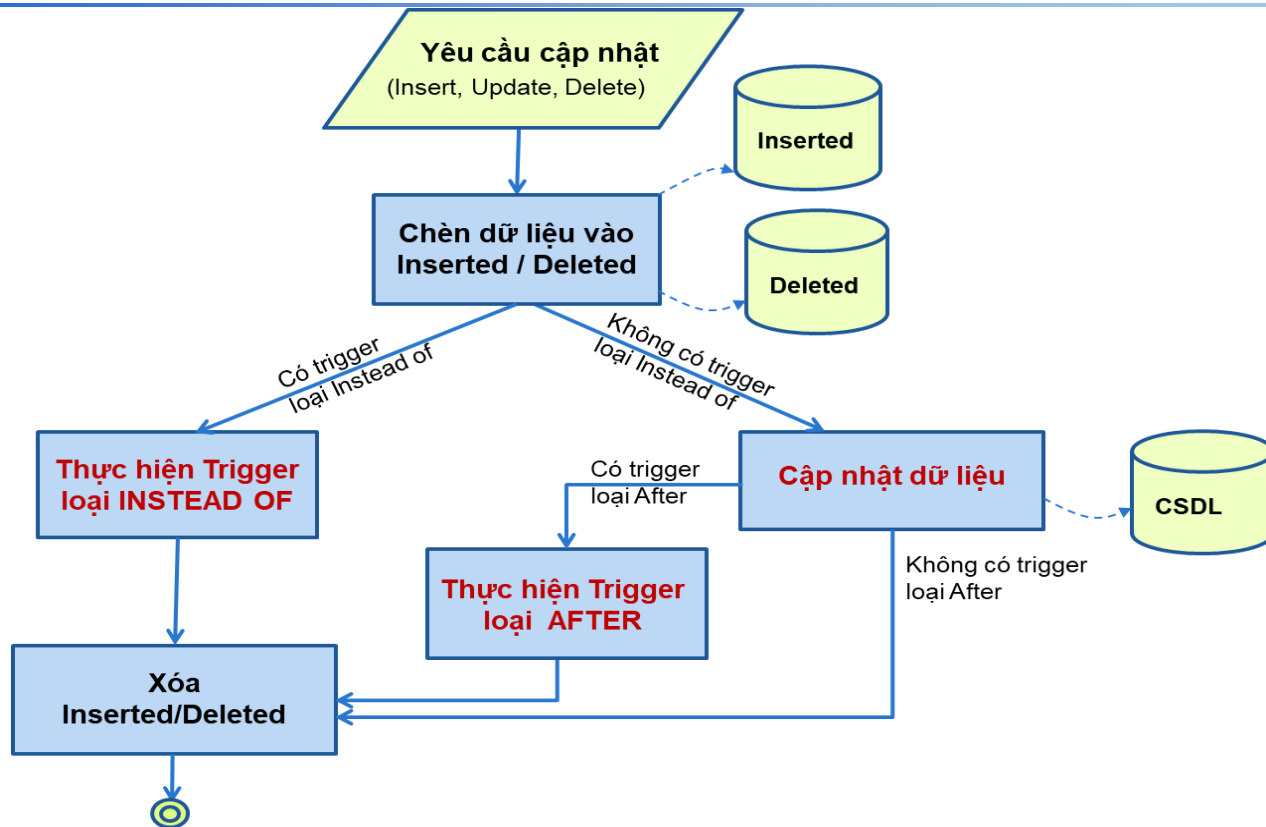


INSTEAD OF trigger

- Trigger được gọi thực hiện thay cho thao tác insert/update/delete
- Thường được dùng để xử lý cập nhật trên view



Hoạt động của trigger





INSERTED & DELETED – “magic tables”

- Khi 1 trigger được kích hoạt, SQL Server tự động tạo ra 2 bảng tạm **INSERTED** và **DELETED**
 - **INSERTED**: chứa bản copy của những dữ liệu được insert trong trigger
 - **DELETED**: chứa bản copy của những dữ liệu được delete trong trigger

Sự kiện kích hoạt	INSERTED	DELETED
INSERT	V	
DELETE		V
UPDATE	V	V



INSERTED & DELETED – “*magic tables*”...

- Đặc điểm
 - Được SQL Server tạo vào xóa tự động
 - Có cấu trúc như bảng mà trigger đang làm việc
 - Chỉ tồn tại trong thời gian xử lý trigger
 - Cục bộ cho mỗi trigger



Chỉnh sửa trigger

- Dùng lệnh ALTER thay cho CREATE



Bật tắt hiệu lực của trigger

- Bật/tắt 1 trigger

```
ALTER TABLE tên_bảng  
DISABLE/ENABLE TRIGGER tên_trigger
```

- Bật/tắt tất cả trigger

```
ALTER TABLE tên_bảng  
DISABLE/ENABLE TRIGGER ALL
```



Một số chú ý

- Lệnh tạo trigger phải là lệnh đầu tiên trong một lô (query batch)
- Trên một bảng có thể định nghĩa nhiều trigger after(for) cho mỗi thao tác, nhưng chỉ có thể định nghĩa một trigger instead of cho mỗi thao tác
- Không thể định nghĩa trigger instead of update/ delete trên bảng có cài đặt khoá ngoại dạng update cascade/ delete cascade



Ví dụ trigger

```
CREATE TRIGGER tCkTime
on transactions
for insert
as
    declare @t_date date, @t_time time
begin
    select @t_date = t_date from inserted
    if @t_date > getdate()
    begin
        print N'Ngày giao dịch không hợp lệ'
        rollback
    end
end
```

```
insert into transactions values('1000000203',0,999999,'2020-09-16','00:00','1000000041')
```

Messages

Ngày giao dịch không hợp lệ
Msg 3609, Level 16, State 1, Line 16
The transaction ended in the trigger. The batch has been aborted.

