

# Quản Trị Dữ Liệu Với Microsoft SQL Server

## Chương: 15

### Quản Lý Lỗi (Error Handling)



- Giải thích các loại lỗi khác nhau
- Giải thích thực thi và quản lý lỗi
- Mô tả khối TRY-CATCH
- Giải thích các thủ tục(procedure) để hiển thị thông tin lỗi
- Mô tả các câu lệnh @@ERROR và RAISERROR
- Giải thích sử dụng ERROR\_STATE, ERROR\_SEVERITY, và ERROR\_PROCEDURE
- Giải thích sử dụng ERROR\_NUMBER, ERROR\_MESSAGE, và ERROR\_LINE
- Mô tả lệnh THROW



# Giới thiệu

- Việc quản lý lỗi trong SQL Server trở lên dễ dàng thông qua một số kỹ thuật khác nhau như:
  - SQL Server cung cấp câu lệnh `TRY...CATCH` giúp quản lý các lỗi hiệu quả ở phía sau (back end).
  - SQL Server cũng cung cấp một số các hàm hệ thống để in ra các thông tin có liên quan đến lỗi, để giúp sửa lỗi(fix) một cách dễ dàng.

# Types of Errors 1-2

Một lập trình viên Transact-SQL cần phải xác định loại lỗi, sau đó sẽ xác định cách quản lý và khắc phục nó.

Dưới đây là một số loại lỗi:

➤ **Các lỗi cú pháp:**

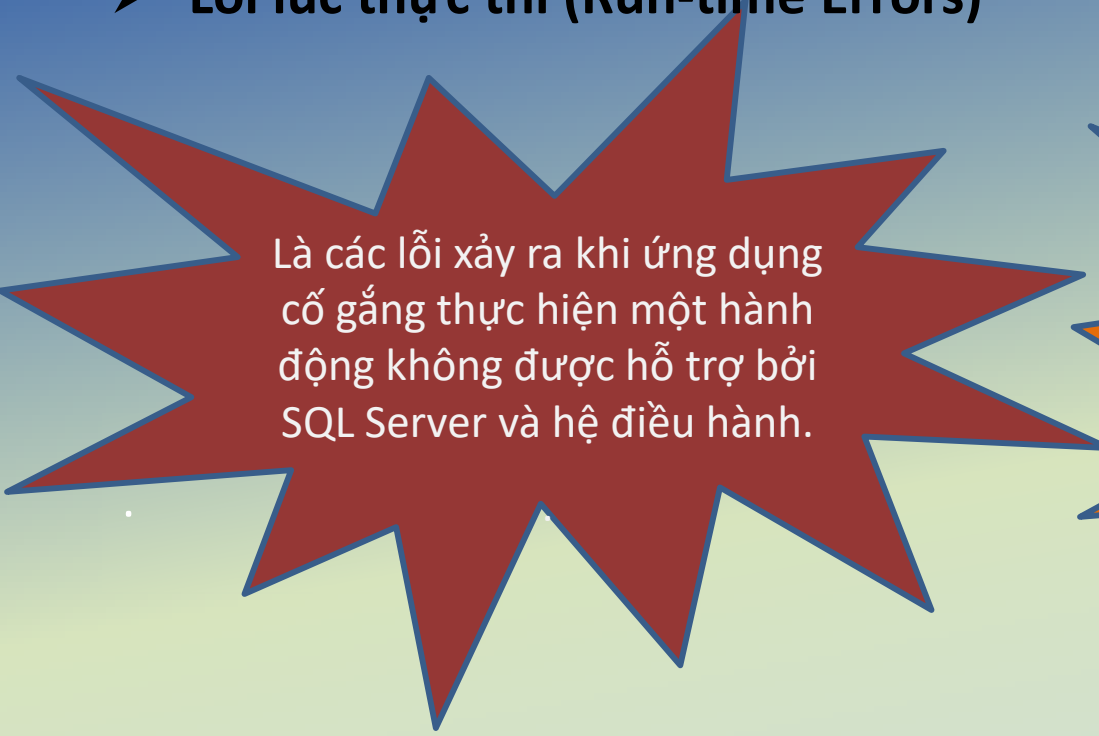
Là các lỗi xảy ra khi code không thể phân tích được bởi SQL Server.

Dễ dàng được xác định khi trình soạn thảo chỉ ra chúng ra, do đó có thể dễ dàng sửa lỗi (fixed).

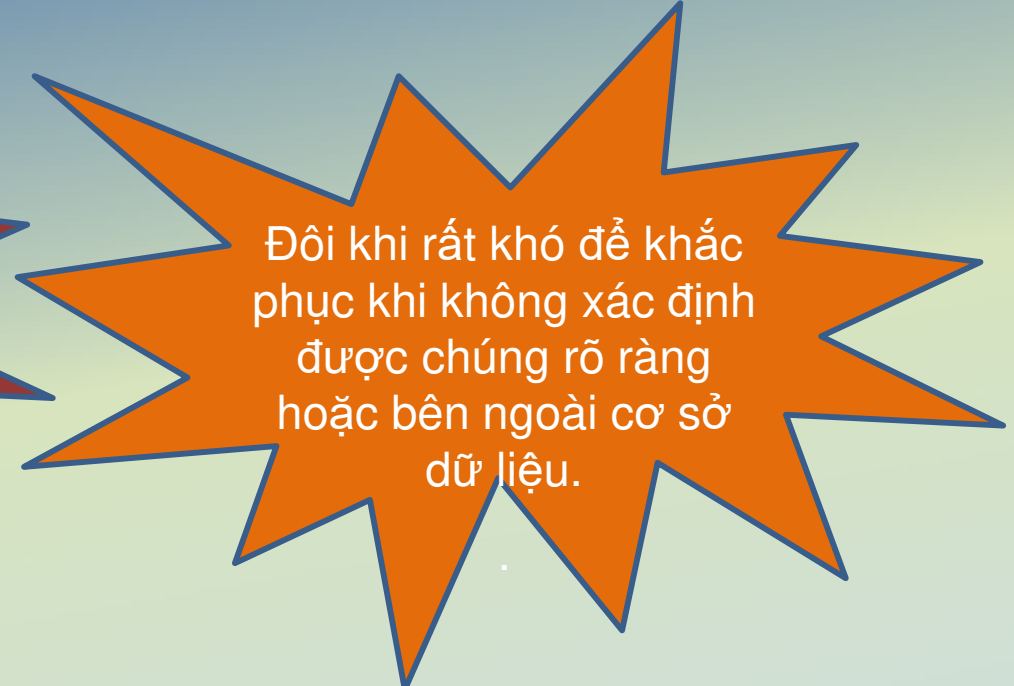
Được phát hiện bởi SQL Server trước khi bắt đầu thực thi xử lý khối Transact-SQL hoặc thủ tục lưu (stored procedure).

## Types of Errors 2-2

### ➤ Lỗi lúc thực thi (Run-time Errors)



Là các lỗi xảy ra khi ứng dụng cố gắng thực hiện một hành động không được hỗ trợ bởi SQL Server và hệ điều hành.



Đôi khi rất khó để khắc phục khi không xác định được chúng rõ ràng hoặc bên ngoài cơ sở dữ liệu.

- Một số trường hợp của lỗi lúc thực thi như sau:
- Thực hiện một phép tính như chia 0.
  - Cố gắng để thực thi mã không được xác định rõ ràng.

# Triển khai quản lý lỗi

Điều quan trọng nhất mà người dùng cần quan tâm là việc quản lý lỗi(error handling).

Người dùng cũng phải quan tâm quản lý lỗi và ngoại lệ(exception) trong khi thiết kế cơ sở dữ liệu.

Các cơ chế quản lý lỗi khác nhau như sau:

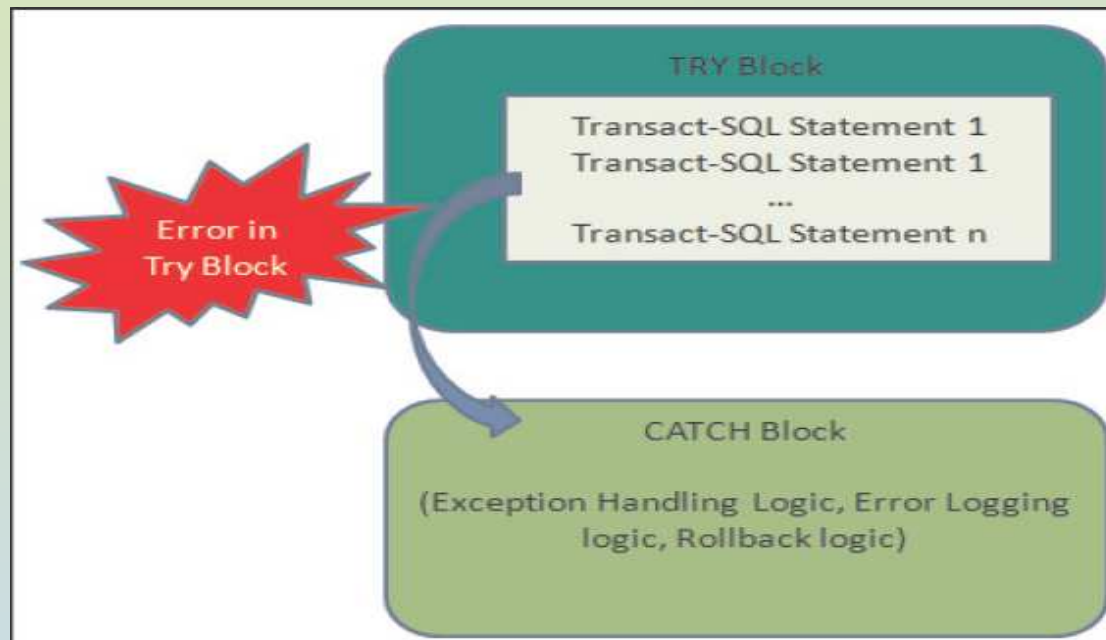
- Khi thi hành(executing) một số các câu lệnh như INSERT, DELETE, và UPDATE, người dùng có thể xử lý(handle) các lỗi để đảm bảo ra kết quả đúng.
- Khi một giao dịch(transaction) thất bại(fails) và người dùng cần hủy bỏ(rollback) giao dịch đó, một thông báo phù hợp được hiển thị.
- Khi làm việc với các cursors(con trỏ) trong SQL Server, người dùng có thể xử lý các lỗi để đảm bảo kết quả đúng.

# TRY...CATCH 1-3

Được sử dụng để thực thi xử lý các ngoại lệ trong Transact-SQL.

Có thể chứa(enclose) một hoặc nhiều câu lệnh Transact-SQL vào trong một khối TRY.

Nếu có một lỗi xảy ra trong khối TRY, chuyển điều khiển đến khối CATCH là khối có thể chứa một hoặc nhiều câu lệnh.



# TRY...CATCH 2-3

## Cú pháp:

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

Trong đó,

sql\_statement: chỉ ra câu lệnh Transact-SQL bất kỳ.

statement\_block: chỉ ra nhóm các câu lệnh Transact-SQL trong khối BEGIN...END.

Cấu trúc TRY ... CATCH sẽ bắt tất cả các lỗi lúc thực thi(run-times) có mức độ nghiêm trọng (severity level) cao hơn 10 và không đóng kết nối cơ sở dữ liệu.

Khối TRY...CATCH không thể trải ra trên nhiều lô (batches) hoặc khối (blocks) các câu lệnh Transact-SQL.



# TRY...CATCH 3-3

- Đoạn code dưới đây trình bày một ví dụ đơn giản của câu lệnh TRY...CATCH:

```
BEGIN TRY
    DECLARE @num int;
    SELECT @num=217/0;
END TRY
BEGIN CATCH
    PRINT 'Error occurred, unable to divide by 0'
END CATCH;
```

Cả hai khối TRY và CATCH có thể chứa các cấu trúc TRY...CATCH lồng

Nếu khối CATCH có chứa (encloses) một cấu trúc TRY...CATCH lồng, bất kỳ lỗi nào trong khối TRY lồng sẽ chuyển(pass) điều khiển điều khiển tới khối CATCH lồng

Nếu không có cấu trúc TRY...CATCH lồng, lỗi được truyền(pass) quay lại nơi gọi

Cấu trúc TRY...CATCH cũng có thể bắt các lỗi không được xử lý từ các trigger hoặc thủ tục lưu thực thi thông qua code trong khối TRY

# Thông tin lỗi 1-3

Thực hành tốt là để hiển thị thông tin lỗi cùng với các lỗi, do đó, nó có thể giúp giải quyết các lỗi một cách nhanh chóng và hiệu quả.

Trong phạm vi của khối CATCH, các hàm hệ thống sau đây có thể được dùng để lấy các thông tin về lỗi là nguyên nhân làm cho khối CATCH được thực thi:

➤ Dưới đây là các hàm hệ thống:

- `ERROR_NUMBER ( )` : trả về một số là mã của lỗi.
- `ERROR_SEVERITY ( )` : returns the severity.
- `ERROR_STATE ( )` : trả về một số là mã trạng thái.
- `ERROR_PROCEDURE ( )` : trả về tên của trigger hoặc stored procedure là nơi mà lỗi xảy ra.
- `ERROR_LINE ( )` : trả về một số là vị trí của dòng gây ra lỗi.
- `ERROR_MESSAGE ( )` : Trả về văn bản(text) đầy đủ của lỗi. Văn bản có chứa giá trị được cung cấp cho các tham số như tên đối tượng, độ dài, hoặc thời gian.

Các hàm trả về `NULL` khi chúng được gọi bên ngoài phạm vi của khối `CATCH` .

# Thông tin lỗi 2-3

## Sử dụng TRY...CATCH với thông tin lỗi

- Đoạn code dưới đây trình bày một ví dụ đơn giản việc hiển thị thông tin lỗi:

```
USE AdventureWorks2012;
GO
BEGIN TRY
    SELECT 217/0;
END TRY
BEGIN CATCH
    SELECT ERROR_NUMBER() AS ErrorNumber,
           ERROR_SEVERITY() AS ErrorSeverity,
           ERROR_LINE() AS ErrorLine,
           ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
GO
```

### Kết quả

(No column name)				
	ErrorNumber	ErrorSeverity	ErrorLine	ErrorMessage
1	8134	16	2	Divide by zero error encountered.

# Thông tin lỗi 3-3

## Sử dụng TRY...CATCH với giao tác (Transaction)

- Đoạn code dưới đây trình bày một ví dụ về các công việc bên trong một giao tác.

```
USE AdventureWorks2012;
GO
BEGIN TRANSACTION;
    BEGIN TRY
        DELETE FROM Production.Product WHERE ProductID = 980;
    END TRY
    BEGIN CATCH
        SELECT      ERROR_SEVERITY() AS ErrorSeverity
                  ,ERROR_NUMBER() AS ErrorNumber
                  ,ERROR_PROCEDURE() AS ErrorProcedure
                  ,ERROR_STATE() AS ErrorState
                  ,ERROR_MESSAGE() AS ErrorMessage
                  ,ERROR_LINE() AS ErrorLine;
        IF @@TRANCOUNT > 0
            ROLLBACK TRANSACTION;
    END CATCH;
    IF @@TRANCOUNT > 0
        COMMIT TRANSACTION;
GO
```

# Need for Transactions

## Giao dịch không thể hoàn tất (Uncommittable Transactions)

Một lỗi được phát sinh trong một khối TRY, gây ra trạng thái của giao dịch hiện tại là không hợp lệ và được xem như là giao dịch không thể hoàn tất

Một giao dịch không hoàn tất chỉ thực hiện `ROLLBACK TRANSACTION` hoặc các thao tác đọc.

Hàm `XACT_STATE` trả về -1 nếu giao dịch được phân loại như là một giao dịch không hoàn thành.

Giao dịch không thực thi bất kỳ lệnh Transact-SQL nào mà thực hiện `COMMIT TRANSACTION` hoặc thao tác ghi.

## @@ERROR 1-2

Hàm @@ERROR trả về mã số lỗi của câu lệnh T-SQL được thực thi gần nhất(last)

### Cú pháp:

@@ERROR

- Hàm trả về một giá trị kiểu nguyên(integer)
- Hàm trả về 0, nếu câu lệnh Transact-SQL trước đó không xảy ra lỗi
- Hàm trả về một mã số lỗi, nếu các câu lệnh trước đó gặp lỗi
- Người dùng có thể xem nội dung lỗi bằng văn bản gắn với mã số lỗi @@ERROR trong view catalog `sys.messages`

## @@ERROR 2-2

- Đoạn code dưới đây cho thấy cách sử dụng @@ERROR để kiểm tra sự vi phạm ràng buộc:

```
USE AdventureWorks2012;
GO
BEGIN TRY
    UPDATE HumanResources.EmployeePayHistory
    SET PayFrequency = 4
    WHERE BusinessEntityID = 1;
END TRY
BEGIN CATCH
    IF @@ERROR = 547
        PRINT N'Check constraint violation has occurred.';
END CATCH
```

### Kết quả:

```
Check constraint violation has occurred.
```

# RAISERROR 1-5

RAISERROR được sử dụng để trả các thông điệp quay lại các ứng dụng có dạng giống như dạng của các thông điệp lỗi hoặc cảnh báo của hệ thống do SQL Server Database Engine phát ra.

Có thể tham khảo thông điệp(message) do người dùng định nghĩa được lưu trữ trong catalog view `sys.messages` hoặc xây dựng các thông báo lỗi động (dynamic error messages) tại lúc thực thi (run-time).

Sử dụng RAISERROR để:

- Giúp đỡ trong việc xử lý sự cố trong code Transact-SQL.
- Kiểm tra các giá trị của dữ liệu.
- Trả về các thông điệp chứa văn bản có thể thay đổi được.
- Gây ra(cause) việc thực thi nhảy từ khối TRY sang khối CATCH liên quan.
- Trả về thông tin lỗi từ khối CATCH tới ứng dụng hoặc khối(batch) đang gọi.



## Cú pháp:

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

Trong đó,

`msg_id`: chỉ ra mã số của thông điệp lỗi do người dùng định nghĩa được lưu trữ trong view `sys.messages` bằng thủ tục `sp_addmessage`.

`msg_str`: Chỉ ra các thông điệp người dùng định nghĩa với việc định dạng(formatting). `msg_str` là một chuỗi kí tự đặc tả chuyển đổi (conversion specifications) giống như hàm ***printf*** của ngôn ngữ **C**. Đặc tả chuyển đổi có dạng như sau:

```
% [[flag] [width] [.precision] [{h | l}]] type
```

# RAISERROR 3-5

- Các tham số có thể được sử dụng trong `msg_str` như sau:
  - `{h | l} type`: chỉ ra sử dụng các kiểu kí tự d, i, o, s, x, X, or u, và tạo các giá trị `shortint(h)` hoặc `longint(l)`.
- Dưới đây là một số đặc tả kiểu:
  - d hoặc i: xác định số nguyên có dấu.
  - o: chỉ ra số hệ bát phân(octal) không dấu.
  - x hoặc X: chỉ ra số hệ 16 (hexadecimal) không dấu.
  - flag: chỉ ra code xác định khoảng cách và căn lề của giá trị thay thế. Điều này có thể bao gồm các kí hiệu(symbols) như – (trừ) và + (cộng) để chỉ ra căn lề trái hoặc chỉ ra giá trị là loại được kí hiệu tương ứng.
  - precision: chỉ ra số lượng kí tự tối đa được lấy từ giá trị tham số của các giá trị chuỗi.
  - width: Chỉ ra một số nguyên để xác định độ rộng tối thiểu cho các trường(field) mà giá trị của đối số sẽ được đặt vào đó.

`@local_variable`: Chỉ ra một biến có kiểu dữ liệu kí tự hợp lệ.

`severity`: Các mức độ nghiêm trọng (severity levels) từ 0 đến 18 được xác định bởi người dùng. Các mức từ 19 đến 25 được chỉ ra bởi các thành viên của `sysadmin`.

# RAISERROR 4-5

`@local_variable`: Chỉ ra một biến có kiểu dữ liệu kí tự hợp lệ.

`severity`: Các mức độ nghiêm trọng (severity levels) từ 0 đến 18 được xác định bởi người dùng. Các mức từ 19 đến 25 được chỉ ra bởi các thành viên của sysadmin.

`option`: chỉ tùy chọn tùy biến cho lỗi.

➤ Bảng dưới đây liệt kê các giá trị ch tùy chọn tùy biến:

Value	Description
LOG	Records the error in the error log and the application log for the instance of the Microsoft SQL Server Database Engine.
NOWAIT	Sends message directly to the client
SETERROR	Sets the ERROR_NUMBER and @@ERROR values to msg_id or 5000 irrespective of the severity level.

## RAISERROR 5-5

- Đoạn code sau đây trình bày cách xây dựng câu lệnh RAISERROR để hiển thị một câu lệnh lỗi được tùy biến:

```
RAISERROR (N'This is an error message %s %d.',  
          10, 1, N'serial number', 23);  
GO
```

### Kết quả:

```
This is error message serial number 23.
```

- Đoạn code dưới đây trình bày cách sử dụng câu lệnh RAISERROR để trả về cùng một chuỗi :

```
RAISERROR (N'%*.*s', 10, 1, 7, 3, N'Hello world');  
GO  
RAISERROR (N'%7.3s', 10, 1, N'Hello world');  
GO
```

Hàm hệ thống ERROR\_STATE trả về mã số trạng thái của lỗi đã gây ra khối CATCH của cấu trúc TRY...CATCH được thực thi.

## Cú pháp:

ERROR\_STATE ( )

- ERROR\_STATE được gọi từ bất cứ đâu bên trong phạm vi của khối CATCH.
  - ERROR\_STATE trả về trạng thái lỗi bất kể nó được thực thi bao nhiêu lần hoặc nó có được thực thi hay không bên trong phạm vi của khối CATCH.
- Đoạn code dưới đây trình bày cách dùng hàm ERROR\_STATE bên trong khối TRY:

```
BEGIN TRY
    SELECT 217/0;
END TRY
BEGIN CATCH
    SELECT ERROR_STATE() AS ErrorState;
END CATCH;
GO
```

# ERROR\_SEVERITY

Hàm ERROR\_SEVERITY trả về sự nghiêm trọng của lỗi đã làm cho khối CATCH trong cấu trúc TRY...CATCH được thực thi.

## Cú pháp:

```
ERROR_SEVERITY ( )
```

- ERROR\_SEVERITY có thể gọi ở bất cứ đâu bên trong phạm vi của khối CATCH.
- ERROR\_SEVERITY sẽ trả về sự nghiêm trọng của lỗi được chỉ ra tới khối CATCH, nơi được xem trong các khối CATCH lồng nhau.

➤ Đoạn code dưới đây trình bày cách hiển thị sự nghiêm trọng của lỗi:

```
BEGIN TRY
    SELECT 217/0;
BEGIN CATCH
    SELECT ERROR_SEVERITY() AS ErrorSeverity;
END CATCH;
GO
END TRY
```

# ERROR\_PROCEDURE

Hàm ERROR\_PROCEDURE trả về tên của trigger hoặc stored procedure, là nơi mà lỗi vừa xảy ra.

## Cú pháp:

```
ERROR_PROCEDURE ( )
```

- ERROR\_PROCEDURE có thể gọi ở bất cứ đâu bên trong phạm vi của khối CATCH.
- ERROR\_PROCEDURE trả về tên của trigger hoặc tên của stored procedure.

➤ Đoạn code dưới đây trình bày cách sử dụng hàm ERROR\_PROCEDURE:

```
USE AdventureWorks2012;  
GO  
IF OBJECT_ID ( 'usp_Example', 'P'  
    ) IS NOT NULL  
DROP PROCEDURE usp_Example;  
GO  
CREATE PROCEDURE usp_Example  
AS SELECT 217/0;  
GO
```

```
BEGIN TRY  
EXECUTE usp_Example;  
END TRY  
BEGIN CATCH  
SELECT ERROR_PROCEDURE() AS  
    ErrorProcedure;  
END CATCH;  
GO
```

# ERROR\_NUMBER

Hàm hệ thống ERROR\_NUMBER khi được gọi trong khối CATCH trả về mã số của lỗi gây ra khối CATCH trong cấu trúc TRY...CATCH được thực thi.

## Cú pháp:

```
ERROR_NUMBER ( )
```

- ERROR\_NUMBER trả về mã số của lỗi bất kể nó được thực thi bao nhiêu lần hoặc nó có được thực thi hay không bên trong phạm vi khối CATCH.
- Đoạn code dưới đây trình bày cách sử dụng ERROR\_NUMBER trong khối CATCH

```
BEGIN TRY  
SELECT 217/0;  
END TRY  
BEGIN CATCH  
SELECT ERROR_NUMBER() AS ErrorNumber;  
END CATCH;  
GO
```



# ERROR\_MESSAGE

Hàm `ERROR_MESSAGE` trả về thông điệp dạng văn bản của lỗi đã làm cho khối `CATCH` trong cấu trúc `TRY...CATCH` được thực thi.

## Cú pháp:

```
ERROR_MESSAGE ( )
```

- Hàm `ERROR_MESSAGE` được gọi trong khối `CATCH`, nó trả về văn bản đầy đủ của thông điệp lỗi đã làm cho khối `CATCH` được thực thi.
  - `ERROR_MESSAGE` trả về `NULL` nếu được gọi bên ngoài phạm vi khối `CATCH`.
- Đoạn code dưới đây trình bày cách sử dụng `ERROR_MESSAGE` trong khối `CATCH`:

```
BEGIN TRY
SELECT 217/0;
END TRY
BEGIN CATCH
SELECT ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
GO
```

# ERROR\_LINE

Hàm `ERROR_LINE` trả về thứ tự của dòng (line number) trong cấu trúc `TRY...CATCH` mà lỗi xảy ra ở đó.

## Cú pháp:

```
ERROR_LINE ( )
```

- Hàm `ERROR_LINE` được gọi trong khối `CATCH`, nó trả về thứ tự của dòng mà lỗi đã xảy ra ở đó.
- `ERROR_LINE` trả về số thứ tự của dòng trong trigger hoặc stored procedure mà lỗi đã xảy ra.

➤ Đoạn code dưới đây cho thấy cách sử dụng hàm `ERROR_LINE` trong khối `CATCH` :

```
BEGIN TRY
SELECT 217/0;
END TRY
BEGIN CATCH
SELECT ERROR_LINE() AS ErrorLine;
END CATCH;
GO
```

# Các lỗi không bị ảnh hưởng bởi cấu trúc TRY...CATCH 1-3

- Cấu trúc TRY...CATCH không bắt các điều kiện sau:
  - Các thông điệp hoặc cảnh báo có một mức độ nghiêm trọng là 10 hoặc thấp hơn.
  - Một lỗi mà có một mức độ nghiêm trọng là 20 hoặc cao hơn làm ngừng(stop) ngay công việc mà SQL Server Database Engine đang xử lý cho phiên giao dịch(session).
  - Chú ý(attentions), như là sự kết nối client bị ngắt hoặc yêu cầu client của bị gián đoạn.
  - Khi phiên làm việc bị kết thúc do người quản trị hệ thống (system administrator) sử dụng lệnh `KILL`.
- Những loại lỗi sau đây không được xử lý bởi một khối CATCH khi chúng xảy ra ở cùng mức thực thi như là cấu trúc TRY ... CATCH:
  - Các lỗi biên dịch, ví dụ như lỗi cú pháp, làm ngăn cản chạy một lô(batch).
  - Các lỗi xảy ra trong quá trình biên dịch lại các lệnh, chẳng hạn như các lỗi phân giải tên đối tượng xảy ra sau khi biên dịch do việc phân giải tên bị hoãn lại.



## Các lỗi không bị ảnh hưởng bởi cấu trúc TRY...CATCH 2-3

- Ví dụ sau đây cho thấy một lỗi phân giải tên đối tượng được tạo ra như thế nào bởi một câu lệnh `SELECT`, và không được bắt bởi cấu trúc `TRY ... CATCH`, nhưng được bắt bởi khối `catch` khi câu lệnh `SELECT` tương tự được thực hiện bên trong một thủ tục lưu :

```
USE AdventureWorks2012;
GO
BEGIN TRY
    SELECT * FROM Nonexistent;
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH
```



# Các lỗi không bị ảnh hưởng bởi cấu trúc TRY...CATCH 3-3

- Đoạn code dưới đây cho thấy cách thông điệp lỗi được hiển thị trong một trường hợp:

```
IF OBJECT_ID ( N'sp_Example', N'P' ) IS NOT NULL
DROP PROCEDURE sp_Example;
GO
CREATE PROCEDURE sp_Example
AS
SELECT * FROM Nonexistent;
GO
BEGIN TRY
    EXECUTE sp_Example;
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber,
        ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

# THROW 1-2

Câu lệnh `THROW` gây ra(raises) một ngoại lệ (exception) và chuyển điều khiển của khối thực thi tới khối `CATCH` của cấu trúc `TRY...CATCH`.

## Cú pháp:

```
THROW [ { error_number | @local_variable },  
{ message | @local_variable }, { state | @local_variable }  
] [ ; ]
```

Trong đó,

`error_number`: chỉ ra một hần hoặc biến kiểu nguyên biểu diễn cho mã số của lỗi.

`message`: chỉ ra một biến hoặc chuỗi có kiểu `nvarchar(2048)` để định nghĩa nội dung thông điệp của ngoại lệ.

`state`: là một biến hoặc hằng số kiểu `tinyint` có giá trị trong khoảng 0 và 255 để chỉ ra trạng thái(state) để gắn với thông điệp.

# THROW 2-2

- Đoạn code dưới đây chỉ ra cách dùng câu lệnh THROW để gây ra ngoại lệ:

```
USE tempdb;
GO
CREATE TABLE dbo.TestRethrow ( ID INT PRIMARY KEY );
BEGIN TRY
    INSERT dbo.TestRethrow(ID) VALUES (1);
    INSERT dbo.TestRethrow(ID) VALUES (1);
END TRY
BEGIN CATCH
    PRINT 'In catch block.';
    THROW;
END CATCH;
```

## Kết quả:

(1 row(s) affected)

(0 row(s) affected)

In catch block.

Msg 2627, Level 14, State 1, Line 6

Violation of PRIMARY KEY constraint 'PK\_\_TestReth\_\_3214EC27AAB15FEE'.  
Cannot insert duplicate key in object 'dbo.TestRethrow'. The duplicate  
key value is (1).

## THROW 3-2

- Đoạn code dưới đây chỉ thấy cách dùng hàm FORMATMESSAGE với THROW để ném ra một thông điệp lỗi tùy ý:

```
EXEC sys.sp_addmessage      @msgnum    = 60000, @severity = 16
, @msgtext = N'This is a test message with one numeric
    parameter (%d), one string parameter (%s), and another string
    parameter (%s).' , @lang = 'us_english';
GO

DECLARE @msg NVARCHAR(2048) = FORMATMESSAGE(60000, 500, N'First
    string', N'second string');

THROW 60000, @msg, 1;
```

### Kết quả:

Msg 60000, Level 16, State 1, Line 2

This is a test message with one numeric parameter (500), one string parameter (First string), and another string parameter (second string).



- Lỗi cú pháp là các lỗi xảy ra khi code không thể phân tích được bởi SQL Server.
- Lỗi thực thi(Run-time)xảy ra khi ứng dụng cố gắng để thực hiện một hành động mà không được hỗ trợ bởi Microsoft SQL Server và bởi hệ điều hành.
- Câu lệnh TRY ... CATCH được sử dụng để xử lý các trường hợp ngoại lệ trong Transact-SQL.
- Cấu trúc TRY ... CATCH cũng có thể bắt các lỗi không được quản lý từ các trigger hoặc thủ tục lưu mà thực thi thông qua code trong một khối TRY.
- Câu lệnh GOTO có thể được sử dụng để nhảy đến một nhãn bên trong cùng khối TRY ... CATCH khối hoặc ra khỏi TRY ... CATCH.
- Các hàm hệ thống khác nhau có sẵn trong Transact-SQL để in thông tin lỗi về lỗi xảy ra.
- Câu lệnh RAISERROR được sử dụng để bắt đầu xử lý lỗi cho một phiên giao dịch và hiển thị thông báo lỗi.