Intelligent Data Management with SQL Server

Session: 15

Error Handling

Objectives

- Explain error handling and its implementation
- Describe the TRY-CATCH block
- Explain the procedure to display error information
- Describe the @@ERROR and RAISERROR statements
- Explain the use of ERROR_STATE, ERROR_SEVERITY, and ERROR_PROCEDURE
- Explain the use of ERROR_NUMBER, ERROR_MESSAGE, and ERROR_LINE
- Describe the THROW statement

Introduction

- Error handling in SQL Server is now easier through different techniques.
- SQL Server has introduced options that can help you to handle errors efficiently.
- Often, it is not possible to capture errors that occur at the user's end.
- SQL Server provides the TRY...CATCH statement that helps to handle errors effectively at the back end.
- Several system functions can print error related information, which can help fix errors easily.

Types of Errors 1-5

- As a Transact-SQL programmer, one must be aware of various types of errors that can occur while working with SQL Server statements.

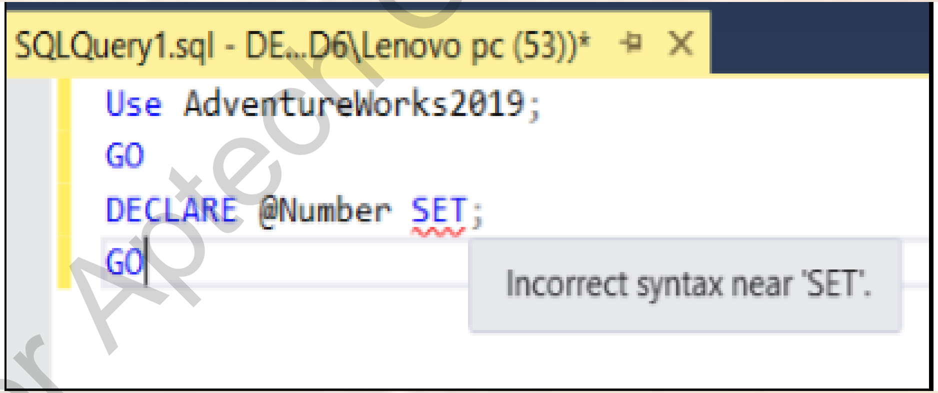
- The first step one can perform is to identify the type of the error and then, determine how to handle or overcome it.

Types of Errors 2-5

Some types of errors are as follows:

Syntax Errors

- Syntax errors are the errors that occur when code cannot be parsed by SQL Server. Such errors are detected by SQL Server before beginning the execution process of a Transact-SQL block or stored procedure.



SQLQuery1.sql - DE...D6\Lenovo pc (53))*

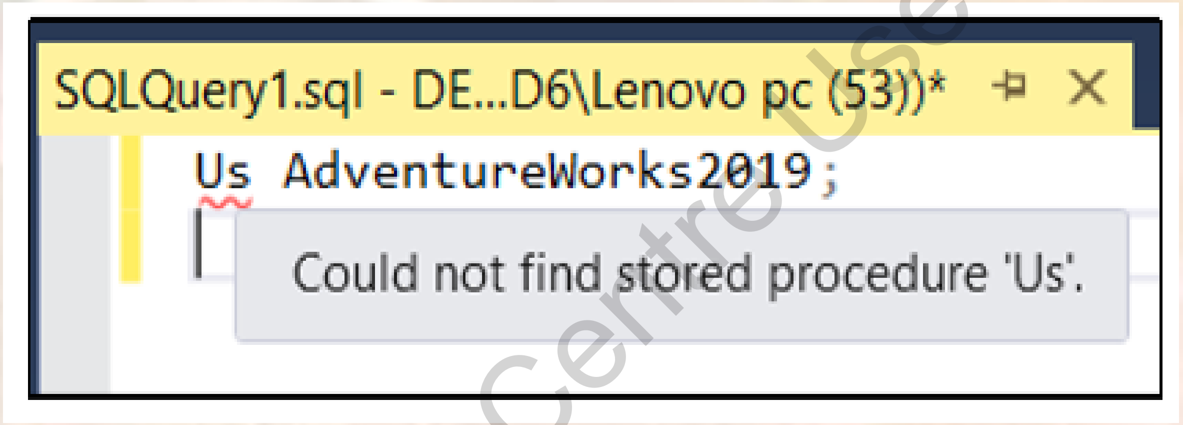
```
Use AdventureWorks2019;
GO
DECLARE @Number SET;
GO
```

```
Incorrect syntax near 'SET'.
```

Syntax Error

Types of Errors 3-5

If a user types a keyword or an operator wrongly because the user does not remember the valid usage, the code editor will appropriately indicate it.



```
Us AdventureWorks2019;
```

Could not find stored procedure 'Us'.

Wrong Keyword Error

If the user forgets to type something that is required, the code editor will display an error once the user executes that statement.

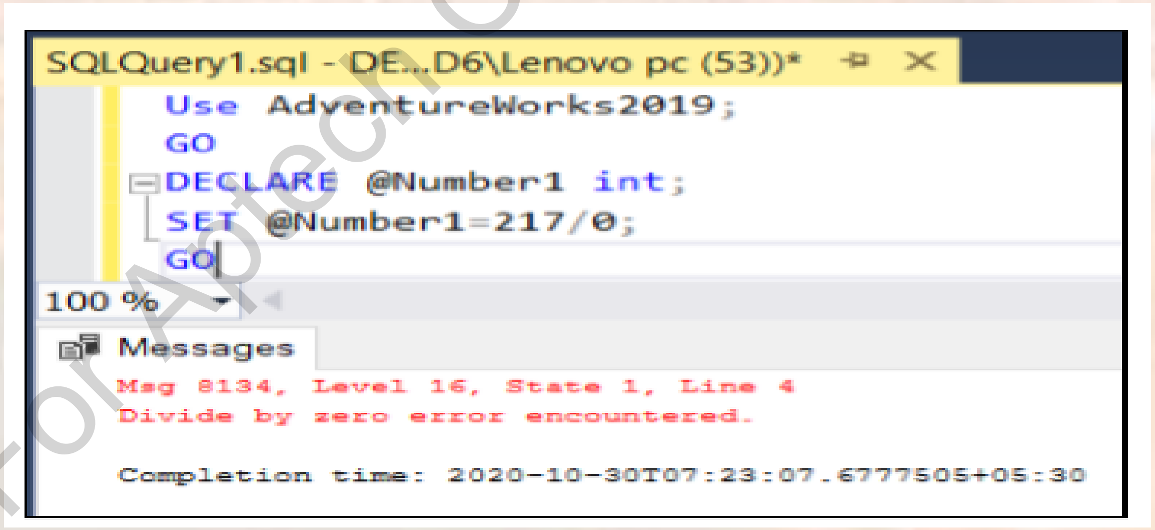
- Syntax errors are easily identified as the code editor points them out. However, if users use a command-based application such as sqlcmd, the error is shown only after the code is executed.

Types of Errors 4-5

Run-time Errors

- Run-time errors are errors that occur when the application tries to perform an action that is supported neither by SQL Server nor by the operating system.

Some instances where run-time errors can occur are as follows:
- Performing a calculation such as division by zero
- Trying to execute code that is not defined clearly



SQLQuery1.sql - DE...D6\Lenovo pc (53))*

```
Use AdventureWorks2019;
GO
DECLARE @Number1 int;
SET @Number1=217/0;
GO
```

100 %

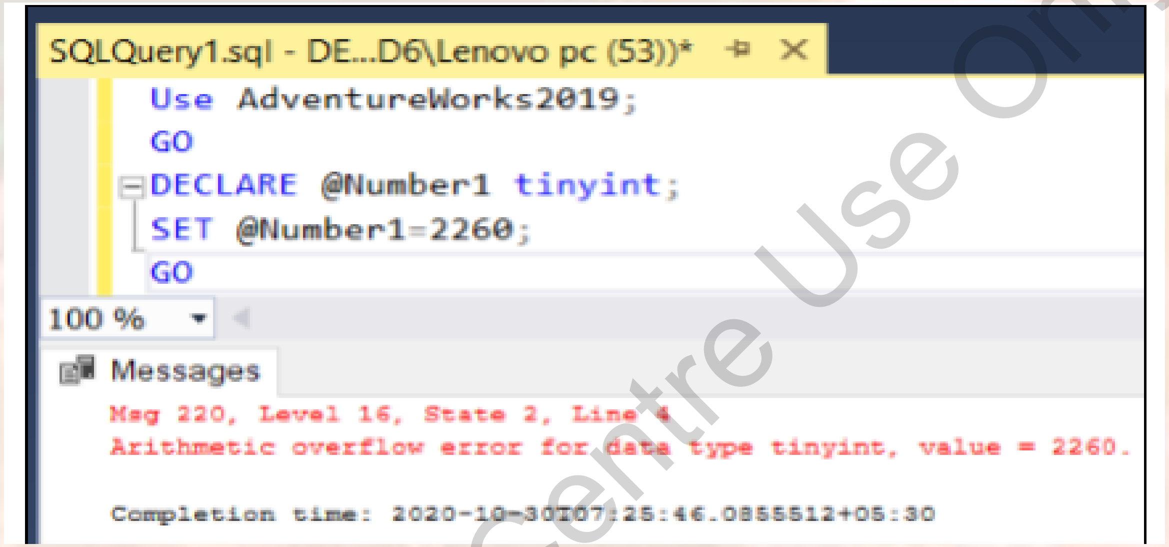
Messages

```
Msg 8134, Level 16, State 1, Line 4
Divide by zero error encountered.

Completion time: 2020-10-30T07:23:07.6777505+05:30
```

Divide by Zero Error

Types of Errors 5-5



```
Use AdventureWorks2019;
GO
DECLARE @Number1 tinyint;
SET @Number1=2260;
GO
```

```
Msg 220, Level 16, State 2, Line 4
Arithmetic overflow error for data type tinyint, value = 2260.

Completion time: 2020-10-30T07:25:46.0855512+05:30
```

Following are some common situations when run-time error occurs:

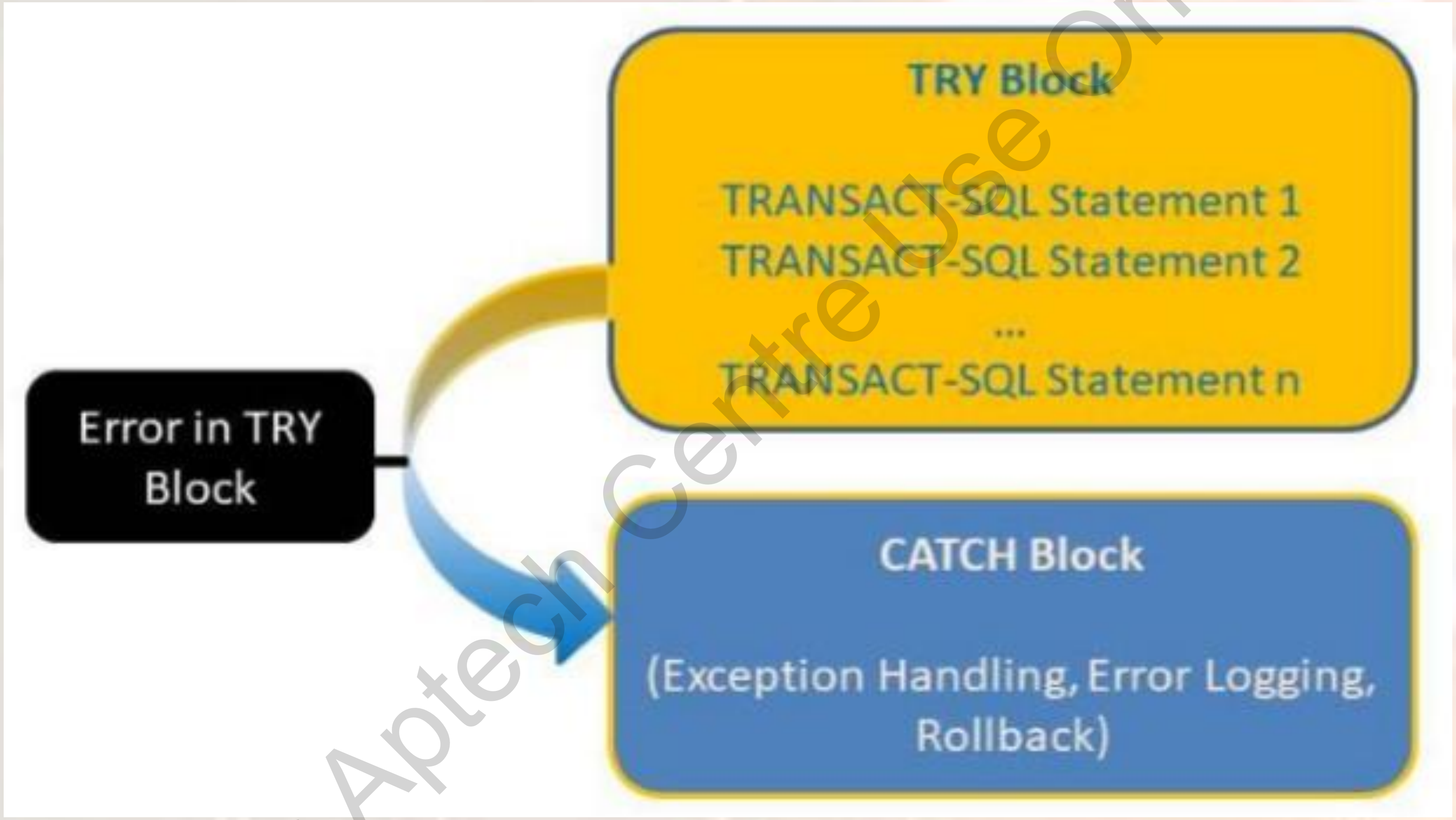
- Using a stored procedure, or a function, or a trigger that is unavailable
- Trying to perform an action that an object or a variable cannot handle
- Attempting to access or use computer memory that is insufficient
- Trying to perform an action on incompatible types
- Using wrongly conditional statements

Implementing Error Handling

- While developing any application, one of the most important things that users need to take care of is error handling.
- In the same way, users also have to take care of handling exception and errors while designing the database.
- Various error handling mechanisms can be used.

- When executing some DML statements such as INSERT, DELETE, and UPDATE, users can handle errors to ensure correct output.
- When a transaction fails and the user must roll back the transaction, an appropriate error message can be displayed.
- When working with cursors in SQL Server, users can handle errors to ensure correct results.

TRY...CATCH



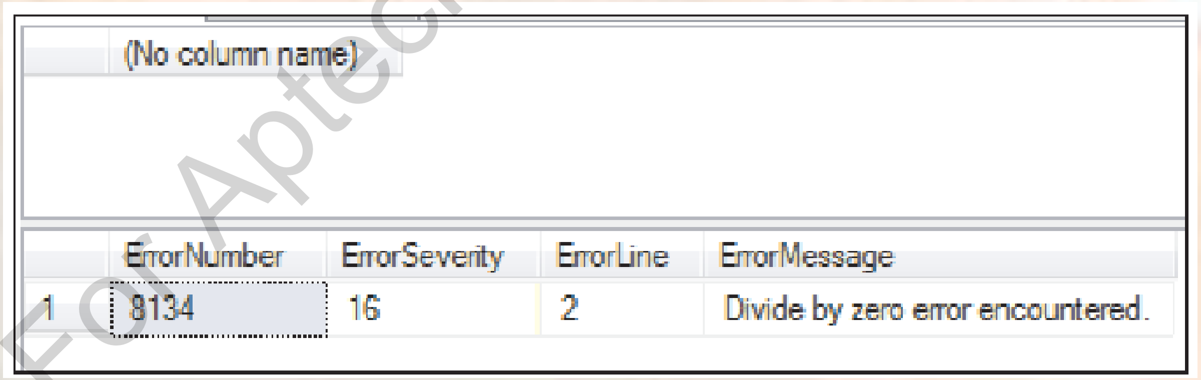
TRY...CATCH Logic

Information 1-3

- It is a good practice to display error information along with the error, so that it can help to solve the error quickly and efficiently.

- To achieve this, system functions need to be used in the CATCH block to find information about the error that initiated the CATCH block to execute.

Using TRY...CATCH with error information



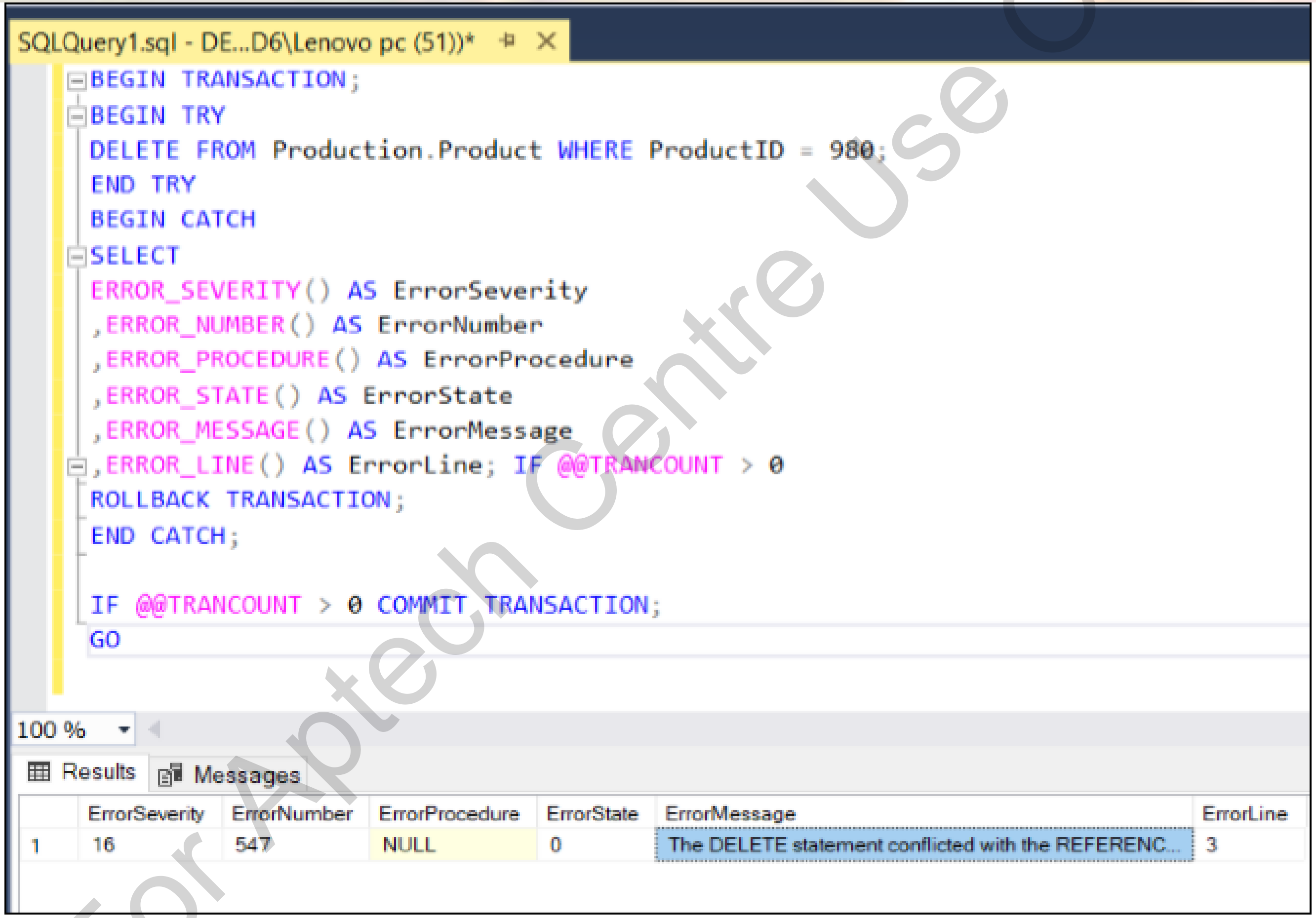
```
   (No column name)

   ErrorNumber   ErrorSeverity   ErrorLine   ErrorMessage
1  8134          16              2           Divide by zero error encountered.
```

Error Information

Information 2-3

Using TRY...CATCH with Transaction



SQLQuery1.sql - DE...D6\Lenovo pc (51))*

```sql
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
```

100 %

Results | Messages

	ErrorSeverity	ErrorNumber	ErrorProcedure	ErrorState	ErrorMessage	ErrorLine
1	16	547	NULL	0	The DELETE statement conflicted with the REFERENC...	3

Constraint Violation Error

Information 3-3

Uncommittable Transactions

- If an error is generated in a TRY block, it causes the state of the current transaction to be invalid and the transaction is considered as an uncommitted transaction.

- An uncommittable transaction performs only ROLLBACK TRANSACTION or read operations.

- The transaction does not execute any Transact-SQL statement that performs a COMMIT TRANSACTION or a write operation.

@@ERROR

The @@ERROR function returns the error number for the last Transact-SQL statement executed.

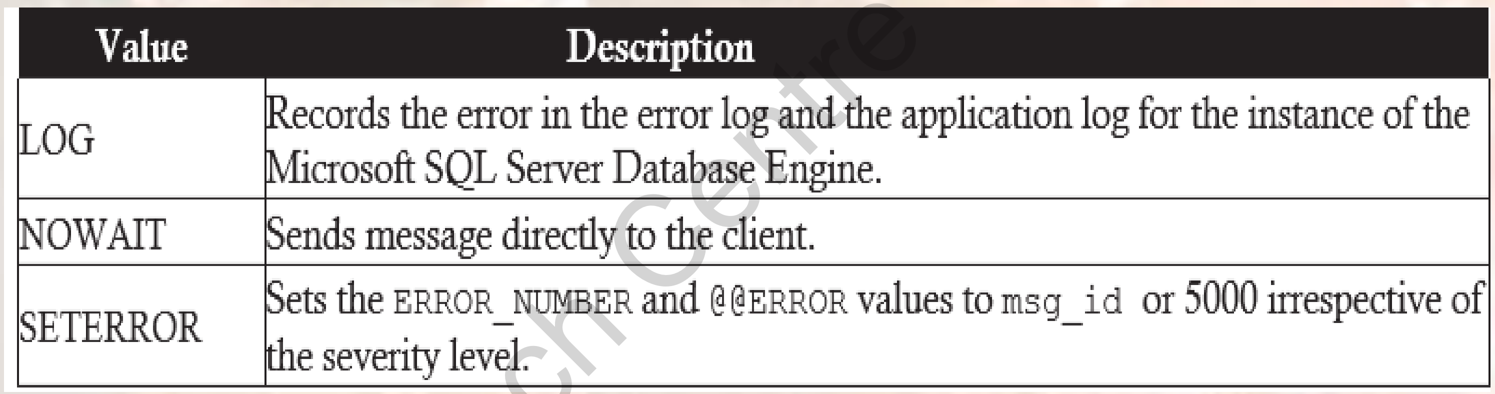
```
USE AdventureWorks2019;
GO
BEGIN TRY
UPDATE HumanResources.EmployeePayHistory SET PayFrequency = 4
WHERE BusinessEntityID = 1;
END TRY
BEGIN CATCH
IF @@ERROR = 547
PRINT N'Check constraint violation has occurred.';
END CATCH
```

It displays the following error message:

```
(0 rows affected)
Check constraint violation has occurred.
```

RAISERROR

- The RAISERROR statement starts error processing for a session and displays an error message.
- RAISERROR can reference a user-defined message stored in the sys.messages catalog view or build dynamic error messages at run-time.



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.

Type Specification Values

Following errors are returned back to the caller if RAISERROR executes:
- Out of scope of any TRY block
- Having severity of 10 or lower in TRY block
- Having severity of 20 or higher that terminates the database connection

ERROR_STATE

- The ERROR_STATE system function returns the state number of the error that causes the CATCH block of a TRY...CATCH construct to execute.

Syntax:

ERROR_STATE ( )

- When called in a CATCH block, it returns the state number of the error message that caused the CATCH block to be run.

- ERROR_STATE is called from anywhere within the scope of a CATCH block. ERROR_STATE returns the error state regardless of how many times it is executed or whether it is executed within the scope of the CATCH block.

ERROR_SEVERITY

- The ERROR_SEVERITY function returns the severity of the error that causes the CATCH block of a TRY…CATCH construct to be executed.

Syntax:
ERROR_SEVERITY ( )

- It returns a NULL value if called outside the scope of the CATCH block. ERROR_SEVERITY can be called anywhere within the scope of a CATCH block.

- In nested CATCH blocks, ERROR_SEVERITY will return the error severity that is specific to the scope of the CATCH block where it is referenced. Users can use the ERROR_SEVERITY function in a CATCH block.

ERROR_PROCEDURE

- The ERROR_PROCEDURE function returns the trigger or a stored procedure name where the error has occurred that has caused the CATCH block of a TRY...CATCH construct to be executed.

Syntax:
ERROR_PROCEDURE( )

- It returns the nvarchar data type. When the function is called in a CATCH block, it will return the name of the stored procedure where the error occurred.

- ERROR_PROCEDURE can be called from anywhere in the scope of a CATCH block.

ERROR_NUMBER

- The ERROR_NUMBER system function when called in a CATCH block returns the error number of the error that causes the CATCH block of a TRY...CATCH construct to be executed.

Syntax:

```
ERROR_NUMBER()
```

- ERROR_NUMBER returns the error number irrespective of how many times it executes or whether it executes within the scope of a CATCH block.

ERROR_MESSAGE

- The ERROR_MESSAGE function returns the text message of the error that causes the CATCH block of a TRY…CATCH construct to execute.

Syntax:

ERROR_MESSAGE( )

When the ERROR_MESSAGE function is called in the CATCH block, it returns the full text of the error message that causes the CATCH block to execute.

The text includes the values that are supplied for any parameter that can be substituted such as object names, times, or lengths.

ERROR_LINE

- The ERROR_LINE function returns the line number at which the error occurred in the TRY...CATCH block.

Syntax:

```
ERROR_LINE( )
```

When this function is called in the CATCH block, it returns the line number where the error has occurred.

If the error has occurred within a trigger or a stored procedure, it returns the line number in that trigger or stored procedure.

Errors Unaffected by the TRY...CATCH Construct 1-2

The TRY...CATCH construct does not trap the following conditions:

- Informational messages or warnings having a severity of 10 or lower
- An error that has a severity of 20 or higher that stops the SQL Server Database Engine task processing for the session
- If errors occur that have severity of 20 or higher and the database connection is not interrupted, the TRY...CATCH will handle the error
- Attentions such as broken client connection or client-interrupted requests
- When the session ends because of the KILL statements used by the system administrator

Errors Unaffected by the TRY…CATCH Construct 2-2

Following types of errors are not handled by a CATCH block that occur at the same execution level as that of the TRY…CATCH construct:

- Compile errors such as syntax errors that restrict a batch from running
- Errors that arise in the statement-level recompilation such as object name resolution errors occurring after compiling due to deferred name resolution.

THROW

- The THROW statement raises an exception and transfers control of the execution to a CATCH block of a TRY...CATCH construct.

Following is the syntax of the THROW statement:

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

where,

error_number: specifies a constant or variable that represents the error_number as int.
message: specifies a variable or string that defines the exception message as nvarchar(2048).
state: specifies a variable or a constant between 0 and 255 that specifies the state to associate with state of message as tinyint.

Summary

- Syntax errors are the errors that occur when code cannot be parsed by SQL Server.
- Run-time errors occur when the application tries to perform an action that is supported neither by Microsoft SQL Server nor by the operating system.
- TRY...CATCH statements are used to handle exceptions in Transact-SQL.
- TRY...CATCH constructs can also catch unhandled errors from triggers or stored procedures that execute through the code in a TRY block.
- GOTO statements can be used to jump to a label inside the same TRY...CATCH block or to leave a TRY...CATCH block.
- Various system functions are available in Transact-SQL to print error information about the error that occurred.
- The RAISERROR statement is used to start the error processing for a session and displays an error message.