Contents

Try… Catch

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

Retrieving Error Details

- The following system functions can be used to obtain information about the error that caused the **CATCH** block to be executed :
  - ERROR_NUMBER: Returns the number of the error.
  - ERROR_SEVERITY: Returns the severity.
  - ERROR_STATE: Returns the error state number.
  - ERROR_PROCEDURE: Returns the name of the stored procedure or trigger where the error occurred.
  - ERROR_LINE: Returns the line number inside the routine that caused the error.
  - ERROR_MESSAGE: Returns the complete text of the error message. The text includes the values supplied for any substitutable parameters, such as lengths, object names, or times.

Try... Catch Example

```
CREATE PROCEDURE InsertDept @DNumber INT, @DName VARCHAR(20),
                                    @MgrSSN CHAR(9), @MgrStartDate DATE

AS BEGIN
        BEGIN TRY
                INSERT INTO DEPARTMENT
                VALUES (@DName, @DNumber, @MgrSSN, @MgrStartDate);
        END TRY
        BEGIN CATCH
                DECLARE @ErrorMessage VARCHAR(255);
                DECLARE @ErrorSeverity INT;
                DECLARE @ErrorState INT;
                SELECT @ErrorMessage = ERROR_MESSAGE(),
                        @ErrorSeverity = ERROR_SEVERITY(),
                        @ErrorState = ERROR_STATE();
                RAISERROR(@ErrorMessage, @ErrorSeverity, @ErrorState);
        END CATCH;
END;
```

Error Propagation

- THROW statement:

  THROW [error_number, message, state];

  - **error_number:**
    - A constant or variable that represents the exception.
    - The error_number argument is **INT.**
    - must be greater than or equal to 50,000, and less than or equal to 2,147,483,647.

  - **message:**
    - A string or variable that describes the exception.
    - The message argument is **NVARCHAR(2048).**

  - **state:**
    - A constant or variable between 0 and 255 that indicates the state to associate with the message.
    - The state argument is **TINYINT**.

Error Propagation

- RAISERROR statement:

  RAISERROR ({ msg_id | msg_str} { ,severity , state });

  - **msg_id:**
    - A user-defined error message number stored in the **sys.messages** catalog view using sp_addmessage.
    - Error numbers for user-defined error messages *should be greater than 50000.*
    - When msg_id isn't specified, RAISERROR raises an error message with an error number of 50000.
  - **msg_str:** A user-defined message with formatting similar to the printf function in the C standard library.
  - **severity:**
    - The user-defined severity level associated with this message.
    - When using msg_id to raise a user-defined message, the severity specified on RAISERROR overrides the severity specified in sp_addmessage.
  - **state:**
    - An integer from 0 through 255.
    - Negative values default to 1.

Throw Example

```
BEGIN TRY
        DELETE FROM Employees
        WHERE SSN= ‘123456788’ ;
END TRY
BEGIN CATCH
        THROW 51000, ' The record does not exist.', 1;
END CATCH
```

Differences between RAISERROR and THROW

RAISERROR statement	THROW statement
If a msg_id is passed to RAISERROR, the ID must be defined in sys.messages.	The error_number parameter doesn't have to be defined in sys.messages.
The msg_str parameter can contain printf formatting styles.	The message parameter doesn't accept printf style formatting.
The severity parameter specifies the severity of the exception.	There's no severity parameter. When THROW is used to initiate the exception, the severity is always set to 16. However, when THROW is used to rethrow an existing exception, the severity is set to that exception's severity level.