# Murach Chapter 15 How to Code User Defined Functions

The purpose of a user-defined function (UDF) is to encapsulate logic that calculates something, possibly based on input parameters, and return a result. UDFs are not allowed to have any side effects. This obviously means that UDFs are not allowed to apply any schema or data changes in the database. This includes TRY. . .CATCH error catching.

SQL Server provides you with tools to handle errors in your T-SQL code. The main tool used for error handling is a construct called TRY. . .CATCH. SQL Server also provides a set of functions that you can invoke to get information about the error. If a TRY. . .CATCH block captures and handles an error, as far as the caller is concerned, there was no error. Typically, error handling involves some work in the CATCH block investigating the cause of the error and taking a course of action. SQL Server gives you information about the error via a set of functions. The ERROR\_NUMBER function returns an integer with the number of the error and is probably the most important of the error functions. The CATCH block usually includes flow code that inspects the error number to determine what course of action to take. The ERROR\_MESSAGE function returns error message text. To get the list of error numbers and messages, query the sys.messages catalog view. The ERROR\_SEVERITY and ERROR\_STATE functions return the error severity and state. The ERROR\_LINE function returns the line number where the error happened. Finally, the ERROR\_PROCEDURE function returns the name of the procedure in which the error happened and returns NULL if the error did not happen within a procedure.

First run the following code, which creates a database named "tc" and create a table called dbo.Employees in the database "tc".

1. Create a procedure that will inserts a new row into the Employees table you just created (create or alter procedure [dbo].[error\_procedur](@id int, @name VARCHAR(25), @manager int). The procedure has three input parameters as show above, to be used in the DML insert statement (INSERT INTO tc.dbo.Employees VALUES(@id, @name, @manager);). Wrap the insert statement in a TRY block, and if an error occurs, shows how to identify the error by inspecting the ERROR\_NUMBER function in the CATCH block. The code will also call a function that you create (see #2) and pass as a parameter the error code. The procedure will then print out the text string returned by the function, and re-throws the error.
2. Create a function that will receive the input parameter of the error generated by the above procedure you created in #1. The function will then return the appropriate error message: See below.

Error 2627 returns: 'Handling PK violation'

Error 547 returns: 'Handling CHECK/FK constraint violation'

Error 515 returns: 'Handling NULL violation'

Error 245 returns: 'Handling conversion error'

Use exec dbo.error\_procedur @id = 1, @name = 'Emp1', @manager = NULL; to test your function and procedure. When you run this code for the first time, the new row is inserted into the Employees table successfully, and therefore the CATCH block is skipped. You get the following output. (1 row(s) affected) When you run the same code a second time, the INSERT statement fails, control is passed to the CATCH block, and a primary key violation error is identified. You get the following output. To see other errors, run the code with the values 0, ‘A’, and NULL as the employee ID (see below);

I know there are far easier ways of doing this but this assignment will allow you to focuse on creating procedures, functions, and error handling without a lot of coding. In the attached example document, the black text is from you procedure that you code, and the red is the error handling (see above description).

create database [tc];

go

use [tc];

go

CREATE TABLE dbo.Employees

(

empid INT NOT NULL,

empname VARCHAR(25) NOT NULL,

mgrid INT NULL,

CONSTRAINT PK\_Employees PRIMARY KEY(empid),

CONSTRAINT CHK\_Employees\_empid CHECK(empid > 0),

CONSTRAINT FK\_Employees\_Employees

FOREIGN KEY(mgrid) REFERENCES dbo.Employees(empid)

);

use [tc];

go

exec dbo.error\_procedur @id = 1, @name = 'Emp1', @manager = NULL;

use [tc];

go

exec dbo.error\_procedur @id = 0, @name = 'Emp1', @manager = NULL;

use [tc];

go

exec dbo.error\_procedur @id = 'A', @name = 'Emp1', @manager = NULL;

use [tc];

go

exec dbo.error\_procedur @id = null, @name = 'Emp1', @manager = NULL;

Answer:

Part 1

create procedure error\_procedur(@id int, @name VARCHAR(25), @manager int)

as

BEGIN

DECLARE @errorNumber INT

BEGIN TRY

INSERT INTO tc.dbo.Employees VALUES(@id, @name, @manager)

END TRY

BEGIN CATCH

SELECT @errorNumber= ERROR\_NUMBER()

DECLARE @viewMessage NVARCHAR(MAX)

DECLARE @message VARCHAR(MAX) = ERROR\_MESSAGE()

DECLARE @severity INT = ERROR\_SEVERITY()

DECLARE @state INT = ERROR\_STATE()

SELECT @viewMessage= tc.dbo.errorCheck(@errorNumber)

PRINT @viewMessage;

RAISERROR (@message,@severity, @state)

END CATCH

END

Part 2

create FUNCTION errorCheck (@errorNumber INT)

RETURNS NVARCHAR(MAX)

AS

BEGIN

DECLARE @message NVARCHAR(MAX)

if @errorNumber= 2627

BEGIN

SET @message= 'Handling PK violation...'

END

if @errorNumber= 547

BEGIN

SET @message= 'Handling CHECK/FK constraint violation...'

END

if @errorNumber= 515

BEGIN

SET @message= 'Handling NULL violation...'

END

if @errorNumber= 245

BEGIN

SET @message= 'Handling conversion error...'

END

RETURN @message

END

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated