Article: SQL – Using TRY and CATCH to handle errors
Date: 27/02/2012
Posted by: HeelpBook Staff
Source: Link
Permalink: Link

SQL Server – Using TRY...CATCH to Handle Errors

The **TRY...CATCH** statement in ***Transact-SQL*** allows you to detect and handle error conditions gracefully within your database applications. This statement is the cornerstone of **SQL Server** error handling and is an extremely important part of developing robust database applications.

Introducing TRY..CATCH

TRY..CATCH works by allowing you to specify two ***Transact-SQL*** statements: one that you wish to "***try***" and another that you wish to use to "***catch***" any errors that might arise. When SQL Server encounters a **TRY..CATCH** statement, it immediately executes the statement included in the **TRY** clause. If the TRY statement executes successfully, **SQL Server** simply moves on.

On the other hand, if your **TRY** statement generates an error, **SQL Server** executes the **CATCH** statement to gracefully handle the error.

TRY...CATCH Example

It's easiest to understand the use of this statement through the use of an example, so let's turn our attention to one. Imagine that you are the administrator of a human resources database that contains a table named "***employees***", containing information about each of the employees in your organization. That table uses an integer employee ID number as the primary key. You might attempt to use the statement below to insert a new employee into your database:

```
INSERT INTO employees(id, first_name, last_name, extension)

VALUES(12497, 'Mike', 'Chapple', 4201)
```

Under normal circumstances, this statement would add a row to the employees table. However, if an employee with **ID 12497** already exists in the database, inserting the row would violate the primary key constraint and result in the following error:

```
Msg 2627, Level 14, State 1, Line 1

Violation of PRIMARY KEY constraint 'PK_employee_id'. Cannot insert duplicate
key in object 'dbo.employees'.

The statement has been terminated.
```

While this error provides you with the information you need to troubleshoot the problem, there are two issues with it. First, the message is cryptic. It includes error codes, line numbers and other information unintelligible to the average user. Second, and more importantly, it causes the statement to abort and could cause an application crash.

The alternative is to wrap the statement in a **TRY...CATCH** statement, as shown below:

```
BEGIN TRY
INSERT INTO employees(id, first_name, last_name, extension)
VALUES(12497, 'Mike', 'Chapple', 4201)
END TRY
BEGIN CATCH
PRINT 'Error: ' + ERROR_MESSAGE();
EXEC msdb.dbo.sp_send_dbmail
@profile_name = 'Employee Mail',
@recipients = 'hr@foo.com',
@body = 'An error occurred creating a new employee record.',
@subject = 'Employee ID Duplication Error' ;
END CATCH
```

In this example, any errors that occur are reported to both the user executing the command and the **hr@foo.com** e-mail address. The error shown to the user appears below:

```
Error: Violation of PRIMARY KEY constraint 'PK_employee_id'. Cannot insert
duplicate key in object 'dbo.employees'.
Mail queued.
```

Most importantly, application execution continues normally, allowing the programmer to gracefully handle the error.
Use of the **TRY..CATCH** statement is an elegant way to proactively detect and handle errors that occur in **SQL Server** database applications.