Transact-SQL: Fehlerbehandlung

Stephan Karrer

Fehlermeldungen

Für jeden Datenbankmodul-Fehler existieren die folgenden Attribute:

- Fehlernummer:
  Jede Fehlermeldung besitzt eine eindeutige Fehlernummer (Nummern bis 50000 sind reserviert)
- Meldungstext:
  Zusätzliche Informationen zum Fehler
- Schweregrad:
  0-25: je höher, desto schwerwiegender
- Status:
  Statuscode für die Fehlerbedingung
- Name der Prozedur:
  Name der Prozedur oder des Triggers (sonst NULL)
- Zeilennummer:
  Zeilennummer der fehlerhaften Anweisung

Katalog der Fehlermeldungen

```
SELECT
    message_id,
    language_id,
    severity,
    is_event_logged,
    text
  FROM sys.messages
  WHERE language_id = 1033;
```

Die Katalogsicht ***sys.messages*** enthält eine Liste von allen system- und benutzerdefinierten Fehlermeldungen

Fehlerbehandlung mit @@ERROR

```
SET NOCOUNT OFF;
UPDATE Employees SET salary = -1
    WHERE Employee_ID = 100;
IF @@ERROR = 547
    PRINT 'A check constraint violation occurred.';
-- Nachfolgende Anweisungen werden ausgeführt
SELECT 2+2;
```

- liefert im Fehlerfall die Fehlernummer der vorherigen Anweisung, sonst 0
- wird durch jede Transact-SQL Anweisung neu gesetzt!
- war bis zu SQL Server 2000 der Standard für Fehlerbehandlung
- Ein Fehler führt nicht zum Abbruch des Stapels oder einer Prozedur!
  Nur die einzelne Anweisung scheitert.

Fehlerbehandlung mit TRY-CATCH (ab SQL Server 2005)

```
BEGIN TRY
    SELECT 1/0; -- Generate a divide-by-zero error
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber
        ,ERROR_SEVERITY() AS ErrorSeverity
        ,ERROR_STATE() AS ErrorState
        ,ERROR_PROCEDURE() AS ErrorProcedure
        ,ERROR_LINE() AS ErrorLine
        ,ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
GO
```

- Der TRY-Block kann mehrere Anweisungen enthalten
- Im CATCH-Block stehen zusätzliche Funktionen zur Fehleranalyse zur Verfügung
- Funktioniert nicht für schwerwiegende (>19) oder leichte Fehler (Warnungen, < 11)

Werfen von Fehlern mit RAISERROR (ab SQL Server 2005)

```
DECLARE @DBID INT = DB_ID();
DECLARE @DBNAME NVARCHAR(128) = DB_NAME();

RAISERROR
    ('DatabaseID is:%d, DatabaseName is:%s.',
    16, -- Severity.
    1, -- State.
    @DBID, -- First substitution argument.
    @DBNAME); -- Second substitution argument

SELECT 2+2;  -- wird ohne TRY CATCH ausgeführt
```

- Erlaubt die Rückgabe einer benutzerdefinierten Fehlermeldung
- Kann parametrisiert verwendet werden (ähnlich printf in der C-Programmierung)
- In dieser Form wird stets Fehlernummer 50000 verwendet

RAISERROR mit TRY CATCH

```
BEGIN TRY
    DECLARE @DBID INT = DB_ID();
    DECLARE @DBNAME NVARCHAR(128) = DB_NAME();

    RAISERROR
    ('DatabaseID is:%d, DatabaseName is:%s.',
    16, -- Severity.
    1, -- State.
    @DBID, -- First substitution argument.
    @DBNAME); -- Second substitution argument

    SELECT 2+2;  -- wird nicht mehr ausgeführt
END TRY
BEGIN CATCH
    -- gibt die Meldung von RAISERROR aus
    SELECT ERROR_MESSAGE() AS ErrorMessage;
END CATCH;
```

- Verarbeitung geht nach dem Catch-Block bzw. beim Aufrufer (innerhalb des gleichen Stapels) weiter

Definition eigener Fehlermeldungen

```
EXECUTE sp_addmessage 50005, -- Message id number
    10, -- Severity
    N'DatabaseID is: %d, DatabaseName is: %s.',
    @lang = 'us_english';
GO
DECLARE @DBID INT;
SET @DBID = DB_ID();
DECLARE @DBNAME NVARCHAR(128);
SET @DBNAME = DB_NAME();

RAISERROR (50005,
    10, -- Severity.
    1, -- State.
    @DBID, -- First substitution argument.
    @DBNAME); -- Second substitution argument.
GO
```

- Mittels der Prozedur *sp_addmessage* kann eine eigene Fehlermeldung definiert werden
- Diese kann anschließend via RAISERROR signalisiert werden

THROW ab Server 2012

```
BEGIN TRY
    DECLARE @FehlerText NVARCHAR(200) = DB_NAME();
    SET @FehlerText = 'Fehler DatabaseName is: '
                          + @FehlerText;
    THROW 55555, @FehlerText, 1;
    SELECT 2+2;  -- wird nicht mehr ausgeführt
END TRY
BEGIN CATCH
    SELECT
        ERROR_NUMBER() AS ErrorNumber
        ,ERROR_SEVERITY() AS ErrorSeverity
        ,ERROR_STATE() AS ErrorState
        ,ERROR_MESSAGE() AS ErrorMessage;
  --THROW  -- kann für Rethrow verwendet werden
END CATCH;
```

- THROW ist etwas einfacher als RAISERROR