11 | Error Handling and Transactions

Graeme Malcolm | Senior Content Developer, Microsoft
Geoff Allix | Principal Technologist, Content Master

Module Overview

- Errors and Error Messages
- Raising Errors
- Catching and Handling Errors
- Introduction to Transactions
- Implementing Explicit Transactions

Errors and Error Messages

Elements of Database Engine Errors	
Error number	Unique number identifying the specific error
Error message	Text describing the error
Severity	Numeric indication of seriousness from 1-25
State	Internal state code for the database engine condition
Procedure	The name of the stored procedure or trigger in which the error occurred
Line number	Which statement in the batch or procedure generated the error

In SQL Server (not Azure SQL Database):

- Error messages are in **sys.messages**
- You can add custom messages using **sp_addmessage**

Raising Errors

- The RAISERROR Command
  - Raise a user-defined error in **sys.messages** (SQL Server only)
  - Raise an explicit error message, severity, and state (SQL Server and Azure SQL Database)

```
RAISERROR ('An Error Occurred', 16, 0);
```

- The THROW Command
  - Replacement for RAISERROR
  - Throw explicit error number, message, and state (severity is 16)
  - Re-throw existing error

```
THROW 50001, 'An Error Occurred', 0;
```

DEMO

Raising Errors

Catching and Handling Errors

- Use a TRY…CATCH Block
- Handle errors in the CATCH block
  - Get error information:
    - @@ERROR
    - ERROR_NUMBER()
    - ERROR_MESSAGE()
    - ERROR_SEVERITY()
    - ERROR_STATE()
    - ERROR_PROCEDURE()
    - ERROR_LINE()
  - Execute custom correction or logging code
  - Re-throw the original error, or throw a custom error

```
DECLARE @Discount INT = 0;
BEGIN TRY
        UPDATE Production.Product
        SET Price = Price / @Discount
END TRY
BEGIN CATCH
        PRINT ERROR_MESSAGE();
        THROW 50001, 'An error occurred', 0;
END CATCH;
```

DEMO

Catching and Handling Errors

Introduction to Transactions

- A transaction is a group of tasks defining a unit of work
- The entire unit must succeed or fail together—no partial completion is permitted

```
--Two tasks that make up a unit of work
INSERT INTO Sales.Order ...
INSERT INTO Sales.OrderDetail ...
```

- Individual data modification statements are automatically treated as standalone transactions
- SQL Server uses locking mechanisms and the transaction log to support transactions

Implementing Explicit Transactions

- Use BEGIN TRANSACTION to start a transaction
- USE COMMIT TRANSACTION to complete a transaction
- USE ROLLBACK TRANSACTION to cancel a transaction
  - Or enable XACT_ABORT to automatically rollback on error
- Use @@TRANCOUNT and XACT_STATE() to check transaction status

```
BEGIN TRY
  BEGIN TRANSACTION
    INSERT INTO Sales.Order…
    INSERT INTO Sales.OrderDetail…
  COMMIT TRANSACTION
END TRY
BEGIN CATCH
  IF @@TRANCOUNT > 0
  BEGIN
    ROLLBACK TRANSACTION
  END
  PRINT ERROR_MESSAGE();
  THROW 50001, 'An error occurred', 0;
END CATCH;
```

DEMO

Implementing Transactions

Error Handling and Transactions

- Errors and Error Messages
- Raising Errors
- Catching and Handling Errors
- Introduction to Transactions
- Implementing Explicit Transactions

- Lab: Error Handling and Transactions

Microsoft