Module 8

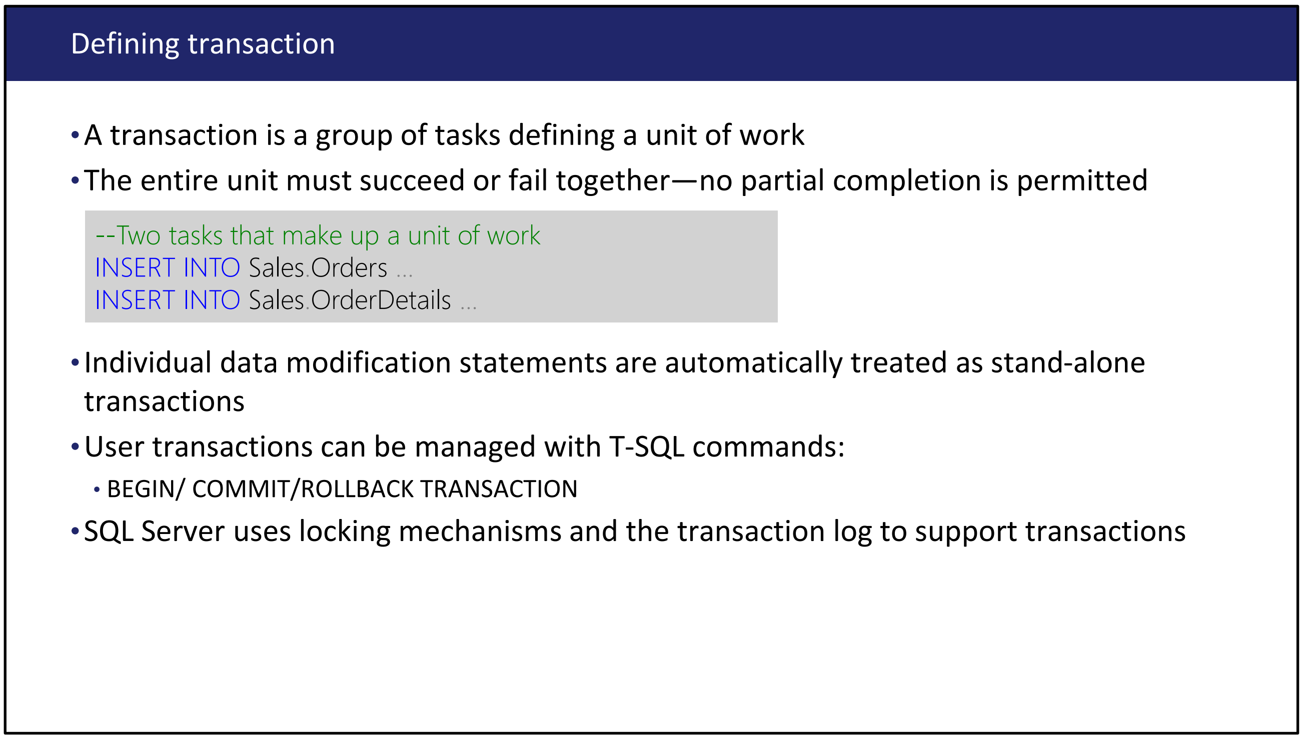
Protecting yourself using transactions

Module Overview

- Overview of transactions
- Controlling transactions

Lesson: Overview of transactions

- Defining transaction
- Understanding transactions
- SQL Server cannot read your mind



Defining transaction

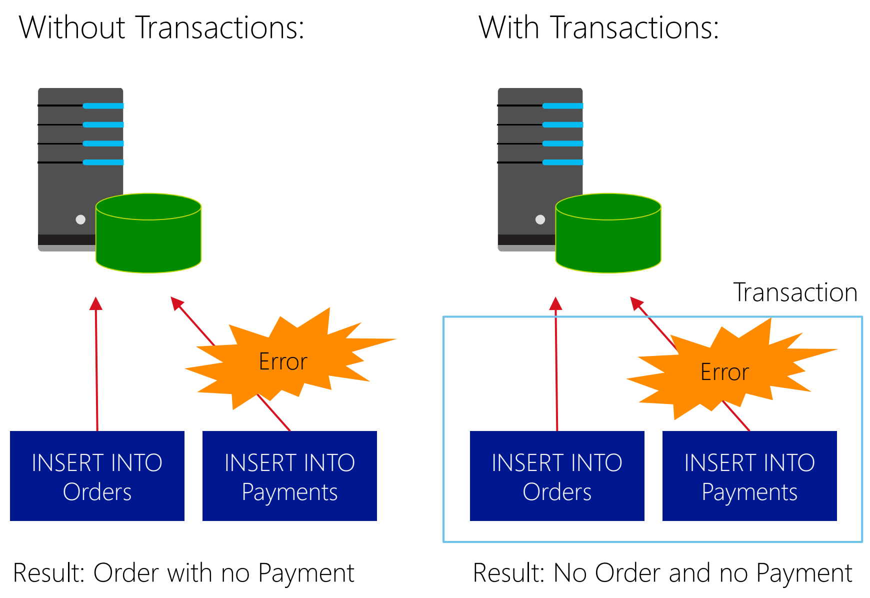
- A transaction is a group of tasks defining a unit of work
- The entire unit must succeed or fail together—no partial completion is permitted

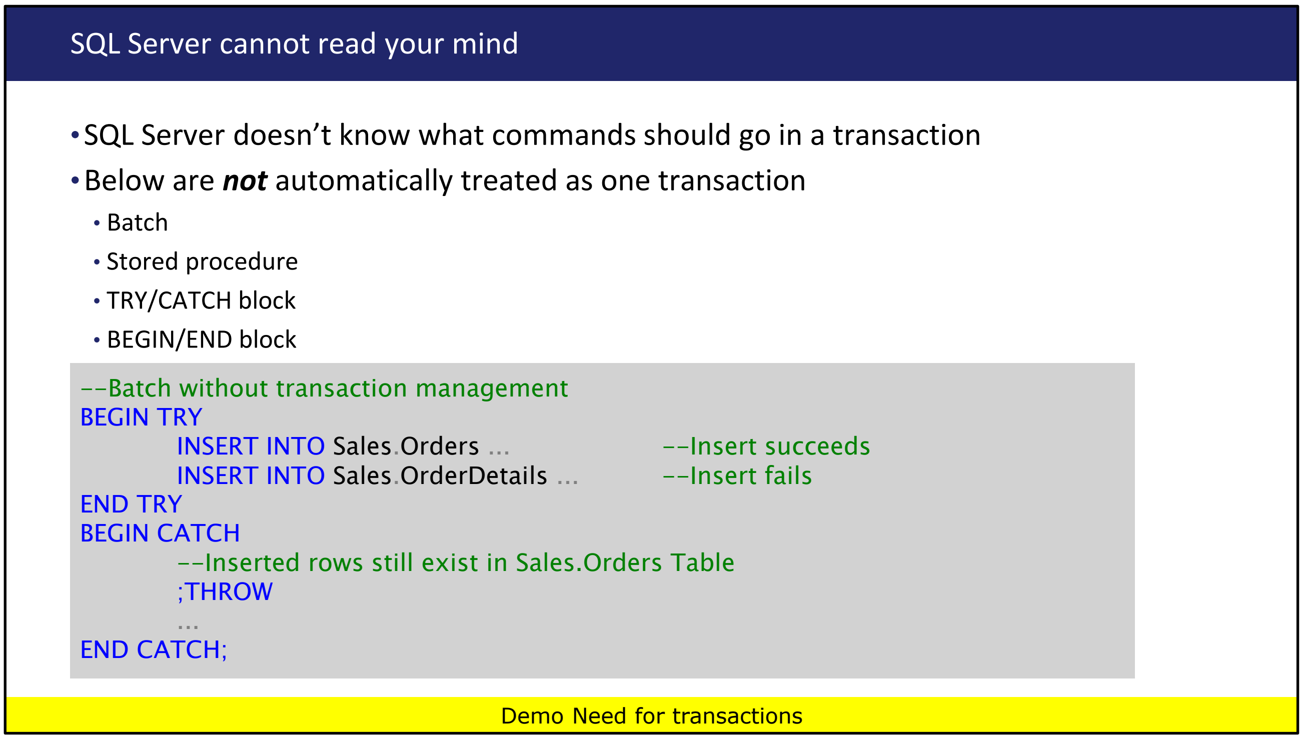
```
--Two tasks that make up a unit of work
INSERT INTO Sales.Orders ...
INSERT INTO Sales.OrderDetails ...
```

- Individual data modification statements are automatically treated as stand-alone transactions
- User transactions can be managed with T-SQL commands:
  - BEGIN/ COMMIT/ROLLBACK TRANSACTION
- SQL Server uses locking mechanisms and the transaction log to support transactions

Transactions in SQL Server for beginners
https://www.sqlshack.com/transactions-in-sql-server-for-beginners/

Understanding transactions





SQL Server cannot read your mind

- SQL Server doesn’t know what commands should go in a transaction
- Below are ***not*** automatically treated as one transaction
  - Batch
  - Stored procedure
  - TRY/CATCH block
  - BEGIN/END block

```
--Batch without transaction management
BEGIN TRY
          INSERT INTO Sales.Orders ...                --Insert succeeds
          INSERT INTO Sales.OrderDetails ...          --Insert fails
END TRY
BEGIN CATCH
          --Inserted rows still exist in Sales.Orders Table
          ;THROW
          ...
END CATCH;
```

Demo Need for transactions

Multiple Batches in a single Transaction
https://brianflove.com/2015-06-02/multiple-batches-in-a-single-transaction/

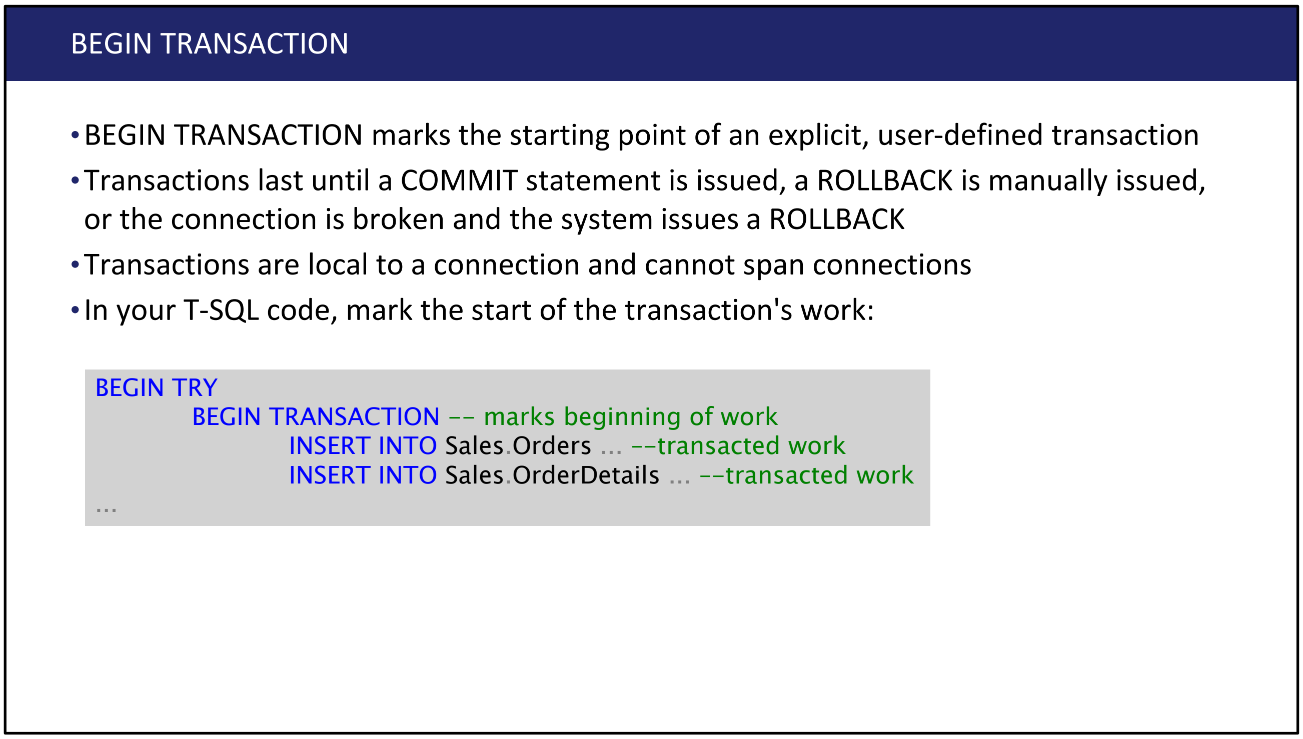
Lesson: Controlling transactions

- BEGIN TRANSACTION
- COMMIT TRANSACTION
- ROLLBACK TRANSACTION
- The Oops situation
- Using XACT_ABORT

Controlling transactions

- Transaction commands identify blocks of code that must succeed or fail together and provide points where the database engine can roll back, or undo, operations:

```
BEGIN TRY
        BEGIN TRANSACTION
                INSERT INTO Sales.Orders ...                --Insert succeeds
                INSERT INTO Sales.OrderDetails ...          --Insert fails
        COMMIT TRANSACTION                                  --If no errors, transaction completes
END TRY
BEGIN CATCH
     --Inserted rows still exist in Sales.Orders Table
    ;THROW
    ROLLBACK TRANSACTION –Undo the work done in the transaction!
END CATCH;
```



BEGIN TRANSACTION

- BEGIN TRANSACTION marks the starting point of an explicit, user-defined transaction
- Transactions last until a COMMIT statement is issued, a ROLLBACK is manually issued, or the connection is broken and the system issues a ROLLBACK
- Transactions are local to a connection and cannot span connections
- In your T-SQL code, mark the start of the transaction's work:

```
BEGIN TRY
        BEGIN TRANSACTION -- marks beginning of work
                INSERT INTO Sales.Orders ... --transacted work
                INSERT INTO Sales.OrderDetails ... --transacted work
...
```

BEGIN TRANSACTION (Transact-SQL)
https://docs.microsoft.com/en-us/sql/t-sql/language-elements/begin-transaction-transact-sql

What does BEGIN TRAN, ROLLBACK TRAN, and COMMIT TRAN mean?
https://www.mssqltips.com/sqlservertutorial/3305/what-does-begin-tran-rollback-tran-and-commit-tran-mean/

COMMIT TRANSACTION

- COMMIT ensures all of the transaction's modifications are made a permanent part of the database
- COMMIT frees resources, such as locks, used by the transaction
- In your T-SQL code, if a transaction is successful, commit it

```
BEGIN TRY
	BEGIN TRAN -- marks beginning of work
		INSERT INTO Sales.Orders ...
		INSERT INTO Sales.OrderDetails ...
	COMMIT TRAN -- mark the work as complete
END TRY
```

COMMIT TRANSACTION (Transact-SQL)
https://docs.microsoft.com/en-us/sql/t-sql/language-elements/commit-transaction-transact-sql

ROLLBACK TRANSACTION

- A ROLLBACK statement undoes all modifications made in the transaction by reverting the data to the state it was in at the beginning of the transaction
- ROLLBACK frees resources, such as locks, held by the transaction
- Before rolling back, you can test the state of the transaction with the XACT_STATE function
- In your T-SQL code, if an error occurs, ROLLBACK to the point of the BEGIN TRANSACTION statement

```
BEGIN CATCH
        SELECT ERROR_NUMBER() --sample error handling
        ROLLBACK TRAN
END CATCH;
```

ROLLBACK TRANSACTION (Transact-SQL)
https://docs.microsoft.com/en-us/sql/t-sql/language-elements/rollback-transaction-transact-sql

The Oops situation

- Imagine you by mistake deleted all rows in the Customers table
  - Talk to the DBA, which likely need to perform some RESTORE operation
- You can give yourself the ability to ROLLBACK
  - If you first begin a transaction
- Don't wait too long finishing the transaction
  - Locks are held until end of transaction

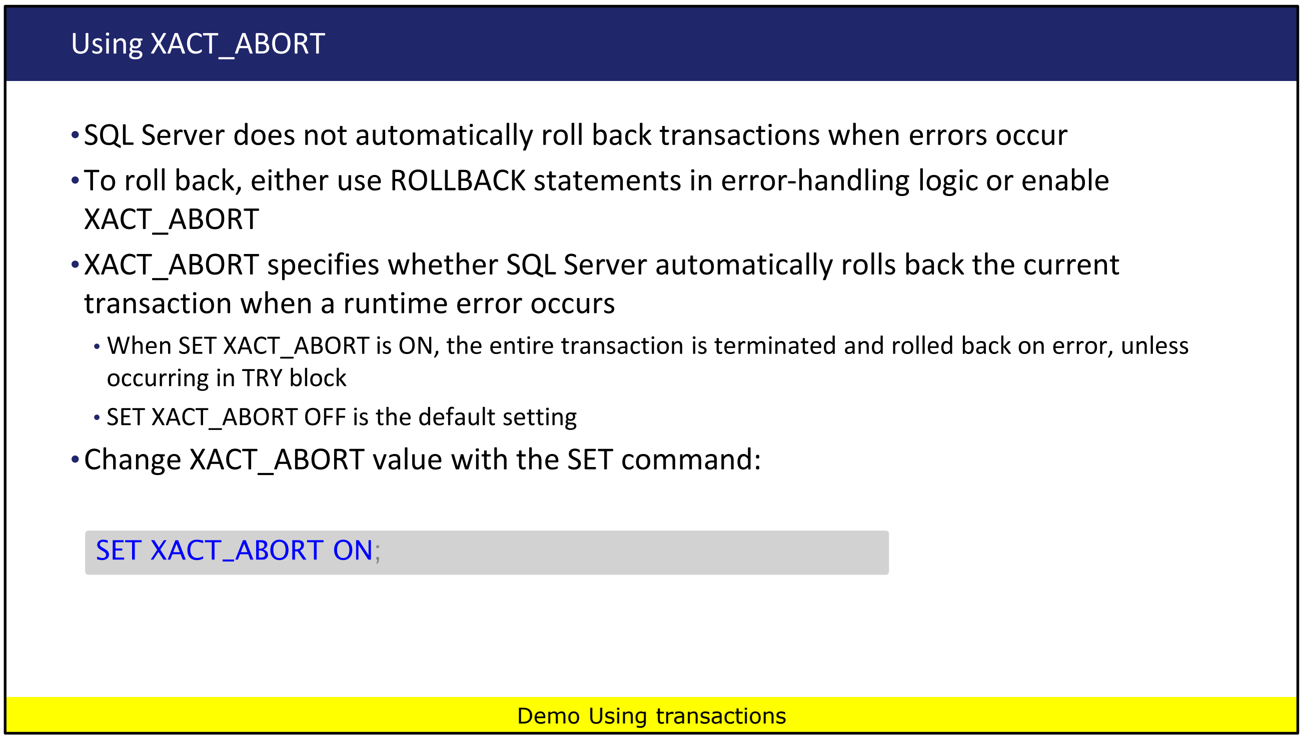
```
BEGIN TRAN

DELETE FROM SalesLT.Customer

--Oops! All customers are gone...

ROLLBACK
--Or COMMIT
```

Demo Oops



Using XACT_ABORT

- SQL Server does not automatically roll back transactions when errors occur
- To roll back, either use ROLLBACK statements in error-handling logic or enable XACT_ABORT
- XACT_ABORT specifies whether SQL Server automatically rolls back the current transaction when a runtime error occurs
  - When SET XACT_ABORT is ON, the entire transaction is terminated and rolled back on error, unless occurring in TRY block
  - SET XACT_ABORT OFF is the default setting
- Change XACT_ABORT value with the SET command:

```
SET XACT_ABORT ON;
```

Demo Using transactions

SET XACT_ABORT (Transact-SQL)
https://docs.microsoft.com/en-us/sql/t-sql/statements/set-xact-abort-transact-sql

Abort, Abort, We Are XACT_ABORT:ing, Or Are We?!
https://nielsberglund.com/2017/01/08/abort-abort-we-are-xact_aborting-or-are-we/

Error and Transaction Handling in SQL Server
https://www.sommarskog.se/error_handling/Part1.html

V1 Optional Lab 8: Implementing Transactions

- ***Note carefully the folder name in the lab instructions!!!***
- Exercise 1: Controlling Transactions with BEGIN, COMMIT, and ROLLBACK
- Exercise 2: Adding Error Handling to a CATCH Block

Estimated Time: 30 minutes

V2 Optional lab 8: Protecting yourself using transactions

- Ex 1. The Oops situation
- Ex 2. Adding transaction management to a stored procedure

Estimated Time: 30 minutes