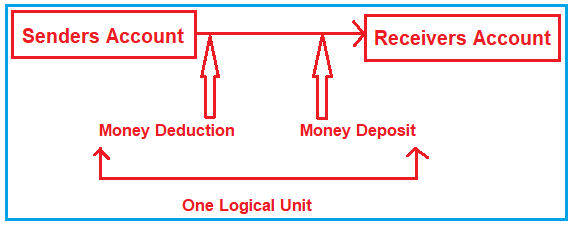
**What is a Transaction in SQL Server?**

A transaction is a set of SQL statements that should be executed as one unit. That means a transaction ensures that either all of the command succeeds or none of them. If one of the commands in the transaction fails, all of the commands fail and any data that is modified in the database is rolled back.

For example, if you are doing a back transaction i.e. transferring money from one account to another account. Then, in this case, there are two updates. One update is happening in the sender’s account from where the money is sending and the other update is happening in the receiver’s account who is receiving the money. Both of these two updates should either get committed or get rollback if there is an error. We don’t want the transaction to be in a half-committed state.



Let’s say you have a Customer table and an Address table. And you want if someone adds a customer and an address, then both of the inserts should pass or both of the inserts should fail, then in such types of scenarios, we need to use transactions in SQL Server.

**What is Transaction Management in SQL Server?**

The process of combining a set of related operations into a single unit and executing those operations by applying to do everything or do nothing principle is called transaction management. For example, the transfer money task is the combination of two operations

1. **Withdraw money from the Senders account**
2. **Deposit Money into the Receivers account.**

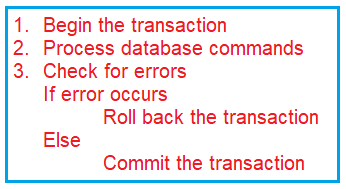
We need to execute these two operations by applying the “do everything or nothing principle” which is nothing but performing the transaction management. So, every transaction has two boundaries

1. **Beginning**
2. **Ending**

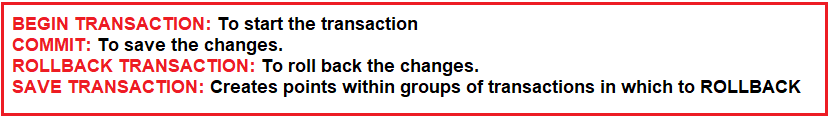
And controlling the boundaries of a transaction is nothing but transaction management.

**How to implement Transaction Management in SQL Server?**

In order to understand how to implement transaction management in SQL Server, please have a look at the following diagram which shows the Transaction processing steps. As you can see in the below image, transaction processing involves three steps. First, we need to begin the transactions. Then we need to write the DML operations which we want to execute as a single unit. In the third step, we need to check for errors. If there is any error i.e. any of the DML statements fails, then roll back the transaction (any data that is modified in the database will be rollback) else commit the transaction so that the data is saved permanently to the database.



To manage the transaction in SQL Server, we have provided transaction control language (TCL). TCL provides the following 4 commands which we can use to implement transactions in SQL Server.

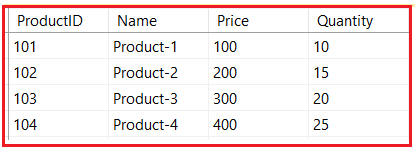


1. **Begin Transaction:** It indicates that the transaction is started.
2. **Commit Transaction:** It indicates that the transaction was completed successfully and all the data manipulation operations performed since the start of the transaction are committed to the database and frees the resources held by the transaction.
3. **Rollback Transaction:** It indicates that the transaction was Failed and will roll back the data to its previous state.
4. **Save Transaction:** This is used for dividing or breaking a transaction into multiple units so that the user has a chance of roll backing a transaction up to a point or location.

**Note:** Transactional Control Language commands are only used with the DML statements such as **INSERT, UPDATE, and DELETE** only. They cannot be used while creating tables or dropping them because these operations are automatically committed to the database. In the next article, we will discuss the different types of Transactions in SQL Server.

**Example to implement Transactions in SQL Server:**

Let us see some examples and understand the transactions in SQL Server. We are going to use the following Product table to understand transactions in SQL Server.



Please use the below SQL Script to create and populate the Product table with the required test data.

--Create Product table

**CREATE** **TABLE** Product

(

ProductID **INT** **PRIMARY KEY**,

Name **VARCHAR**(40),

Price **INT**,

Quantity **INT**

)

**GO**

-- Populate Product Table with test data

**INSERT** **INTO** Product **VALUES**(101, 'Product-1', 100, 10)

**INSERT** **INTO** Product **VALUES**(102, 'Product-2', 200, 15)

**INSERT** **INTO** Product **VALUES**(103, 'Product-3', 300, 20)

**INSERT** **INTO** Product **VALUES**(104, 'Product-4', 400, 25)

**Example of COMMIT transaction in SQL Server with DML statements**

The following is an example of a Commit Transaction.

**BEGIN** **TRANSACTION**

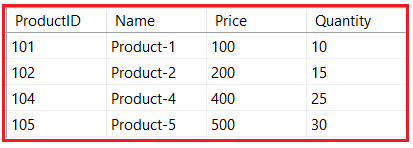
**INSERT** **INTO** Product **VALUES**(105,'Product-5',500, 30)

**UPDATE** Product **SET** Price =350 **WHERE** ProductID = 103

**DELETE** **FROM** Product **WHERE** ProductID = 103

**COMMIT** **TRANSACTION**

Once you commit the transaction, then you cannot Rollback the above three statements. After committing the transaction, now your Product will have the following data.



**What is the need for the ROLLBACK command in SQL Server?**

The Rollback command in SQL Server is used to undo the transactions that have not already been saved to the database and get back to the initial state from where the transaction was started. So, if you want to restore the data into its previous state then you need to use the ROLLBACK command at any time after the DML queries have been written but remember once the COMMIT has been written then we cannot ROLLBACK the data.

**Example of ROLLBACK TRANSACTION in SQL Server with DML statements**

Please execute the below statement to understand the Rollback command.

**BEGIN** **TRANSACTION**

**INSERT** **INTO** Product **VALUES**(106,'Product-6',600, 30)

**UPDATE** Product **SET** Price =550 **WHERE** ProductID = 105

**DELETE** **FROM** Product **WHERE** ProductID = 104

When you execute the above transaction and as you can see the transaction is not committed, then you have the chance to roll back all the operations (DML operations) on the table by using the Rollback Transaction command like below.

**ROLLBACK TRANSACTION**

**Understanding @@Error Global variable in SQL Server:**

This is a global variable and we can use this variable to check if there is any error or not. Let us see an example to understand this. As you can see in the below example, first we start the transaction using the **Begin Transaction** statement. Then we write two insert statements. Then we check if there is an error using the global system variable **@@ERROR**. A value greater than 0 means, there is some error. If there is some error then we roll back the transaction else we commit the transaction.

**BEGIN** **TRANSACTION**

**INSERT** **INTO** Product **VALUES**(110,'Product-10',600, 30)

**INSERT** **INTO** Product **VALUES**(110,'Product-10',600, 30)

**IF**(@@**ERROR** > 0)

**BEGIN**

Rollback Transaction

**END**

**ELSE**

**BEGIN**

Commit Transaction

**END**

Once you execute the above transaction, then you will see that the transaction is rollback. This is because we try to insert a duplicate value into the Primary key column.