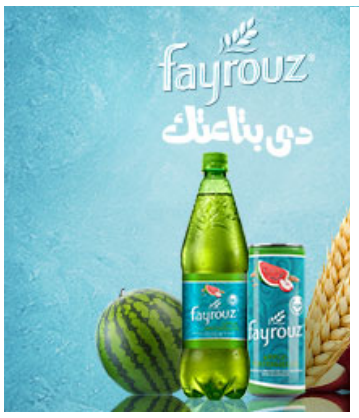
DOT NET TUTORIALS

Types of Transactions in SQL Server

Back to: SQL Server Tutorial For Beginners and Professionals



Types of Transactions in SQL Server with Examples

In this article, I am going to discuss the **Different Types of Transactions (Auto Commit, Implicit, and Explicit Mode of Transactions) in SQL Server** with Examples. Please read our previous article where we discussed the basics of **Transaction in SQL Server** before proceeding to this article. As part of this article, we are going to discuss the following pointers.

1. **Auto Commit Transaction Mode in SQL Server**
2. **Implicit Transaction Mode in SQL Server**
3. **Explicit Transaction Mode in SQL Server**

Types of Transactions in SQL Server:

The SQL Server transactions are classified into three types, they are as follows

1. **Auto Commit Transaction Mode (default)**
2. **Implicit Transaction Mode**
3. **Explicit Transaction Mode**

We are going to use the following Employee table to understand the Transactions in SQL Server. So, please use the below SQL Script to create the employee table.

```
CREATE TABLE Customer
(
    CustomerID INT PRIMARY KEY,
    CustomerCode VARCHAR(10),
    CustomerName VARCHAR(50)
)
```

Auto Commit Transaction Mode in SQL Server:

This is the default transaction mode in SQL Server. In this mode, each SQL statement is treated as a separate transaction. In this Transaction Mode, as a developer, we are not responsible for either beginning the transaction (i.e. Begin Transaction) or ending a transaction (i.e. either Commit or Roll Back). Whenever we perform or execute any DML statement, the SQL Server will automatically begin the transaction and end the transaction with a Commit or Rollback i.e. if the transaction completed successfully, it is committed. If the transaction faces any error failed, it is rolled back. So the programmer does not have any control over them.

Let us understand Auto Commit Transaction Mode with some examples. Please execute the below insert statement.

INSERT INTO Customer VALUES (1, ‘CODE_1’, ‘David’)

When you execute the above statement, the SQL Server will automatically begin the transaction and end the transaction with commit. Now, try to execute the below Insert query.

INSERT INTO Customer VALUES (1, ‘CODE_2’, ‘John’)

When you try to execute the above Insert Query, the insert failed as we are trying to insert a duplicate value into the primary key table, so the SQL Server will automatically begin the transaction and end the transaction with a Rollback. And when you execute the query you will get the below Primary Key Violation error.

In ...nsible for begin any DML statement and the developers are responsible to end the transaction with a commit or rollback. Once the transaction is ended, then automatically a new transaction will start by SQL Server. That means, in the case of implicit mode, a new transaction will start automatically by SQL Server after the current transaction is committed or rolled back by the programmer.

In order to use implicit transaction mode, first, you need to set the implicit transaction mode to ON using the **SET IMPLICIT_TRANSACTIONS** statement in SQL Server. The value for **IMPLICIT_TRANSACTIONS can be ON or OFF**. When the value for implicit mode is set to ON then a new transaction is automatically started by SQL Server whenever any SQL statement (Insert, Select, Delete, and Update) is executed.

Examples to understand Implicit Mode of Transactions in SQL Server:

Before going to do anything, first, DELETE all the data from the Customer table.

```
DELETE FROM Customer
```

Step1: Set the Implicit transaction mode to ON

```
SET IMPLICIT_TRANSACTIONS ON
```

Step2: Execute the DML Statement

Now let us try to insert two records using the implicit mode of transaction.

```
INSERT INTO Customer VALUES (1, 'CODE_1', 'David');
INSERT INTO Customer VALUES (2, 'CODE_2', 'John');
```


Step3: Commit the transaction

```
COMMIT TRANSACTION
```

When you execute the Commit Transaction statement, then data saved permanently into the database, and after that a new transaction will automatically be started by SQL Server.

Step4: Now execute the following DML Statements

```
INSERT INTO Customer VALUES (3, 'CODE_3', 'Pam');
UPDATE Customer SET CustomerName = 'John Changed' WHERE CustomerID = 2;
SELECT * FROM Customer;
```

When you execute the above statement and you will get the below data.

CustomerID	CustomerCode	CustomerName
1	CODE_1	David
2	CODE_2	John Changed
3	CODE_3	Pam

Step5: Rollback the transaction

As of now, we have not either committed or rollback the transaction, so let rollback the transaction and see the table data.

```
ROLLBACK TRANSACTION
```

Once you rollback the transaction, issue a select query against the customer table and you will see the following data.

CustomerID	CustomerCode	CustomerName
1	CODE_1	David
2	CODE_2	John

Note: If you don't want to use implicit transaction mode, then you can turn it off by executing the below statement.

SET IMPLICIT_TRANSACTIONS OFF

Explicit Transaction Mode in SQL Server:

In the Explicit mode of transaction, the programmer is only responsible for beginning the transaction as well as ending the transaction. In other words, we can say that the transactions that have a START and END explicitly written by the programmer are called explicit transactions.

Here, every transaction should start with a BEGIN TRANSACTION statement and ends with either a ROLLBACK TRANSACTION statement (when the transaction does not complete successfully) or a COMMIT TRANSACTION statement (when the transaction completes successfully). Explicit mode of transaction is most commonly used in triggers, stored procedures, and application programs.

Example using Stored Procedure:

Please create a simple procedure to add customers by executing the following script. As you can see, here, as a programmer we are beginning the transaction by using the **BEGIN TRANSACTION** statement. Then we are checking if there is an error using the global system **@@ERROR** variable and if we found any error then we rollback the transaction by executing the **ROLLBACK TRANSACTION** statement else we commit the transaction by executing the **COMMIT TRANSACTION** statement.

```
CREATE PROC SPAddCustommer
AS
BEGIN
 BEGIN TRANSACTION
  INSERT INTO Customer VALUES(3, 'CODE_3', 'Pam')
  INSERT INTO Customer VALUES(4, 'CODE_4', 'Sara')

        IF(@@ERROR > 0)
 BEGIN
     ROLLBACK TRANSACTION
 END
 ELSE
 BEGIN
     COMMIT TRANSACTION
 END
END
```

In the next article, I am going to discuss **Nested Transactions in SQL Server** with Examples. Here, in this article, I try to explain the **Different Types of Transactions (Auto Commit, Implicit, and Explicit Mode of Transactions) in SQL Server** with Examples. I hope you enjoy this Different Types of Transactions in SQL Server with Examples article and understand the difference between Auto, Implicit, and Explicit mode of transactions in SQL Server.

← Previous Lesson
Transaction Management in SQL Server

Next Lesson →
Nested Transactions in SQL Server

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

Website

Post Comment