ACID Properties in SQL Server

ACID Properties in SQL Server ensures Data Integrity during a transaction. The SQL ACID is an acronym for Atomicity, Consistency, Isolation, Durability.

In our previous article, we already explained about the Transaction and Nested Transactions. So, before these ACID Properties in SQL Server, I suggest you refer the same. In this article, Let me define every ACID property in SQL Server:

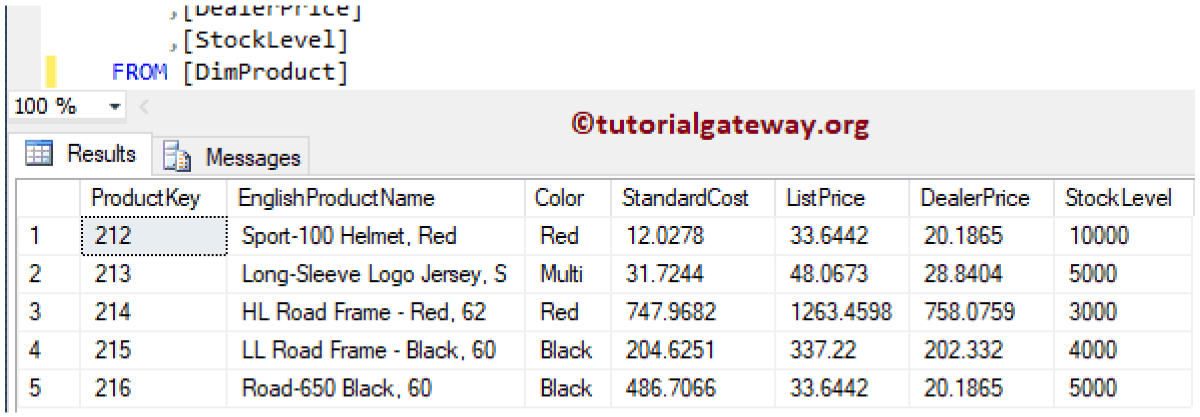
- **Atomicity:** The atomicity acid property in SQL. It means either all the operations (insert, update, delete) inside a transaction take place or none. Or you can say, all leave your database in a half-completed state.

- If the transaction completed successfully, then it will apply all the changes to the database.
- If there is an error in a transaction, then all the changes that already made will be rolled back automatically. It means the database will restore to its state that it had before the transaction started.
- If there is a system failure in the middle of the transaction, then also, all the changes made already will automatically rollback.

- **Isolation:** Every transaction is individual, and One transaction can't access the result of other transactions until the transaction completed. Or, you can't perform the same operation using multiple transactions at the same time. We will explain this SQL acid property in a separate article.
- **Durability:** Once the transaction completed, then the changes it has made to the database will be permanent. Even if there is a system failure, or any abnormal changes also, this SQL acid property will safeguard the committed data.

ACID Properties in SQL Server Example

We are going to use Dim products and Sales table to explain the Sql Server ACID properties. Below screenshot will show you the data inside DimProduct table



```
        ,[DealerPrice]
        ,[StockLevel]
  FROM [DimProduct]
```

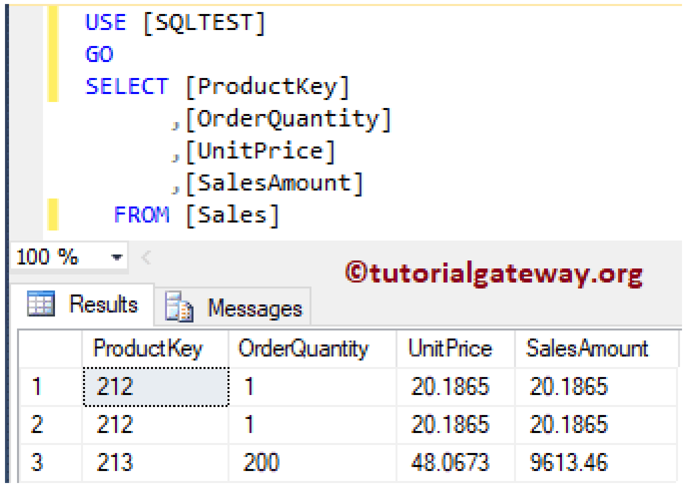
100 %

©tutorialgateway.org

Results | Messages

	ProductKey	EnglishProductName	Color	StandardCost	ListPrice	DealerPrice	StockLevel
1	212	Sport-100 Helmet, Red	Red	12.0278	33.6442	20.1865	10000
2	213	Long-Sleeve Logo Jersey, S	Multi	31.7244	48.0673	28.8404	5000
3	214	HL Road Frame - Red, 62	Red	747.9682	1263.4598	758.0759	3000
4	215	LL Road Frame - Black, 60	Black	204.6251	337.22	202.332	4000
5	216	Road-650 Black, 60	Black	486.7066	33.6442	20.1865	5000

and the data inside a sales table is:



```
USE [SQLTEST]
GO
SELECT [ProductKey]
      ,[OrderQuantity]
      ,[UnitPrice]
      ,[SalesAmount]
  FROM [Sales]
```

100 %

©tutorialgateway.org

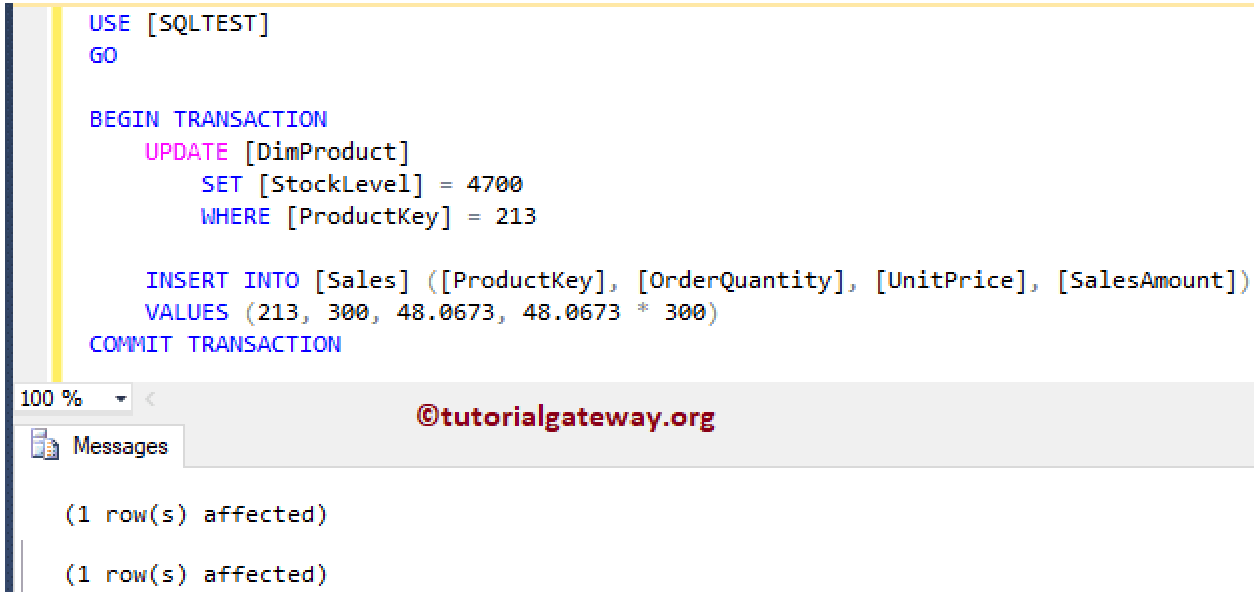
Results | Messages

	ProductKey	OrderQuantity	UnitPrice	SalesAmount
1	212	1	20.1865	20.1865
2	212	1	20.1865	20.1865
3	213	200	48.0673	9613.46

Atomicity in SQL ACID

It means all the statements inside a transaction should either succeed or fail as a unit. To demonstrate this SQL Atomicity Acid property, we are using the one UPDATE and an INSERT statement inside a transaction. Please refer to the Transaction and Nested Transactions articles.

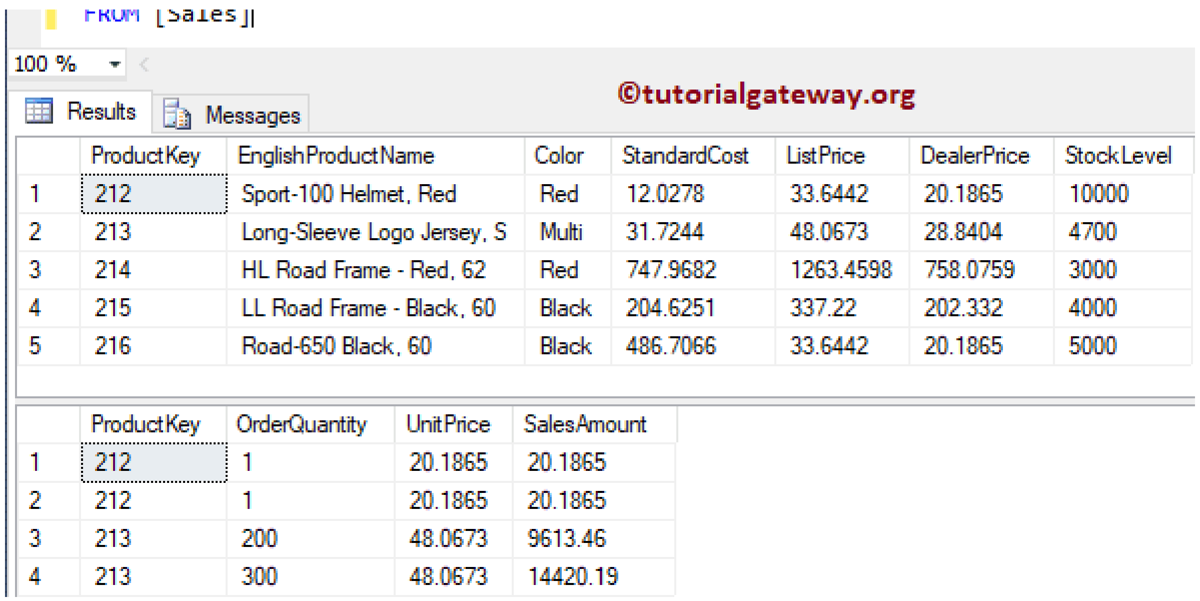
```
        INSERT INTO [Sales] ([ProductKey], [OrderQuantity], [UnitPrice], [SalesAmount])
        VALUES (213, 300, 48.0673, 48.0673 * 300)
COMMIT TRANSACTION
```



```
USE [SQLTEST]
GO

BEGIN TRANSACTION
    UPDATE [DimProduct]
        SET [StockLevel] = 4700
        WHERE [ProductKey] = 213

    INSERT INTO [Sales] ([ProductKey], [OrderQuantity], [UnitPrice], [SalesAmount])
    VALUES (213, 300, 48.0673, 48.0673 * 300)
COMMIT TRANSACTION
```

```
(1 row(s) affected)

(1 row(s) affected)
```

Let me show you the records in DimProduct, and Sales tables after that transaction.



```
FROM [Sales]
```

100 %

Results Messages

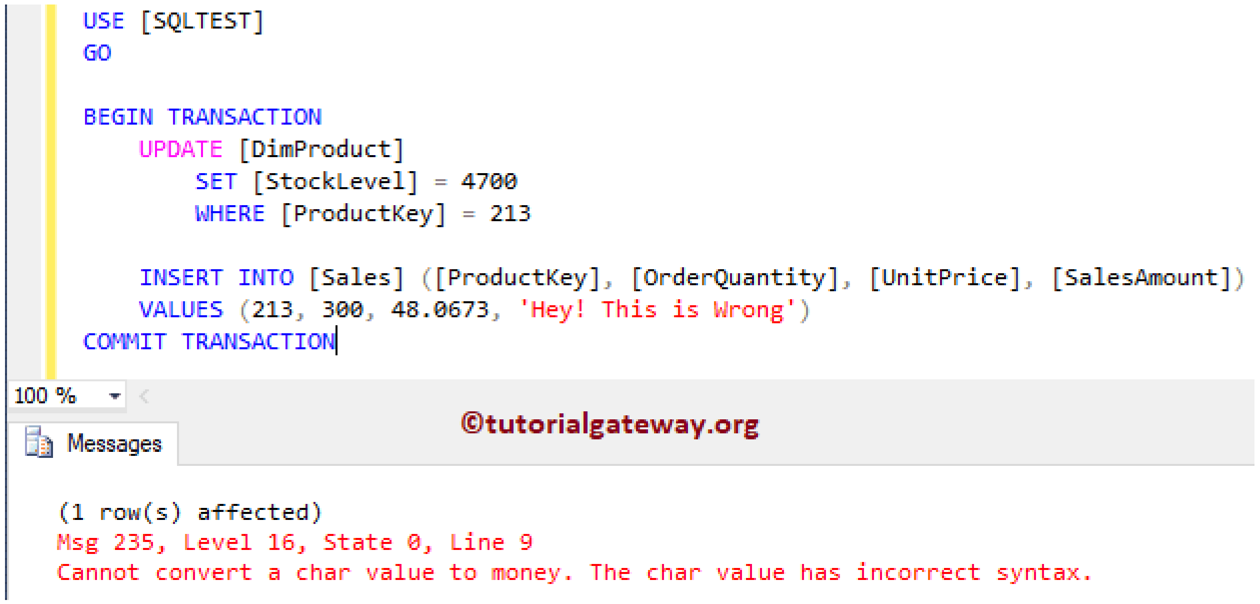
©tutorialgateway.org

	ProductKey	EnglishProductName	Color	StandardCost	ListPrice	DealerPrice	StockLevel
1	212	Sport-100 Helmet, Red	Red	12.0278	33.6442	20.1865	10000
2	213	Long-Sleeve Logo Jersey, S	Multi	31.7244	48.0673	28.8404	4700
3	214	HL Road Frame - Red, 62	Red	747.9682	1263.4598	758.0759	3000
4	215	LL Road Frame - Black, 60	Black	204.6251	337.22	202.332	4000
5	216	Road-650 Black, 60	Black	486.7066	33.6442	20.1865	5000

	ProductKey	OrderQuantity	UnitPrice	SalesAmount
1	212	1	20.1865	20.1865
2	212	1	20.1865	20.1865
3	213	200	48.0673	9613.46
4	213	300	48.0673	14420.19

This time we will insert wrong information in the Sales table to fail the insertion deliberately.

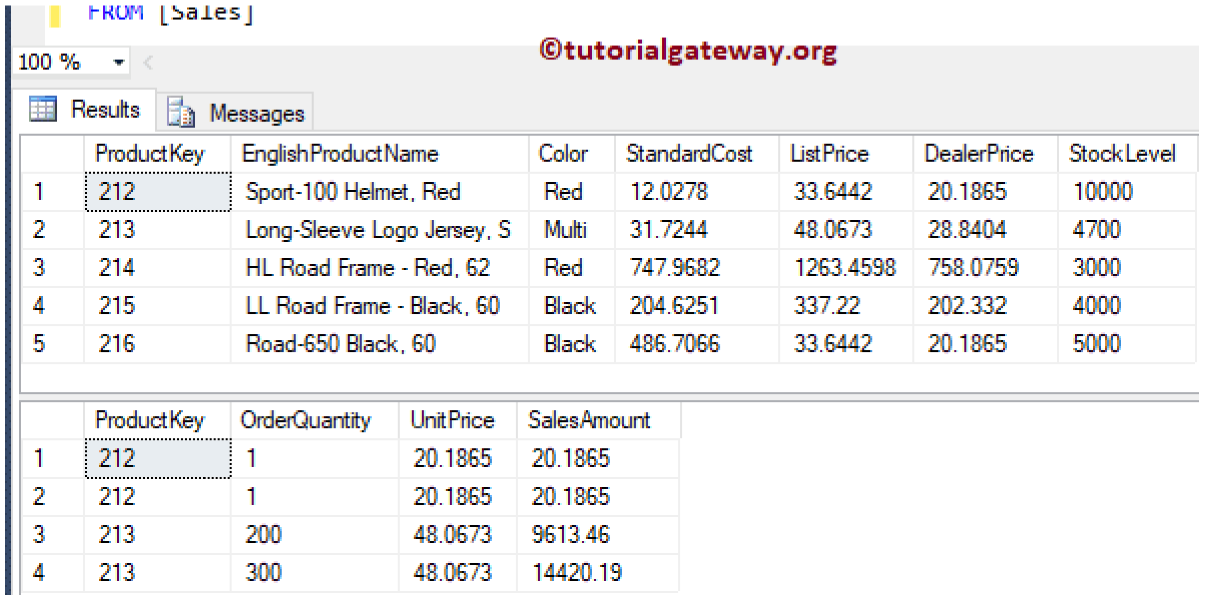
```
            WHERE [ProductKey] = 213

        INSERT INTO [Sales] ([ProductKey], [OrderQuantity], [UnitPrice], [SalesAmount])
        VALUES (213, 300, 48.0673, 'Hey! This is Wrong')
COMMIT TRANSACTION
```



```
USE [SQLTEST]
GO

BEGIN TRANSACTION
    UPDATE [DimProduct]
        SET [StockLevel] = 4700
        WHERE [ProductKey] = 213

    INSERT INTO [Sales] ([ProductKey], [OrderQuantity], [UnitPrice], [SalesAmount])
    VALUES (213, 300, 48.0673, 'Hey! This is Wrong')
COMMIT TRANSACTION
```

```
(1 row(s) affected)
Msg 235, Level 16, State 0, Line 9
Cannot convert a char value to money. The char value has incorrect syntax.
```

Let me show you the records in Dim Product and Sales tables after that transaction. As you can see from the above acid properties screenshot, a committed row (Update Statement) had rolled back.



```
FROM [Sales]
```

©tutorialgateway.org

	ProductKey	EnglishProductName	Color	StandardCost	ListPrice	DealerPrice	StockLevel
1	212	Sport-100 Helmet, Red	Red	12.0278	33.6442	20.1865	10000
2	213	Long-Sleeve Logo Jersey, S	Multi	31.7244	48.0673	28.8404	4700
3	214	HL Road Frame - Red, 62	Red	747.9682	1263.4598	758.0759	3000
4	215	LL Road Frame - Black, 60	Black	204.6251	337.22	202.332	4000
5	216	Road-650 Black, 60	Black	486.7066	33.6442	20.1865	5000

	ProductKey	OrderQuantity	UnitPrice	SalesAmount
1	212	1	20.1865	20.1865
2	212	1	20.1865	20.1865
3	213	200	48.0673	9613.46
4	213	300	48.0673	14420.19

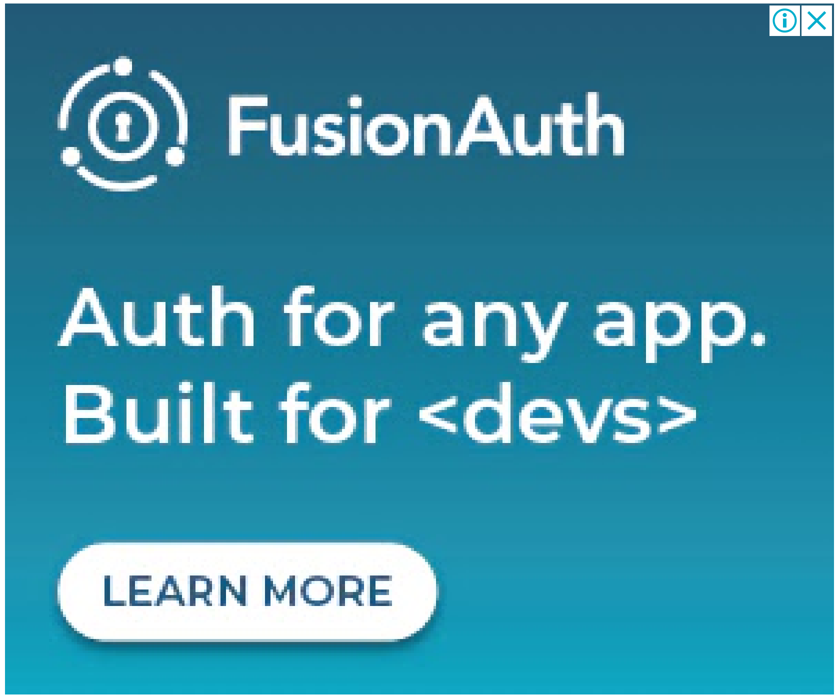
Consistency in SQL Server ACID

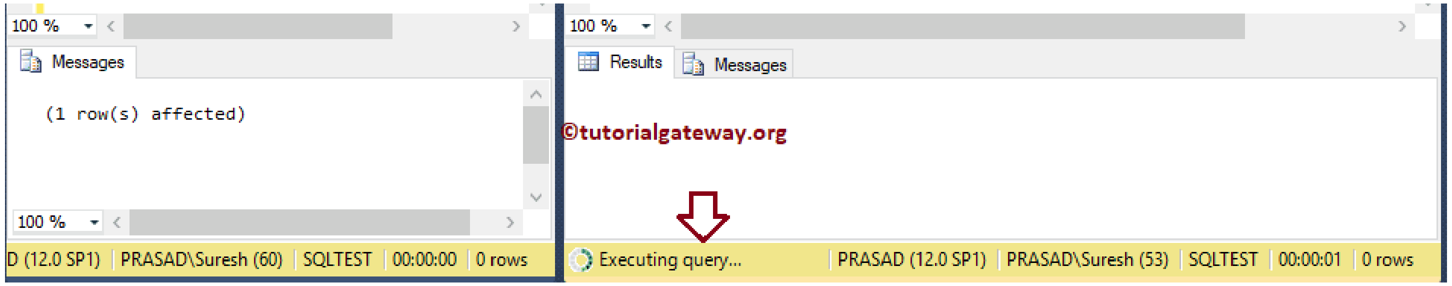
Let me take the above example to explain this SQL ACID property. Say, the transaction has updated the stock with new data, and suddenly there is a system failure (right before the insertion into sales or in the middle). In this situation, the system will roll back the updates. Otherwise, you can't trace the stock information.

Isolation in SQL Server ACID

- **First Instance:** we started the transaction and updating the record, but we haven't committed or rolled back the transaction.
- **Second Instance:** Using the Select statement to select the records present in the Dim Product table.

As you can see from the below acid properties screenshot, the select statement is not returning any information. Because we can't access one transaction result without completing the transaction.




```
(1 row(s) affected)
```
100 %
Messages
Results
©tutorialgateway.org
D (12.0 SP1) | PRASAD\Suresh (60) | SQLTEST | 00:00:00 | 0 rows
Executing query... | PRASAD (12.0 SP1) | PRASAD\Suresh (53) | SQLTEST | 00:00:01 | 0 rows

Let me execute the Rollback transaction. It will immediately show the result of the Select statement because the lock released from the Dim Product table.

Hope you understood the ACID Properties in SQL Server.

FusionAuth
Auth for any a
Built for <devs
LEARN MORE

خصم 30%
kaspersky
Kaspersky Internet Security
وقت للتأكد من أن
الخاصة تبقى خاصة
اشترِ الآن