**Difference between Truncate and Delete**

**By** [**Prashant Pandey**](http://www.sqlservercentral.com/Authors/Articles/Prashant__Pandey/487749/)**, 2011/07/04 (first published: 2007/11/22)**

Total article views: 24854 | Views in the last 30 days: 8988

http://www.sqlservercentral.com/Resources/Images/FilledStar.pnghttp://www.sqlservercentral.com/Resources/Images/FilledStar.pnghttp://www.sqlservercentral.com/Resources/Images/FilledStar.pnghttp://www.sqlservercentral.com/Resources/Images/FilledStar.pnghttp://www.sqlservercentral.com/Resources/Images/HalfFilledStar.png[Rate this](javascript:;) | http://www.sqlservercentral.com/Resources/Images/Discuss.gif  [Join the discussion](http://www.sqlservercentral.com/Forums/FindPost424812.aspx) | [[http://www.sqlservercentral.com/Resources/Images/Briefcase.gif](javascript:;) Briefcase](javascript:;) | http://www.sqlservercentral.com/Resources/Images/print.gif  [Print](http://www.sqlservercentral.com/articles/delete/61387/)

**3**diggsdigg

[Delicious](http://delicious.com/post?url=http://www.sqlservercentral.com/articles/delete/61387/&title=Difference+between+Truncate+and+Delete)

[Delicious](http://technorati.com/faves?add=http://www.sqlservercentral.com/articles/delete/61387/)

[in**Share**](javascript:void(0);)

**Introduction**

Hi to all, and here I am with a new article. The topic of this article is ***"The Difference between Truncate and Delete".***

*Truncate* and *Delete* both are used to delete data from the table. Both these commands will only delete the data of the specified table; they cannot remove the whole table: data along with structure. Now it's ok that both the SQL statements are used to delete only the data from the table but they both differ from each other in many aspects like syntax, performance, resource uses, etc., so let's take a look of both of these commands.

**Truncate**

The TRUNCATE command in SQL removes all rows from a table without logging the individual row deletions in the transaction log. The TRUNCATE statement has the sane functionality as the DELETE statement has in that it deletes the data from the table without modifying or deleting the structure of the table, however you can't use the WHERE Clause with the TRUNCATE statement.

The Syntax for this statement is:  
*TRUNCATE TABLE [ { database\_name.[ schema\_name ]. | schema\_name . } ] table\_name*

*Table\_name : Is the name of the table to truncate or from which all rows are removed.*  
  
To execute it, use a query like the one below:

TRUNCATE TABLE authors

The above command will delete all data from the table *author*.

**Delete**

The DELETE command in SQL also removes rows from a table, but it logs the individual row deletions in the transaction log. You can also use the WHERE Clause with the DELETE statement to qualify which rows are to be deleted.

Here I am showing just the simple syntax. For a more detailed explanation of the DELETE syntax, visit this link: [DELETE (Transact-SQL)](http://msdn2.microsoft.com/en-us/library/ms189835.aspx)

Syntax:

DELETE FROM TABLE\_NAME[ { database\_name.[ schema\_name ]. | schema\_name . } ] table\_name

*Database\_name:* Is the name of the database in which the table exists. This is optional. If it is not included, the current database context is assumed.

*Schema\_name:* Is the name of the schema in which the table exists. This is optional. If it is not included, the current database context is assumed.

*Table\_name :* Is the name of the table to truncate or from which all rows are removed.  
  
A simple command looks like this query:

DELETE FROM authors

The above query will delete all data from the table author.

In DELETE statements you can limit your DELETE query using the WHERE clause to delete only particular records that fulfill the condition of the WHERE clause. In this case, only those records matching the WHERE clause will be deleted, not the all records. A limited DELETE query is shown below.

DELETE FROM authors Where AuthorId IN (1,2,3)

This statement only deletes rows from the authors table which have the author ids of 1, 2, or 3.

**The Differences between Truncate and Delete**

Now above you have seen the DELETE and TRUNCATE statements. Both the statements are similar, but there are many differences that exist between them. Those similarities and differences are explaned below:

**TRUNCATE and DELETE remove the data not the structure**  
Both commands remove rows from a table, but the table structure and its columns, constraints, indexes, and so on remain. To remove the table definition in addition to its data, use the DROP TABLE statement.

**Conditional based deletion of data**  
Conditional based deletion of data means that not all rows are deleted. Let's suppose I have a table *authors* and from this table I want to delete the authors that are living in Australia. Let's examine what our options for doing this with each command.

* TRUNCATE - In case of the TRUNCATE command we can't perform the conditional based deletion because there is no WHERE clause allowed with this command.
* DELETE - THe DELETE command provides the functionality of conditional based deletion of data from the table using the WHERE clause.

**Delete and Truncate both are logged operations:**

On most of the articles I have read on the Internet, I have seen this written: "delete is a logged operation and truncate is not a logged operation", which means when we run the delete command it logs (records) the information about the deleted rows and when we run the truncate command it doesn't log any data. But this is not true; truncate is also a logged operation but in a different way. It uses fewer system and transaction log resources than delete. The TRUNCATE command uses minimum logging resources, which is why it is faster than delete. So both delete and truncate are logged operations, but they work differently as shown below.

* ***DELETE is a logged operation on a per row basis.*** The DELETE statement removes rows one at a time and records an entry in the transaction log for each deleted row. So, in case if you are deleting a huge number of records then it can cause your transaction log to grow. This means the deletion of a huge number of records will use more server resources as it logs each and every row that is deleted. That is why your transaction log will grow very rapidly. Since the delete statement records each deleted row it is also slow. Some people ask that if this is done for each row then why does not Microsoft modify the delete statement to not record each deleted row*???* The answer is when you run your databases in full recovery mode, detailed logging is necessary for SQL Server to be able to recover your database to the most recent state.
* ***TRUNCATE logs the deallocation of the data pages in which the data exists.*** TRUNCATE is faster than DELETE due to the way TRUNCATE "removes" rows from the table. It won't log the deletion of each row; instead it logs the deallocation of the data pages of the table. The TRUNCATE statement removes the data by deallocating the data pages used to store the table data and records only the page deallocation in the transaction log. *Actually, TRUNCATE does not remove data, but rather deallocates whole data pages and removes pointers to indexes. The data still exists until it is overwritten or the database is shrunk.* This action does not require a lot of resources and is therefore very fast. It is a common mistake to think that TRUNCATE is not logged. This is wrong. The deallocation of the data pages is recorded in the log file. Therefore, "Books Online (BOL)" refers to TRUNCATE operations as "minimally logged" operations. You can use TRUNCATE within a transaction, and when this transaction is rolled-back, the data pages are reallocated again and the database is again in its original, consistent state.

To be more specific lets take a look of an example, which will tell you that truncate is also a logged operation. So take a look of following example.

/\*Create a dummy table in a non-production(dummy) database.\*/

CREATE TABLE tranTest

(

Id int Identity(1,1),

Name Varchar(100)

)

/\*Now insert the records in the tranTest table.\*/

INSERT INTO tranTest(Name) VALUES('prashant')

INSERT INTO tranTest(Name) VALUES('prateek')

INSERT INTO tranTest(Name) VALUES('praveen')

INSERT INTO tranTest(Name) VALUES('prakash')

INSERT INTO tranTest(Name) VALUES('prabhat')

/\*Then as you know delete is a looged operation that means

in a transaction if we rollback the transaction after deleting

the records from table, it will restore all deleted records.\*/

BEGIN TRAN

DELETE FROM tranTest

SELECT \* FROM tranTest

ROLLBACK

SELECT \* FROM tranTest

/\*Now delete all records from the table.\*/

DELETE FROM tranTest

/\*And Insert new fresh records in the table.\*/

INSERT INTO tranTest(Name) VALUES('prashant')

INSERT INTO tranTest(Name) VALUES('prateek')

INSERT INTO tranTest(Name) VALUES('praveen')

INSERT INTO tranTest(Name) VALUES('prakash')

INSERT INTO tranTest(Name) VALUES('prabhat')

/\*Then as you now you know that Truncate is also a logged

opertion so it must restore all records that are deleted from

the tranTest table in the below tansaction\*/

BEGIN TRAN

TRUNCATE TABLE tranTest

SELECT \* FROM tranTest

ROLLBACK

SELECT \* FROM tranTest

So when we run the above command and TRUNCATE also restores all the records that means somewhere the logging of the TRUNCATE operation is also being done, but with minimal resources. Here I think minimal resources means you can restore the Truncate statements for that particular session but if you close the connection then i think it will not able to restore your data. BUT Delete will be able to restore your data later also.

**Behavior of Delete and Truncate for identity columns**  
OK, now the case of identity columns. Both the TRUNCATE and DELETE commands behave differently against Identity columns. When we use truncate it will reset the counter used by an identity column for new rows to the seed value defined for the column. But in the case of DELETE it will not reset the counter of your identity column. Rather it maintains the same counter for new rows. In both the cases, if no seed was defined the default value 1 is used. As TRUNCATE resets the identity column counter, in the case where you want to retain the identity counter, use DELETE instead of TRUNCATE.

Why do these two behave differently? I don't know, but T-SQL is providing you the two ways you can use as needed. In the case where you want all the data from the table deleted and the counter will restarting from 1, then truncate can help you. If you want to delete all the records but don't want to reset you counter, then delete is there for you.

An example for this is here. :

/\* First if you already created the tranTest table then delete it from your database.\*/

DROP Table tranTest

/\* Then Create a dummy table in a non-production(dummy) database with an Identity column

\*/

CREATE TABLE tranTest

(

Id int Identity(1,1),

Name Varchar(100)

)

/\* Now insert the records in the tranTest table. \*/

INSERT INTO tranTest(Name) VALUES('prashant')

INSERT INTO tranTest(Name) VALUES('prateek')

INSERT INTO tranTest(Name) VALUES('praveen')

INSERT INTO tranTest(Name) VALUES('prakash')

INSERT INTO tranTest(Name) VALUES('prabhat')

/\* If you run the below querry you'll see that the

max value of the identity column is 5 caz we have

inserted only five records in this table. \*/

SELECT \* FROM tranTest

/\* Now delete all the records from the table using the delete command. \*/

DELETE FROM tranTest

/\* Now by running the above command your table is empty,

so insert the new 5 records in the table to see that after

using the delete command from where identity will tart, fom 1 or from 6. \*/

INSERT INTO tranTest(Name) VALUES('prashant')

INSERT INTO tranTest(Name) VALUES('prateek')

INSERT INTO tranTest(Name) VALUES('praveen')

INSERT INTO tranTest(Name) VALUES('prakash')

INSERT INTO tranTest(Name) VALUES('prabhat')

/\* After running the below query u'll see that your identity now

starts from 6 because delete do not resets the counter of your

identity column \*/

SELECT \* FROM tranTest

/\* Now drop your table again and create it again. \*/

DROP TABLE tranTest

/\* Create a dummy table again in a non-production(dummy) database

to see the effect of truncate command on identity columns \*/

CREATE TABLE tranTest

(

Id int Identity(1,1),

Name Varchar(100)

)

/\* And Insert new fresh records in teh table. \*/

INSERT INTO tranTest(Name) VALUES('prashant')

INSERT INTO tranTest(Name) VALUES('prateek')

INSERT INTO tranTest(Name) VALUES('praveen')

INSERT INTO tranTest(Name) VALUES('prakash')

INSERT INTO tranTest(Name) VALUES('prabhat')

/\* Now at this point the counter of this table's

identity column is 5 taht is the max value of id column \*/

SELECT \* FROM tranTest

/\* Then truncate the table. \*/

TRUNCATE TABLE tranTest

/\* and insert new records \*/

INSERT INTO tranTest(Name) VALUES('prashant')

INSERT INTO tranTest(Name) VALUES('prateek')

INSERT INTO tranTest(Name) VALUES('praveen')

INSERT INTO tranTest(Name) VALUES('prakash')

INSERT INTO tranTest(Name) VALUES('prabhat')

/\* Now you'll see that after truncating the table the

identity is reset to its seed value. \*/

SELECT \* FROM tranTest

/\* So this example explains the beahviour of both of these command for Identity columns. \*/

**TRUNCATE is a DDL command whereas DELETE is a DML command**  
This is also a common difference you might have read in many articles. That is TRUNCATE is a DDL (data definition language) operation and DELETE is a DML (data manipulation language) operation. Yes according to SQL server it's true. But why it is so, why is TRUNCATE DDL and DELETE DML? Let's look at this;

When we run the TRUNCATE command it puts a "Schema modification (Sch-M)" lock on the table. What is "*schema modification (Sch-M)"?*

The Database Engine uses schema modification (Sch-M) locks during a table data definition language (DDL) operation, such as adding a column or dropping a table. During the time that it is held, the Sch-M lock prevents concurrent access to the table. This means the Sch-M lock blocks all outside operations until the lock is released. Now you'll ask how it blocks any modification to the table when in the case of TRUNCATE we are performing modifications because we are deleting data? But deleting the data is the one side of coin only. What we see with the internal workings of truncate is because as you read above, that it doesn't remove the data. Rather it deallocates the data pages. *Because TRUNCATE doesn't perform any data modification in the table that is why the DELETE TRIGGER is not called.* I think we are not modifying the data of the table, BUT as you know TRUNCATE resets the Identity counter of the column in the table, which means the TRUNCATE is modifying the table definition or structure, which comes under the DDL operations. Also when you are truncating a table, you can't modify or add any data to the table. So, to become a DDL operation you have to fulfill some of the conditions written below:

* Modifying a table structure or definition comes under DDL operations, and
* When you are modifying the table structure, you can't access the table to do any data modification.

Since TRUNCATE is doing all the activities above, that proves that TRUNCATE is a DDL operation.

Now we move to the DELETE command. In case of the DELETE command I am not sure which lock is implemented, but as we know and you can read above that DELETE command deletes the rows one by one. It is modifying the data by deleting it from the table, and because DELETE performs data modifications that is why the DELETE TRIGGER is called. The DELETE command does not modify the table structure in any manner, such as like how TRUNCATE modifies the identity column by resetting its value.

To become a DML operation you have to fulfill some of the conditions written below:

* Modifying the table data.
* When you are modifying the table data in the mean time you can't perform any table structure modification on the table.

Here the DELETE command is modifying the data of the table and also when delete statement is running you can't modify the table structure. So we can say that DELETE is a DML operation.

**Behavior of Truncate and Delete for Triggers**  
Triggers are important topic in SQL Server, and here I am talking about how both TRUNCATE and DELETE behave differently for Triggers. As you all know triggers fire whenever any data modification happens in the table. In case of TRUNCATE and DELETE, because they are deleting data from the table, the DELETE TRIGGER will fire if present and also the INSTEAD OF and AFTER triggers can be triggered if present. The INSERT and UPDATE triggers will not be fired here.

So let's take both one by one:

* TRUNCATE - When we run the TRUNCATE command to delete all the rows of a table it actually doesn't remove any row rather it deallocates the data pages. So in the case of the TRUNCATE command, triggers will not be fired because here no modification takes place. As we know that TRUNCATE is a DDL command, and DDL commands doesn't modify your data; instead they modify your table structure and definition.
* DELETE - In case of DELETE the DELETE trigger will be fired if present and also if the INSTEAD OF and AFTER triggers for DELETE TRIGGER is present then they also will be fired. As we know that delete command is a DML command and it deletes the data on row-by-row basis. So that means delete is modifying the data by deleting it from the table. As we know delete is a DML command and trigger will be fired whenever any DML operation on the table takes place.

**Where we can use these statements**  
There are some restrictions on the use of both of these statements as follows:

*For Delete*

* The DELETE statement may fail if it violates a trigger or tries to remove a row referenced by data in another table with a FOREIGN KEY constraint. If the DELETE removes multiple rows, and any one of the removed rows violates a trigger or constraint, the statement is canceled, an error is returned, and no rows are removed.

*For Truncate*  
You cannot use TRUNCATE TABLE on tables that:

* Are referenced by a FOREIGN KEY constraint.
* Participate in an indexed view.
* Are published using transactional replication or merge replication.

**Permissions of performing TRUNCATE or DELETE operation**  
For using both the statements you need some permissions on the server, which decides whether you can perform the Delete or Truncate action on the table or not. To truncate a table, you need at least ALTER permissions on the table (which is granted by default to the table owner, members of the sysadmin fixed server role, and the db\_owner and db\_ddladmin fixed database roles). To delete rows using the delete statement, you only need the DELETE permission.

*The above is the detailed explanation of the differences between both of these statements. Please feel free to comment or ask questions about this article.*