How to delete all rows from all tables in a SQL Server database?

Asked 9 years, 9 months ago Active 11 months ago Viewed 162k times

147

How to delete all rows from all tables in a SQL Server database?

sql-server-2005

58

edited May 31 '12 at 10:32
Michał Powaga
18.2k 6 42 59

asked Dec 14 '09 at 9:19
surajit khamrai

See codeguru.com/forum/showthread.php?t=458182 and scroll down... – Wim ten Brink Dec 14 '09 at 9:24

4 by drop database will be deleted i just want to reset data – surajit khamrai Dec 14 '09 at 9:27

11 Answers

251

Note that TRUNCATE won't work if you have any referential integrity set.

In that case, this will work:

```
EXEC sp_MSForEachTable 'DISABLE TRIGGER ALL ON ?'
GO
EXEC sp_MSForEachTable 'ALTER TABLE ? NOCHECK CONSTRAINT ALL'
GO
EXEC sp_MSForEachTable 'DELETE FROM ?'
GO
EXEC sp_MSForEachTable 'ALTER TABLE ? CHECK CONSTRAINT ALL'
GO
```

Mark Rendle
7,561 1 26 51

1 nice. good thinking.... – Preet Sangha Dec 14 '09 at 9:28

1 and what about triggers ????????? – surajit khamrai Dec 14 '09 at 9:30

1 Actually, that's only for DDL triggers. In which case: EXECP sp_MSForEachTable 'DISABLE TRIGGER ALL ON ?' – Mark Rendle Dec 14 '09 at 9:34

9 Not available in SQL Azure :(– Akash Kava Apr 17 '13 at 12:31

1 Got it - if a backup file already exists then it looks like SSMS appends to it rather than replacing it (I didn't realize this). So I deleted the file and now the 'empty' database backup file is only 3.7 MB – Ben Dec 8 '14 at 15:34

19

In my recent project my task was to clean an entire database by using sql statement and each table having many constraints like Primary Key and Foreign Key. There are more than 1000 tables in database so its not possible to write a delete query on each and ever table.

By using a stored procedure named sp_MSForEachTable which allows us to easily process some code against each and every table in a single database. It means that it is used to process a single T-SQL command or a different T-SQL commands against every table in the database.

So follow the below steps to truncate all tables in a SQL Server Database:

Step 1- Disable all constraints on the database by using below sql query :

```
EXEC sys.sp_msforeachtable 'ALTER TABLE ? NOCHECK CONSTRAINT ALL'
```

Step 2- Execute a Delete or truncate operation on each table of the database by using below sql command :

```
EXEC sys.sp_msforeachtable 'DELETE FROM ?'
```

Step 3- Enable all constraints on the database by using below sql statement:

4,533 5 33 66

401 4 4

1 You can simply execute step 2 mutliple times so that first time it deletes tables with non dependencies, 2nd time to delete those tables failed in first time, 3rd time to delete faild in 2nd time, etc – user586399 Jun 3 '16 at 22:10

any ideas on how to do this on `sql server azure` ? – Zapnologica Jan 30 '18 at 8:36

This approach will work also in Azure as it uses only plain SQL: sqlrelease.com/delete-all-rows-from-all-tables – Jakob Lithner Nov 19 '18 at 12:37

14

I had to delete all the rows and did it with the next script:

```
DECLARE @Nombre NVARCHAR(MAX);
DECLARE curso CURSOR FAST_FORWARD
FOR
Select Object_name(object_id) AS Nombre from sys.objects where type = 'U'

OPEN curso
FETCH NEXT FROM curso INTO @Nombre

WHILE (@@FETCH_STATUS <> -1)
BEGIN
IF (@@FETCH_STATUS <> -2)
BEGIN
DECLARE @statement NVARCHAR(200);
SET @statement = 'DELETE FROM ' + @Nombre;
print @statement
execute sp_executesql @statement;
END
FETCH NEXT FROM curso INTO @Nombre
END
CLOSE curso
DEALLOCATE curso
```

Hope this helps!

answered Oct 3 '12 at 17:27

9

Here is a solution that:

1. Drops constraints (thanks to this post)
2. Iterates through `INFORMATION_SCHEMA.TABLES` for a particular database
3. `SELECTS` tables based on some search criteria
4. Deletes all the data from those tables
5. Re-adds constraints
6. Allows for ignoring of certain tables such as `sysdiagrams` and `__RefactorLog`

I initially tried `EXECUTE sp_MSforeachtable 'TRUNCATE TABLE ?'` , but that deleted my diagrams.

```
USE <DB name>;
GO

-- Disable all constraints in the database
EXEC sp_msforeachtable "ALTER TABLE ? NOCHECK CONSTRAINT all"

declare @catalog nvarchar(250);
declare @schema nvarchar(250);
declare @tbl nvarchar(250);
DECLARE i CURSOR LOCAL FAST_FORWARD FOR select
                                        TABLE_CATALOG,
                                        TABLE_SCHEMA,
                                        TABLE_NAME
                                        from INFORMATION_SCHEMA.TABLES
                                        where
                                        TABLE_TYPE = 'BASE TABLE'
                                        AND TABLE_NAME != 'sysdiagrams'
                                        AND TABLE_NAME != '__RefactorLog'

OPEN i;
FETCH NEXT FROM i INTO @catalog, @schema, @tbl;
WHILE @@FETCH_STATUS = 0
    BEGIN
        DECLARE @sql NVARCHAR(MAX) = N'DELETE FROM [' + @catalog + '].[' + @schema + '].
```

```
CLOSE i;
DEALLOCATE i;

-- Re-enable all constraints again
EXEC sp_msforeachtable "ALTER TABLE ? WITH CHECK CHECK CONSTRAINT all"
```

edited May 3 '18 at 10:35

answered Mar 1 '17 at 9:17

Zach Smith
2,921 5 29 61

This is great but it doesn't take non-dbo schemas into account. – influent Jan 19 '18 at 22:19

I've never used non dbo schemas, so I wouldn't catch that. But why doesn't it work? I'm not specifying schema anywhere so does it default to dbo only? – Zach Smith Jan 20 '18 at 18:32

If you have a table, for example, called test.Table1, where "test" is the schema, your deletes will fail if trying to execute "DELETE FROM Table1". It needs to be DELETE FROM test.Table1. – influent Jan 22 '18 at 19:03

2 @influent - now it takes non-dbo schemas into account – Zach Smith Jan 30 '18 at 12:19

Unfortunately this seems to fail if there are FK constraints. The `ALTER TABLE` bit to disable constraints fails. – Douglas Gaskell Nov 10 '18 at 2:48

4

```
Set nocount on

Exec sp_MSForEachTable 'Alter Table ? NoCheck Constraint All'

Exec sp_MSForEachTable
'
If ObjectProperty(Object_ID(''?''), ''TableHasForeignRef'')=1
Begin
-- Just to know what all table used delete syntax.
Print ''Delete from '' + ''?''
Delete From ?
End
Else
Begin
-- Just to know what all table used Truncate syntax.
```

edited Feb 20 '14 at 13:08

Yaroslav
5,651 10 39 78

answered Jan 3 '13 at 8:08

yadavr
754 3 15 36

3

In my case, I needed to set QUOTED_IDENTIFIER on. This led to a slight modification of Mark Rendle's answer above:

```
EXEC sp_MSForEachTable 'DISABLE TRIGGER ALL ON ?'
GO
EXEC sp_MSForEachTable 'ALTER TABLE ? NOCHECK CONSTRAINT ALL'
GO
EXEC sp_MSForEachTable 'SET QUOTED_IDENTIFIER ON; DELETE FROM ?'
GO
EXEC sp_MSForEachTable 'ALTER TABLE ? CHECK CONSTRAINT ALL'
GO
EXEC sp_MSForEachTable 'ENABLE TRIGGER ALL ON ?'
GO
```

answered Apr 9 '18 at 14:43

William Jockusch
9,015 43 158 275

1

You could delete all the rows from all tables using an approach like Rubens suggested, or you could just drop and recreate all the tables. Always a good idea to have the full db creation scripts anyway so that may be the easiest/quickest method.

answered Dec 14 '09 at 9:26

AdaTheDev
112k 23 171 179

seems OP is concerned about referential integrity and triggers; this case, your got best solution. I'm dropping my answer =) – Rubens Farias Dec 14 '09 at 9:31

0

For some requirements we might have to skip certain tables. I wrote the below script to add some extra conditions to filter the list of tables. The below script will also display the pre delete count and post delete count.

```
IF OBJECT_ID('TEMPDB..#TEMPRECORDCOUNT') IS NOT NULL
DROP TABLE #TEMPRECORDCOUNT

CREATE TABLE #TEMPRECORDCOUNT
    (    TABLENAME NVARCHAR(128)
        ,PREDELETECOUNT BIGINT
        ,POSTDELETECOUNT BIGINT
    )

INSERT INTO #TEMPRECORDCOUNT (TABLENAME, PREDELETECOUNT, POSTDELETECOUNT)

SELECT   O.name TableName
        ,DDPS.ROW_COUNT PREDELETECOUNT
        ,NULL  FROM sys.objects O

INNER JOIN (

            SELECT OBJECT_ID, SUM(row_count) ROW_COUNT
            FROM SYS.DM_DB_PARTITION_STATS
            GROUP BY OBJECT_ID
           ) DDPS ON DDPS.OBJECT_ID = O.OBJECT_ID
WHERE O.type = 'U' AND O.name NOT LIKE 'OC%' AND O.schema_id = 1

DECLARE @TableName NVARCHAR(MAX);
DECLARE TableDeleteCursor CURSOR FAST_FORWARD
FOR
SELECT TableName from #TEMPRECORDCOUNT

OPEN TableDeleteCursor
FETCH NEXT FROM TableDeleteCursor INTO @TableName

WHILE (@@FETCH_STATUS <> -1)
BEGIN
IF (@@FETCH_STATUS <> -2)
BEGIN
DECLARE @STATEMENT NVARCHAR(MAX);
SET @STATEMENT = ' DISABLE TRIGGER ALL ON ' + @TableName +
                 '; ALTER TABLE ' + @TableName + ' NOCHECK CONSTRAINT ALL' +
                 '; DELETE FROM ' + @TableName +
```

```
END
CLOSE TableDeleteCursor
DEALLOCATE TableDeleteCursor

UPDATE T
 SET T.POSTDELETECOUNT = I.ROW_COUNT
 FROM #TEMPRECORDCOUNT T
 INNER JOIN (
                SELECT O.name TableName, DDPS.ROW_COUNT ROW_COUNT
                FROM sys.objects O
                INNER JOIN (

                        SELECT OBJECT_ID, SUM(row_count) ROW_COUNT
                        FROM SYS.DM_DB_PARTITION_STATS
                        GROUP BY OBJECT_ID
                       ) DDPS ON DDPS.OBJECT_ID = O.OBJECT_ID
                WHERE O.type = 'U' AND O.name NOT LIKE 'OC%' AND O.schema_id = 1

            ) I ON I.TableName COLLATE DATABASE_DEFAULT = T.TABLENAME

SELECT * FROM #TEMPRECORDCOUNT
ORDER BY TABLENAME ASC
```

answered Mar 16 '17 at 16:10

Balasubramanian S

11 4

0

This answer builds on Zach Smith's answer by **resetting the identity column** as well:

1. Disabling all constraints
2. Iterating through all tables except those you choose to exclude
3. Deletes all rows from the table
4. **Resets the identity column if one exists**
5. Re-enables all constraints

```
declare @catalog nvarchar(250);
declare @schema nvarchar(250);
declare @tbl nvarchar(250);
DECLARE i CURSOR LOCAL FAST_FORWARD FOR select
                                        TABLE_CATALOG,
                                        TABLE_SCHEMA,
                                        TABLE_NAME
                                        from INFORMATION_SCHEMA.TABLES
                                        where
                                        TABLE_TYPE = 'BASE TABLE'
                                        AND TABLE_NAME != 'sysdiagrams'
                                        AND TABLE_NAME != '__RefactorLog'
                                        -- Optional
                                        -- AND (TABLE_SCHEMA = 'dbo')

OPEN i;
FETCH NEXT FROM i INTO @catalog, @schema, @tbl;
WHILE @@FETCH_STATUS = 0
    BEGIN
        DECLARE @sql NVARCHAR(MAX) = N'DELETE FROM [' + @catalog + '].[' + @schema + '].
[' + @tbl + '];'
        /* Make sure these are the commands you want to execute before executing */
        PRINT 'Executing statement: ' + @sql
        --EXECUTE sp_executesql @sql

        -- Reset identity counter if one exists
        IF ((SELECT OBJECTPROPERTY( OBJECT_ID(@catalog + '.' + @schema + '.' + @tbl),
'TableHasIdentity')) = 1)
        BEGIN
            SET @sql = N'DBCC CHECKIDENT ([' + @catalog + '.' + @schema + '.' + @tbl +
'], RESEED, 0)'
            PRINT 'Executing statement: ' + @sql
            --EXECUTE sp_executesql @sql
        END

        FETCH NEXT FROM i INTO @catalog, @schema, @tbl;
    END
CLOSE i;
DEALLOCATE i;

-- Re-enable all constraints again
EXEC sp_msforeachtable "ALTER TABLE ? WITH CHECK CHECK CONSTRAINT all"
```

For one reason or another this mostly fails as it throws FK constraint errors. – Douglas Gaskell Nov 10 '18 at 2:46

0

```
--Load tables to delete from
SELECT
DISTINCT
' Delete top 1000000 from <DBName>.<schema>.' + c.TABLE_NAME + ' WHERE <Filter Clause
Here>' AS query,c.TABLE_NAME AS TableName, IsDeleted=0, '<InsertSomeDescriptorHere>' AS
[Source]--,t.TABLE_TYPE, c.*
            INTO dbo.AllTablesToDeleteFrom
            FROM INFORMATION_SCHEMA.TABLES AS t
            INNER JOIN information_schema.columns c ON c.TABLE_NAME = t.TABLE_NAME
    WHERE c.COLUMN_NAME = '<column name>'
           AND c.TABLE_SCHEMA = 'dbo'
           AND c.TABLE_CATALOG = '<DB Name here>'
           AND t.TABLE_TYPE='Base table'
           --AND t.TABLE_NAME LIKE '<put filter here>'

            DECLARE @TableSelect NVARCHAR(1000)= '';
            DECLARE @Table NVARCHAR(1000)= '';
            DECLARE @IsDeleted INT= 0;
            DECLARE @NumRows INT = 1000000;
            DECLARE @Source NVARCHAR(50)='';


            WHILE ( @IsDeleted = 0 )
                BEGIN
                --This grabs one table at a time to be deleted from. @TableSelect has
the sql to execute. it is important to order by IsDeleted ASC
                --because it will pull tables to delete from by those that have a
0=IsDeleted first. Once the loop grabs a table with IsDeleted=1 then this will pop out
of loop

                    SELECT TOP 1
                           @TableSelect = query,
                           @IsDeleted = IsDeleted,
                           @Table = TableName,
                           @Source=[a].[Source]
                    FROM   dbo.AllTablesToDeleteFrom a
                    WHERE a.[Source]='SomeDescriptorHere'--use only if needed
```

```
                    WHILE ( @NumRows = 1000000 )--only delete a million rows at a time?

                    BEGIN
                    EXEC sp_executesql @TableSelect;
                    SET @NumRows = @@ROWCOUNT;
                    --IF @NumRows = 1000000 --can do something here if needed
                    --One wants this loop to continue as long as a million rows is
deleted. Once < 1 million rows is deleted it pops out of loop
                    --and grabs next table to delete
                    --    BEGIN
                    --SELECT  @NumRows;--can add this in to see current number of
deleted records for table
                            INSERT  INTO dbo.DeleteFromAllTables
                                    ( tableName,
                                      query,
                                      cnt,
                                      [Source]
                                    )
                            SELECT  @Table,
                                    @TableSelect,
                                    @NumRows,
                                    @Source;
                     --    END;
                END;



SET @NumRows = 1000000;

UPDATE  a
SET     a.IsDeleted = 1
FROM    dbo.AllTablesToDeleteFrom a
WHERE   a.TableName = @Table;
--flag this as deleted so you can move on to the next table to delete from

END;
```

edited Oct 3 '18 at 15:38

habib

1,268 1 13 29

answered Oct 3 '18 at 14:38

jps

1

-3

edited Apr 23 '12 at 19:56

MADCookie
1,551 2 21 44

answered Apr 23 '12 at 19:25

malej
67 1 1

4 this doesn't accomplishes the initial requirement, to delete ALL TABLES... – franko_camron Jan 28 '16 at 16:32