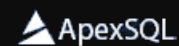




How to generate random SQL Server test data using T-SQL

January 26, 2017 by [Daniel Calbimonte](#)

100% free SQL tools



Introduction

In this article, we will talk about generating random values for testing purposes.

I once had a customer with software that worked fine in the demo with 30 rows, but after some months, the software had more than a million rows and it became very slow. The problem was not SQL Server, the problem was the application, which was not designed for tables with millions of rows. The customer sued to the software provider and lawyers were needed to create a resolution. If the provider had tested the software with millions of rows, this problem would have never happened.

That is why, it is very important to generate data and test the software with millions of rows. This is not always an easy task. In this article, we will give you some useful T-SQL tips that may help or at least inspire you on this. In general, random data is very useful for testing purposes, to learn about query efficiency, demos and more.

In this article, we will teach how to generate up to a million rows of random data in SQL Server including:

1. combinations of user names and last names
2. integer values
3. real numbers with a specific range
4. passwords in SQL Server
5. emails
6. country names



Requirements

Requirements

1. SQL Server
2. SQL Server Management Studio (SSMS)
3. Adventure Works 2014 Full and Adventure Works DW 2014 databases

Getting started

1. Generate a million first and last names

In the first example, we will use the DimCustomer table from the AdventureWorksDW database mentioned in the requirements. This table contains 18,000 rows. We will use a cross join to generate all the possible combinations of names and last names. With the cross join you can generate a total combination of 341,658,256 users for your tests. The following example shows how to create a combination of 1 million user names and last names:

```
se
USE [AdventureWorksDW2014]
GO
--Change 1000000 to the number of your preference for your needs
SELECT TOP 1000000
    c1.[FirstName],
    c2.[LastName]

    FROM [dbo].[DimCustomer] c1
CROSS JOIN
DimCustomer c2
```

The example will show 1,000,000 rows of names and last names:



	FirstName	LastName
1	Jon	Yang
2	Eugene	Yang
3	Ruben	Yang
4	Christy	Yang
5	Elizabeth	Yang
6	Julio	Yang
7	Janet	Yang
8	Marco	Yang
9	Rob	Yang

Figure 1. Generating all the possible combinations between the first and last name

If you want to generate 34 million rows, you have to replace this line:



```
SELECT TOP 1000000
```

With this one:

```
SELECT TOP 34000000
```

The query generates a Cartesian product with all the combinations and TOP limits the number of rows.

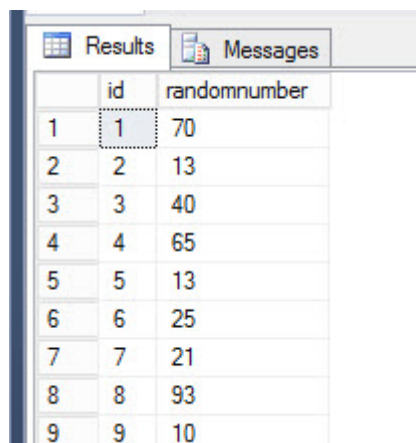
2. Generate random integer values

The following example will show how to create a table of 1000 rows with random values from 1 to 100. We will use the RAND function to create random values and CHECKSUM(NEWID()) to generate distinct values. We use the cast to convert the values from real to integer:

```
with randowvalues
as (
    select 1 id, CAST(RAND(CHECKSUM(NEWID()))*100 as int) randomnumber
    --select 1 id, RAND(CHECKSUM(NEWID()))*100 randomnumber
    union all
    select id + 1, CAST(RAND(CHECKSUM(NEWID()))*100 as int) randomnumber
    --select id + 1, RAND(CHECKSUM(NEWID()))*100 randomnumber
    from randowvalues
    where
        id < 1000
)

select *
from randowvalues
OPTION (MAXRECURSION 0)
```

The code will show 100 values between 1 to 100:



	id	randomnumber
1	1	70
2	2	13
3	3	40
4	4	65
5	5	13
6	6	25
7	7	21
8	8	93
9	9	10



Figure 2. Integer random values generated in SQL Server

If you want to generate 10000 values, change this line:

```
id < 1000
```

With this one:

```
id < 10000
```

If you want to generate values from 1 to 10000 change these lines:

```
select 1 id, CAST(RAND(CHECKSUM(NEWID()))*10000 as int) randomnumber
union all
select id + 1, CAST(RAND(CHECKSUM(NEWID()))*10000 as int) randomnumber
from randowvalues
```

If you want to generate **real** values instead of integer values use these lines replace these lines of the code displayed before:

```
select 1 id, CAST(RAND(CHECKSUM(NEWID()))*10000 as int) randomnumber
union all
select id + 1, CAST(RAND(CHECKSUM(NEWID()))*10000 as int) randomnumber
from randowvalues
```

And use these ones:

```
select 1 id, RAND(CHECKSUM(NEWID()))*10000 randomnumber
union all

select id + 1, RAND(CHECKSUM(NEWID()))*10000 randomnumber
from randowvalues
```

The query will show real numbers from 0 to 100

3. Random real numbers with a specific range

Another typical request is to provide random values with specific ranges. The following example will show a range of temperatures in °F (I really prefer the metric system, but I will do an exception this time).

The human body has the following fluctuations of temperature: 95 to 105.8 °F (Normal temperature is from 97.7–99.5 °F, higher values means fever, Hyperthermia and lower values



Hypothermia).

In this example, we will generate values between 95 to 105.8 °F:

```
with randowvalues
as (

--10.8 is the difference between 105.8 minus 95

    select 1 id,CAST(RAND(CHECKSUM(NEWID()))*10.8 as real) +95 as randomnu
mber
    union all
    select id + 1,CAST(RAND(CHECKSUM(NEWID()))*10.8 as real) +95 as rand
omnumber
    from randowvalues
    where
        id < 100
)

select *
from randowvalues
OPTION (MAXRECURSION 0)
```

The result of the T-SQL statement will be values from 95 to 105.8 °F:

Figure 3. Random real numbers from 0 to 100

If you want real numbers from 6 to 10, change these lines of code:

```
select 1 id,CAST(RAND(CHECKSUM(NEWID()))*10.8 as real) +95 as randomnumber
union all
select id + 1,CAST(RAND(CHECKSUM(NEWID()))*10.8 as real) +95 as randomnumber
```

With these ones:

```
select 1 id,CAST(RAND(CHECKSUM(NEWID()))*4 as real) +6 as randomnumber
union all
select id + 1,CAST(RAND(CHECKSUM(NEWID()))*4 as real) +6 as randomnumber
```

Where 6 is the minimum value and 4 is the difference between 10 and 6.

4. Random passwords in SQL Server



Another common request is to generate passwords. This example is used for initial passwords that

will be changed latter by the user or when the user forgets the password.

The following example will generate 100 passwords:

```
with randowvalues
as(
    select 1 id, CONVERT(varchar(20), CRYPT_GEN_RANDOM(10)) as mypassword
    union all
    select id + 1, CONVERT(varchar(20), CRYPT_GEN_RANDOM(10)) as mypassw
ord
    from randowvalues
    where
        id < 100
)

select *
from randowvalues
OPTION(MAXRECURSION 0)
```

The values displayed by the T-SQL statements are the following:

Figure 4. Random passwords

We use the `CRYPT_GEN_RANDOM` function to generate passwords and we will then convert them to a varchar. The function returns hexadecimal values and we convert it to characters.

5. Generating random emails

The following example, will generate some passwords. We will use the First names and last names of the example 1 of the table DimCustomer to generate random fake emails in SQL Server. If we have for example a Customer named John Smith, we will generate an email that can be jsmith@gmail.com, or use a Hotmail or Yahoo account. Let's take a look to the code:

```
USE [AdventureWorksDW2014]
GO
WITH random
as
(
    SELECT TOP 10000
        c1.[FirstName],
        c2.[LastName], CAST(RAND(CHECKSUM(NEWID()))*3 as int) randomemail

    FROM [dbo].[DimCustomer] c1
    CROSS JOIN
    DimCustomer c2
)
select
    Firstname,
    Lastname,
    email=
```



```

CASE
  when randomemail =0 then
    lower(left(FirstName,1)+[LastName])+'@hotmail.com'
  when randomemail =1 then
    lower(left(FirstName,1)+[LastName])+'@gmail.com'
  else
    lower(left(FirstName,1)+[LastName])+'@yahoo.com'
END
from random

```

The code will extract the first letter of the Firstname and concatenate with the last name and concatenate Hotmail or gmail or yahoo randomly:

Figure 5. Random emails

6. Generate country names randomly

This last example will show how to generate random country names. We will use the table Person.CountryRegion from the adventureworks database and we will add an id using the Row_number function:

```

SELECT ROW_NUMBER() OVER(ORDER BY Name) AS id,
       [Name]
FROM [AdventureWorks2016CTP3].[Person].[CountryRegion]

```

This table contains 238 countries:

Figure 6. List of countries in the Person.CountryRegion table of the adventureworks

We will use the list of random numbers of the second example to generate values from 1 to 238 (238 is the total number of countries) we will use an inner join to join the random numbers with the countries and generate country names randomly:

```

;with countries
as
(
  -- Create a country id and a country name. The countryid will be used to ---j
  oin with the random numbers
  SELECT ROW_NUMBER() OVER(ORDER BY Name) AS countryid,
         [Name]
  FROM [AdventureWorks2016CTP3].[Person].[CountryRegion]
),
---Create 1000 random numbers from 1 to 238

```



```

randomvalues
as (
    select 1 id, CAST(RAND(CHECKSUM(NEWID()))*238 as int) randomnumber
    union all
    select id + 1, CAST(RAND(CHECKSUM(NEWID()))*238 as int) randomnumber
    from randomvalues
    where
        id < 1000
)

--- Join countries with random numbers to generate country names randomly

select randomnumber,c.Name
from randomvalues r
inner join countries c
on r.randomnumber=c.countryid
order by id

OPTION (MAXRECURSION 0)

```

The T-SQL statements will generate a list of countries randomly:

Figure 7. List of countries generated randomly

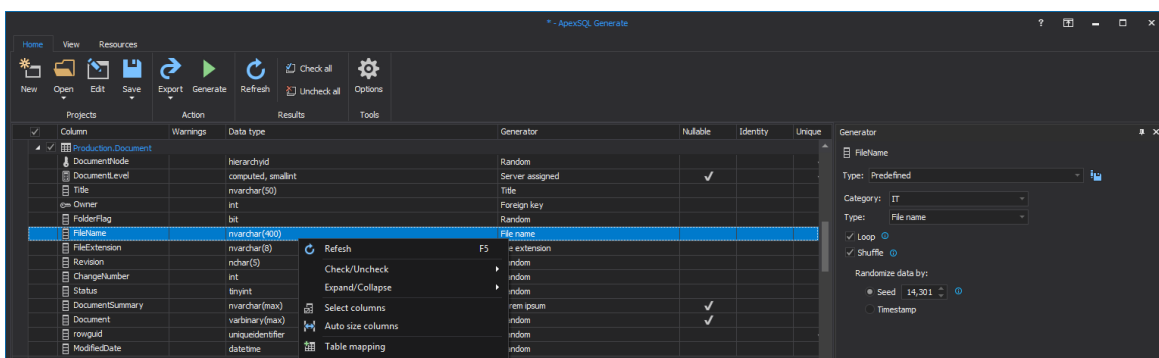
Conclusions

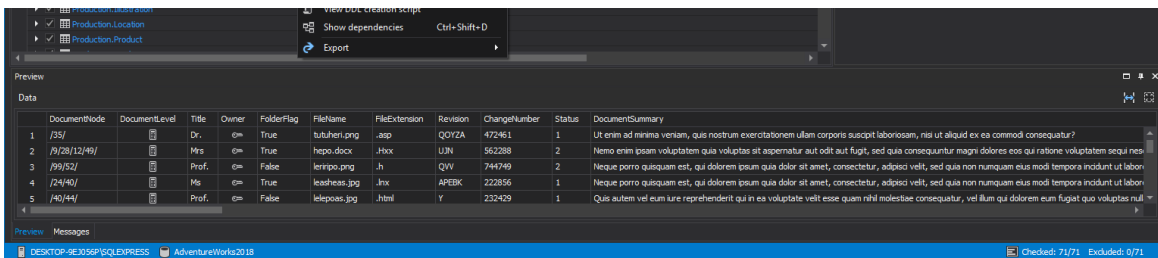
Generate random values for testing can be difficult. However, this article can be useful to inspire you to create your own data. Sometimes we can use existing tables to generate more values. Sometimes we can create the data from zero. In this example, we show how to create data using the Random function.

In this article, we generated millions of first names and last names, random integer values, real values with specific ranges, random passwords, random emails using first and last names and random country names.

See more

To generate millions of rows of test data quickly, consider ApexSQL Generate, a [test data generator](#) specifically designed for SQL Server developers





	DocumentId	DocumentLevel	Title	Owner	FolderFlag	Filename	FileExtension	Revision	ChangeNumber	Status	DocumentSummary
1	/35/		Dr.	o=	True	tutuhui.png	.asp	QOYZA	472461	1	Ut enim ad minima veniam, quis nostrum exercitationem ullam corporis suscipit laboriosam, nisi ut aliquid ex ea commodi consequatur?
2	/9/28/12/49/		Mrs.	o=	True	hepo.docx	.hex	UJN	562288	2	Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt.
3	/99/52/		Prof.	o=	False	terripa.png	.h	QVW	744749	2	Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem.
4	/24/40/		Ms	o=	True	leasheat.jpg	.inx	APBKB	222856	1	Neque porro quisquam est, qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit, sed quia non numquam eius modi tempora incidunt ut labore et dolore magnam aliquam quaerat voluptatem.
5	/40/44/		Prof.	o=	False	lelepoas.jpg	.html	Y	232429	1	Quis autem vel eum iure reprehenderit qui in ea voluptate velit esse quam nihil molestiae consequatur, vel illum qui dolorem eum fugiat quo voluptas nulla pariatur?

Generate synthetic SQL Server test data



Daniel Calbimonte

Daniel Calbimonte is a Microsoft Most Valuable Professional, Microsoft Certified Trainer and Microsoft Certified IT Professional for SQL Server. He is an accomplished SSIS author, teacher at IT Academies and has over 13 years of experience working with different databases.

He has worked for the government, oil companies, web sites, magazines and universities around the world. Daniel also regularly speaks at SQL Servers conferences and blogs. He writes SQL Server training materials for certification exams.

He also helps with [translating SQLShack articles to Spanish](#)

[View all posts by Daniel Calbimonte](#)

Related Posts:

1. [Functions and stored procedures comparisons in SQL Server](#)
2. [How to work with SQL random numbers in SSIS](#)
3. [3 ways to improve T-SQL performance](#)
4. [SQL Server performance myth busters](#)
5. [How to quickly generate a large number of dimension tables for reporting applications](#)

T-SQL, Testing

33,740 Views

Comments

Community

 Login

 Recommend 3

 Tweet

 Share

Sort by Best



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

**Dennis V McEnaney** • 6 months ago • edited

I find the following to be a slightly simpler solution - e.g. to generate 10 rows:

```
WITH  
BT AS (  
SELECT  
1 AS RowNum  
)  
DT AS (  
SELECT  
*  
FROM  
BT  
UNION ALL  
SELECT  
DT.RowNum + 1 AS RowNum  
FROM  
BT DT2
```

[see more](#)[^](#) | [v](#) • [Reply](#) • [Share](#) ›

ALSO ON SQL SHACK

Cómo poder importar los datos de un archivo de

2 comments • 5 months ago



Luis Santaella — La parte que dice NOTA me salvo la vida ejecutaba desde el sql

sp_updatestats overview and usage

1 comment • 4 months ago



Vesna — Great article! Thanks!

Difference between SQL Truncate and SQL Delete

1 comment • 5 months ago



zouhir Boudelli — thank you for this amazing tutos .

SQL Carriage Return or Tab in SQL Server string

1 comment • a month ago



Gayathridevi Batchu — Very nicely illustrated. I couldnt however simulate CR and LF

