← Back to all posts

Creating Random SQL Server Test Data

SEP 12, 2008 / DEVELOPMENT >> SQL

Recently I was tasked with the creation of a large database system that consisted of a database table with 5 date columns, and a varchar primary key. This table was to hold upwards of 3.5 million records, and I needed to know exactly how much disk space was going to be needed to store not only the database but also the index required to facilitate the search requirements. After looking for a number of different ways to do this, and many free third-party tools I decided that the most simple way to do this with the tools that I had available was to generate my own method to populate a test database.

To get started with this I took the specific requirements of my database table and identified a routine to load data that would mimic the actual data. The reason for this was that I was two months out from being able to obtain actual data. My table as I mentioned consisted of a varchar primary key column that was 10 characters in length. This obviously had to be generated in a unique manner. I then had a total of 5 date columns each date entry for each record had to be different, but the dates would all be within the last 2 years. With this in mind, I had a population scheme that I could work with.

- Must populate primary key value with 10 characters unique identifier
- All date values should be dates between today and 2 years before today
- All date values in individual records should be different
- An index is needed to cover the primary key and ALL date columns due to specific business needs

The following sections will demonstrate how I loaded the data, including randomization to ensure a varied collection of date values were included.

Creating the Test Table

Obviously the first step of this process was to create my test table, for the sake of this exercise I am going to leave out the extra columns that were included in my testing as this article is really talking about how to create test datasets, and using TSQL loops and random number generators. The script below creates a minimal test table.

Create Test Table

```
CREATE TABLE dbo.TestTableSize
(
    MyKeyField VARCHAR(10) NOT NULL,
    MyDate1 DATETIME NOT NULL,
    MyDate2 DATETIME NOT NULL,
    MyDate3 DATETIME NOT NULL,
    MyDate4 DATETIME NOT NULL,
    MyDate5 DATETIME NOT NULL
)
```

Creating the Population Process

Now that I have the test table to load data into, I can start to actually create the TSQL that will fill my table. This code will be discussed in a few portions: Needed variables, one-time value setting, and the population loop, in the end, it will create the entire process needed to fill a table with three million records.

Variable Declarations

To start out process we will need a few variables; one to count the rows, one for a string representation of the row, one to hold the random number that is used to calculate our date, two to hold the upper/lower limits for the random number, and lastly one to hold our insert date. Below is the script needed to create the various variables.

Variable Declarations

```
DECLARE @RowCount INT
DECLARE @RowString VARCHAR(10)
DECLARE @Random INT
DECLARE @Upper INT
DECLARE @Lower INT
DECLARE @InsertDate DATETIME
```

Set One Time Values

After we have defined our variables it is necessary to configure the items that are set once per run. This involves setting the row count value to 0, setting the lower to -730 as valid dates can be 2 years prior to today, thus a negative 365 * 2. And lastly to set the upper limit to -1 to only allow yesterday to be selected.

One-Time Values

```
SET @Lower = -730
SET @Upper = -1
SET @RowCount = 0
```

Populate the Table

This is the most complicated piece of code, therefore it will be presented in sections, when then, in the end, we will re-present the entire code needed to populate the table. First we start a while loop that runs while the row count is less than three million.

Populating the Table

```
WHILE @RowCount < 3000000
BEGIN
```

Now that we have this completed we perform the calculations needed to ready the inserts. First, we will convert the row count into a string value, then we will get a random number between the upper and lower bounds set by our variables. This will get the day to offset that we need. Lastly, we will calculate the insert date using the random offset established previously. This code allows us to properly work with our insert later.

Preparing Values

```
SET @RowString = CAST(@RowCount AS VARCHAR(10))

    SELECT @Random = ROUND(((@Upper - @Lower -1) * RAND() + @Lower), 0)

    SET @InsertDate = DATEADD(dd, @Random, GETDATE())
```

Now that we have our values, we can start our insert. As noted in our requirements none of the 5 dates in a single row can be the same, so to get around the issue we will simply add a day to the MyDate2 column, two days tot he MyDate3 column, etc until we have all five dates. At the end of this, we will increment the counter and end the block of code.

Perform Inserts

```
INSERT INTO TestTableSize
        (MyKeyField
        ,MyDate1
        ,MyDate2
        ,MyDate3
        ,MyDate4
        ,MyDate5)
    VALUES
        (REPLICATE('0', 10 - DATALENGTH(@RowString)) + @RowString
        , @InsertDate
        ,DATEADD(dd, 1, @InsertDate)
        ,DATEADD(dd, 2, @InsertDate)
        ,DATEADD(dd, 3, @InsertDate)
        ,DATEADD(dd, 4, @InsertDate))

    SET @RowCount = @RowCount + 1
END
```

Complete Population Script

In the end, we have presented the following script that will load random content into the table for testing. Simply execute this script and wait for the load to complete.

Full Script

```
DECLARE @RowCount INT
DECLARE @RowString VARCHAR(10)
DECLARE @Random INT
DECLARE @Upper INT
DECLARE @Lower INT
DECLARE @InsertDate DATETIME

SET @Lower = -730
SET @Upper = -1
SET @RowCount = 0

WHILE @RowCount < 3000000
BEGIN
    SET @RowString = CAST(@RowCount AS VARCHAR(10))
    SELECT @Random = ROUND(((@Upper - @Lower -1) * RAND() + @Lower), 0)
    SET @InsertDate = DATEADD(dd, @Random, GETDATE())

    INSERT INTO TestTableSize
        (MyKeyField
        ,MyDate1
        ,MyDate2
        ,MyDate3
        ,MyDate4
        ,MyDate5)
    VALUES
        (REPLICATE('0', 10 - DATALENGTH(@RowString)) + @RowString
        , @InsertDate
        ,DATEADD(dd, 1, @InsertDate)
        ,DATEADD(dd, 2, @InsertDate)
        ,DATEADD(dd, 3, @InsertDate)
        ,DATEADD(dd, 4, @InsertDate))

    SET @RowCount = @RowCount + 1
END
```

Summary

This article presented a quick overview of a method to insert random test data into a database table, additionally presented is the method of calculating a random number as well as randomizing data input based on a random offset. This may not be the best way to create a random data set, but it does work very well if you do not have any of the popular third-party tools!

About Mitch

Sharing information with a worldwide audience is Mitch's passion. If you would like him to speak at your event, reach out today!

Book Mitch

Stay Connected

Subscribe to receive email updates with new technology blog posts.

Subscribe