# Creating a comma-separated list (SQL Spackle)

By Wayne Sheffield

*“SQL Spackle” is a collection of short articles written based on multiple requests for similar code. These short articles are NOT meant to be complete solutions. Rather, they are meant to “fill in the cracks”.*

*--Phil McCracken*

## Introduction

A frequent request found on the forums is to generate a comma-separated list of values, usually while maintaining other columns in a normal column list.

***The Test Data***

The test data will contain 1000 account numbers, and 100 random 3 characters for each account. This method utilizes an inline tally table. Please see Jeff Moden’s article *The "Numbers" or "Tally" Table: What it is and how it replaces a loop* at <http://www.sqlservercentral.com/articles/T-SQL/62867/> for more information on how a tally table works and what all you can do with it.

-- Conditionally drop the test table to make reruns easier.

IF OBJECT\_ID('tempdb..#TestData','U') IS NOT NULL DROP TABLE #TestData;

CREATE TABLE #TestData (AccountNumber INT,

 Value char(3));

-- Build 1000 account numbers with random 3 character data.

;WITH

TENS (N) AS (SELECT 0 UNION ALL SELECT 0 UNION ALL SELECT 0 UNION ALL

 SELECT 0 UNION ALL SELECT 0 UNION ALL SELECT 0 UNION ALL

 SELECT 0 UNION ALL SELECT 0 UNION ALL SELECT 0 UNION ALL SELECT 0),

THOUSANDS (N) AS (SELECT 1 FROM TENS t1 CROSS JOIN TENS t2 CROSS JOIN TENS t3),

MILLIONS (N) AS (SELECT 1 FROM THOUSANDS t1 CROSS JOIN THOUSANDS t2),

TALLY (N) AS (SELECT ROW\_NUMBER() OVER (ORDER BY (SELECT 0)) FROM MILLIONS)

INSERT INTO #TestData

SELECT TOP (100000)

 AccountNumber = CASE WHEN (N%1000) = 0 THEN 1000 ELSE N%1000 END,

 Value = CHAR(CONVERT(INT,RAND(CHECKSUM(NEWID()))\*10)+64) +

 CHAR(CONVERT(INT,RAND(CHECKSUM(NEWID()))\*10)+64) +

 CHAR(CONVERT(INT,RAND(CHECKSUM(NEWID()))\*10)+64)

 FROM TALLY;

-- Add a clustered index to the table

CREATE CLUSTERED INDEX IX\_#TestData\_Cover ON #TestData (AccountNumber, Value);

***The Solution (SQL 2005+)***

This solution utilizes features introduced in SQL 2005. It starts off with a CTE (common table expression) of all of the distinct AccountNumbers in the table. For each AccountNumber, we get a comma separated list of the Value field, sorted by the Value field.

WITH CTE AS

(

SELECT DISTINCT

 AccountNumber

 FROM #TestData

)

SELECT AccountNumber,

 CommaList = STUFF((SELECT ',' + Value

 FROM #TestData

 WHERE AccountNumber = CTE.AccountNumber

 ORDER BY Value

 FOR XML PATH(''),

 TYPE).value('.','varchar(max)'),1,1,'')

 FROM CTE

 ORDER BY AccountNumber;

The key to creating the comma separated list is the correlated subquery. Working from the inside out, we get each value prefixed with a comma, order by the Value. The FOR XML PATH('') generates an XML structure, with an empty string as the root node. Since the field is ',' + Value (an unnamed expression), there is no name for the individual elements. What is left is a list of values, with each value prefixed with a comma. The TYPE clause specifies to return the data as an XML type. The .value('.','varchar(max)') takes each value, and converts it into a varchar(max) data type. The combination of the TYPE and .value means that values are created at XML tags (such as the ampersand (&), and the greater than (>) and less than (<) signs), will not be tokenized into their XML representations and will remain as is.

At this point, you will have a comma separated list of all values starting with a comma for each value. All that remains is to remove the very first leading comma from the entire string. To do this, we utilize the STUFF function. Using the string created by the FOR XML PATH(''), TYPE, and starting with the first character, we replace one character (the leading comma) with an empty string. (Note that if you wanted the string to be separated with a comma and a space, you would specify ', ', and replace 2 characters in the STUFF function with an empty string.

The subquery is correlated, meaning that it references a value outside of itself to control what it is doing. In this case, it is referencing the current AccountNumber from the CTE.