T-SQL split string

Ask Question

109

I have a SQL Server 2008 R2 column containing a string which I need to split by a comma. I have seen many answers on StackOverflow but none of them works in R2. I have made sure I have select permissions on any split function examples. Any help greatly appreciated.

45

sql sql-server tsql sql-server-2008 split

edited Jan 6 at 21:59
Hadi
26.5k 7 31 75

asked Jun 6 '12 at 12:50
Lee Grindon
686 2 8 8

7 This is one of the million answers that I like stackoverflow.com/a/1846561/227755 – nurettin Jun 6 '12 at 12:53

2 What do you mean "none of them work"? Can you be more specific? – Aaron Bertrand Jun 6 '12 at 13:01

Andy did point me in the right direction as I was executing the function incorrectly. This is why none of the other stack answers worked. My fault. – Lee Grindon Jun 6 '12 at 13:08

2 possible duplicate of Split string in SQL – Luv Jul 10 '13 at 4:35

There's a `mdq.RegexSplit` function in the "Master Data

Home

PUBLIC

Stack Overflow

Tags

Users

25 Answers

187

I've used this SQL before which may work for you:-

```
CREATE FUNCTION dbo.splitstring ( @stringToSplit VARCHAR(M
RETURNS
 @returnList TABLE ([Name] [nvarchar] (500))
AS
BEGIN

 DECLARE @name NVARCHAR(255)
 DECLARE @pos INT

 WHILE CHARINDEX(',', @stringToSplit) > 0
 BEGIN
  SELECT @pos  = CHARINDEX(',', @stringToSplit)
  SELECT @name = SUBSTRING(@stringToSplit, 1, @pos-1)

  INSERT INTO @returnList
  SELECT @name

  SELECT @stringToSplit = SUBSTRING(@stringToSplit, @pos+1
 END

 INSERT INTO @returnList
 SELECT @stringToSplit

 RETURN
END
```

and to use it:-

```
SELECT * FROM dbo.splitstring('91,12,65,78,56,789')
```

edited Aug 23 '12 at 7:57

5,357 2 23 19

1 Nice one, this is exactly what I was looking for thanks very much – Lee Grindon Jun 6 '12 at 12:56

1 Thanks a lot Andy. I made a small enhancement to your script to allow the function to return an item at a specific index in the split string. It is useful only in situations when you the structure of the column one is parsing. gist.github.com/klimaye/8147193 – CF_Maintainer Dec 27 '13 at 13:57

1 I posted some improvements (with backing test cases) to my github page here. I will post it as an answer in this Stack Overflow thread when I have enough rep to exceed post "protection" – mpag Jun 16 '16 at 19:29

8 Although this is a great answer, it is outdated... Procedural approaches (especially loops) are something to avoid... It's worth to look into newer answers... – Shnugo Feb 2 '17 at 10:56

2 I totally agree with @Shnugo. The looping splitters work but horribly slow. Something like this sqlservercentral.com/articles/Tally+Table/72993 is far better. Some other excellent set based options can be found here. sqlperformance.com/2012/07/t-sql-queries/split-strings – Sean Lange Apr 26 '18 at 18:45

54

Instead of recursive CTEs and while loops, has anyone considered a more set-based approach?

```
CREATE FUNCTION [dbo].[SplitString]
    (
        @List nvarchar(MAX),
        @Delim nvarchar(255)
    )
```

```
        (
          SELECT
            [Value] = LTRIM(RTRIM(SUBSTRING(@List, [Numb
            CHARINDEX(@Delim, @List + @Delim, [Number])
          FROM (SELECT Number = ROW_NUMBER() OVER (ORDER
            FROM sys.all_columns) AS x
            WHERE Number <= LEN(@List)
            AND SUBSTRING(@Delim + @List, [Number], LEN(
        ) AS y
      );
```

If you want to avoid the limitation of the length of the string being <= the number of rows in `sys.all_columns` (9,980 in `model` in SQL Server 2017; much higher in your own user databases), you can use other approaches for deriving the numbers, such as building your own table of numbers. You could also use a recursive CTE in cases where you can't use system tables or create your own:

```
CREATE FUNCTION [dbo].[SplitString]
(
    @List nvarchar(MAX),
    @Delim nvarchar(255)
)
RETURNS TABLE WITH SCHEMABINDING
AS
    RETURN ( WITH n(n) AS (SELECT 1 UNION ALL SELECT n+1 F
      WHERE n < LEN(@List))
      SELECT [Value] FROM
      (
        SELECT
          [Value] = LTRIM(RTRIM(SUBSTRING(@List, n,
          CHARINDEX(@Delim, @List + @Delim, n) - n)))
        FROM n
        WHERE SUBSTRING(@Delim + @List, n, LEN(@Delim)) = (
      ) AS y
    );
```

But you'll have to append `OPTION (MAXRECURSION 0)` (or `MAXRECURSION <longest possible string length if < 32768>`) to the outer query in order to avoid errors with recursion for

alternative then see this answer as pointed out in the comments.

More on split functions, why (and proof that) while loops and recursive CTEs don't scale, and better alternatives, if splitting strings coming from the application layer:

http://www.sqlperformance.com/2012/07/t-sql-queries/split-strings

http://www.sqlperformance.com/2012/08/t-sql-queries/splitting-strings-now-with-less-t-sql

https://sqlblog.org/2010/07/07/splitting-a-list-of-integers-another-roundup

edited Mar 5 at 16:18

answered Nov 12 '13 at 17:13

Aaron Bertrand
216k 29 375 412

There is a small bug in this procedure for the case where there would be a null value at the end of the string - such as in '1,2,,4,' - as the final value is not parsed. To correct this bug, the expression "WHERE Number <= LEN(@List)" should be replaced with "WHERE Number <= LEN(@List) + 1". – SylvainL Sep 15 '14 at 8:33

@SylvainL I guess that depends on what behavior you want. In my experience, most people want to ignore any trailing commas as they don't really represent a real element (how many copies of a blank string do you need)? Anyway, the *real* way to do this - if you'll follow the second link - is to step messing around with splitting big ugly strings in slow T-SQL anyway. – Aaron Bertrand Sep 15 '14 at 8:37

Like you have said, most people want to ignore any trailing

this case but my comment is just a little note to make sure that no one forget about this possibility, as it can be quite real in many cases. – SylvainL Sep 15 '14 at 9:02

I have a weird behavior with that function. If I use directly a string as a parameter -- it works. If I have a varchar, it does not. You can reproduce easily: declare invarchar as varchar set invarchar = 'ta;aa;qq' SELECT Value from [dbo].[SplitString](invarchar, ';') SELECT Value from [dbo].[SplitString]('ta;aa;qq', ';') – Patrick Desjardins Jun 10 '15 at 4:00

I like this approach, but if the number of objects returned by `sys.all_objects` is less than number of the characters in the input string then it will truncate the string and values will go missing. Since `sys.all_objects` is just being used as a bit of a hack to generate rows, then there are better ways to do this, e.g. this answer. – knuckles Oct 10 '16 at 15:48

36

Finally the wait is over in **SQL Server 2016** they have introduced Split string function : `STRING_SPLIT`

```
select * From STRING_SPLIT ('a,b', ',') cs
```

All the other methods to split string like XML, Tally table, while loop, etc.. has been blown away by this `STRING_SPLIT` function.

Here is an excellent article with performance comparison : **Performance Surprises and Assumptions : STRING_SPLIT**

answered Mar 30 '16 at 10:32

Pரதீப்

76.7k 12 84 121

updated servers, but those of us still stuck on 2008/2008R2, will have to go with one of the other answers here. – mpag Jun 16 '16 at 19:34

2 You need to take a look at the compatibility level in your database. If it is lower than 130 you won't be able to use the STRING_SPLIT function. – Luis Teijon Jun 20 '17 at 16:06

Actually, if the compatibility isn't 130 and you're running 2016 (or Azure SQL) you can set the compatibility up to 130 using: ALTER DATABASE DatabaseName SET COMPATIBILITY_LEVEL = 130 – Michieal Dec 15 '17 at 15:26

19

The easiest way to do this is by using `XML` format.

1. Converting string to rows without table

QUERY

```
DECLARE @String varchar(100) = 'String1,String2,String3'
-- To change ',' to any other delimeter, just change ',' t
DECLARE @Delimiter CHAR = ','

SELECT LTRIM(RTRIM(Split.a.value('.', 'VARCHAR(100)'))) 'V
FROM
(
     SELECT CAST ('<M>' + REPLACE(@String, @Delimiter, '</
Data
) AS A
CROSS APPLY Data.nodes ('/M') AS Split(a)
```

RESULT

```
x---------x
| Value   |
x---------x
| String1 |
```

2. Converting to rows from a table which have an ID for each CSV row

SOURCE TABLE

```
x-----x-------------------------x
| Id  |           Value         |
x-----x-------------------------x
|  1  |  String1,String2,String3 |
|  2  |  String4,String5,String6 |
x-----x-------------------------x
```

QUERY

```
-- To change ',' to any other delimeter, just change ',' b
desired one
DECLARE @Delimiter CHAR = ','

SELECT ID,LTRIM(RTRIM(Split.a.value('.', 'VARCHAR(100)')))
FROM
(
     SELECT ID,CAST ('<M>' + REPLACE(VALUE, @Delimiter, '<
Data
     FROM TABLENAME
) AS A
CROSS APPLY Data.nodes ('/M') AS Split(a)
```

RESULT

```
x-----x----------x
| Id  |  Value   |
x-----x----------x
|  1  |  String1 |
|  1  |  String2 |
|  1  |  String3 |
|  2  |  String4 |
|  2  |  String5 |
|  2  |  String6 |
x-----x----------x
```

Sarath Avanavu
11.6k 7 46 66

This approach will break if `@String` contains forbidden characters... I just posted an answer to overcome this issue. – Shnugo Feb 2 '17 at 10:46

10

I needed a quick way to get rid of the `+4` from a **zip code**.

```
UPDATE #Emails
  SET ZIPCode = SUBSTRING(ZIPCode, 1, (CHARINDEX('-', ZIPC
  WHERE ZIPCode LIKE '%-%'
```

No proc... no UDF... just one tight little inline command that does what it must. Not fancy, not elegant.

Change the delimiter as needed, etc, and it will work for anything.

edited Jan 23 '14 at 20:08
Ignacio Correia
2,502 7 28 61

answered Jan 23 '14 at 18:33
CorvetteGuru
125 1 2

3 This isn't what the question is about. The OP has a value like '234,542,23' and they want to split it out into three rows ... 1st row: 234, 2nd row: 542, 3rd row: 23. Its a tricky thing to do in SQL. – codeulike Apr 28 '15 at 19:32

8

if you replace

```
WHILE CHARINDEX(',', @stringToSplit) > 0
```

with

```
WHILE LEN(@stringToSplit) > 0
```

you can eliminate that last insert after the while loop!

```
CREATE FUNCTION dbo.splitstring ( @stringToSplit VARCHAR(M
RETURNS
 @returnList TABLE ([Name] [nvarchar] (500))
AS
BEGIN

 DECLARE @name NVARCHAR(255)
 DECLARE @pos INT

 WHILE LEN(@stringToSplit) > 0
 BEGIN
  SELECT @pos  = CHARINDEX(',', @stringToSplit)


if @pos = 0
        SELECT @pos = LEN(@stringToSplit)


  SELECT @name = SUBSTRING(@stringToSplit, 1, @pos-1)

  INSERT INTO @returnList
  SELECT @name

  SELECT @stringToSplit = SUBSTRING(@stringToSplit, @pos+1
 END

 RETURN
END
```

edited Nov 8 '12 at 19:44

answered Nov 8 '12 at 19:31

AviG
97 1 2

This would result in the last character of the last element being truncated. i.e. "AL,AL" would become "AL" | "A" i.e. "ABC,ABC,ABC" would become "ABC" | "ABC" | "AB" – Microsoft Developer Apr 9 '13 at 16:37

appending `+1` to `SELECT @pos = LEN(@stringToSplit)` appears to address that issue. However, the `SELECT @stringToSplit = SUBSTRING(@stringToSplit, @pos+1, LEN(@stringToSplit)-@pos)` will return `Invalid length parameter passed to the LEFT or SUBSTRING function` unless you add `+1` to the third parameter of SUBSTRING as well. or you could replace that assignment with `SET @stringToSplit = SUBSTRING(@stringToSplit, @pos+1, 4000) --MAX len of nvarchar is 4000` – mpag Jun 15 '16 at 23:23

1 I posted some improvements (with backing test cases) to my github page here. I will post it as an answer in this Stack Overflow thread when I have enough rep to exceed post "protection" – mpag Jun 16 '16 at 19:29

I too have noted the issue pointed out by Terry above. But the given logic by @AviG is so cool that it does not fail in the middle for a long list of tokens. Try this test call to verify (This call should return 969 tokens) select * from dbo.splitstring('token1,token2,,,,,,,,token969') Then I tried the code given by mpag to check the results for same call above and found it can return only 365 tokens. Finally I fixed code by AviG above and posted the bug free function as a new reply below since comment here allows only limited text. Check reply under my name to try it. – Gemunu R Wickremasinghe Jun 23 '18 at 9:18

3

be replaced with set-based solution.

This code executes excellent.

```
CREATE FUNCTION dbo.SplitStrings
(
   @List       NVARCHAR(MAX),
   @Delimiter  NVARCHAR(255)
)
RETURNS TABLE
WITH SCHEMABINDING
AS
   RETURN
   (
      SELECT Item = y.i.value('(./text())[1]', 'nvarchar(4
      FROM
      (
        SELECT x = CONVERT(XML, '<i>'
          + REPLACE(@List, @Delimiter, '</i><i>')
          + '</i>').query('.')
      ) AS a CROSS APPLY x.nodes('i') AS y(i)
   );
GO
```

answered Jan 29 '17 at 14:32

Igor Micev
679 6 8

This approach will break if `@List` contains forbidden characters... I just posted an answer to overcome this issue. – Shnugo Feb 2 '17 at 10:44

2

I had to write something like this recently. Here's the solution I came up with. It's generalized for any delimiter string and I think it would perform slightly better:

```
    , @delim nvarchar(100) )
RETURNS
    @result TABLE
        ( [Value] nvarchar(4000) NOT NULL
        , [Index] int NOT NULL )
AS
BEGIN
    DECLARE @str nvarchar(4000)
          , @pos int
          , @prv int = 1

    SELECT @pos = CHARINDEX(@delim, @string)
    WHILE @pos > 0
    BEGIN
        SELECT @str = SUBSTRING(@string, @prv, @pos - @prv
        INSERT INTO @result SELECT @str, @prv

        SELECT @prv = @pos + LEN(@delim)
             , @pos = CHARINDEX(@delim, @string, @pos + 1)
    END

    INSERT INTO @result SELECT SUBSTRING(@string, @prv, 40
    RETURN
END
```

answered Jul 17 '13 at 3:46

p.s.w.g
122k 19 218 259

2

A solution using a CTE, if anyone should need that (apart from me, who obviously did, that is why I wrote it).

```
declare @StringToSplit varchar(100) = 'Test1,Test2,Test3';
declare @SplitChar varchar(10) = ',';

with StringToSplit as (
  select
      ltrim( rtrim( substring( @StringToSplit, 1, charinde
```

```
  union all

  select
      ltrim( rtrim( substring( Tail, 1, charindex( @SplitCl
    , substring( Tail, charindex( @SplitChar, Tail ) + 1,
  from StringToSplit
  where charindex( @SplitChar, Tail ) > 0

  union all

  select
      ltrim( rtrim( Tail ) ) Head
    , '' Tail
  from StringToSplit
  where charindex( @SplitChar, Tail ) = 0
    and len( Tail ) > 0
)
select Head from StringToSplit
```

answered Jul 19 '13 at 10:29

Torsten B. Hagemann

21 1

2

There is a correct version on here but I thought it would be nice to add a little fault tolerance in case they have a trailing comma as well as make it so you could use it not as a function but as part of a larger piece of code. Just in case you're only using it once time and don't need a function. This is also for integers (which is what I needed it for) so you might have to change your data types.

```
DECLARE @StringToSeperate VARCHAR(10)
SET @StringToSeperate = '1,2,5'

--SELECT @StringToSeperate IDs INTO #Test
```

```
DECLARE @CommaSeperatedValue NVARCHAR(255) = ''
DECLARE @Position INT = LEN(@StringToSeperate)

--Add Each Value
WHILE CHARINDEX(',', @StringToSeperate) > 0
BEGIN
    SELECT @Position  = CHARINDEX(',', @StringToSeperate)
    SELECT @CommaSeperatedValue = SUBSTRING(@StringToSeper

    INSERT INTO #IDs
    SELECT @CommaSeperatedValue

    SELECT @StringToSeperate = SUBSTRING(@StringToSeperate
LEN(@StringToSeperate)-@Position)

END

--Add Last Value
IF (LEN(LTRIM(RTRIM(@StringToSeperate)))>0)
BEGIN
    INSERT INTO #IDs
    SELECT SUBSTRING(@StringToSeperate, 1, @Position)
END

SELECT * FROM #IDs
```

answered Feb 12 '15 at 18:03

Bryce
415 5 15

if you were to `SET @StringToSeperate = @StringToSeperate+','` immediately before the `WHILE` loop I think you might be able to eliminate the "add last value" block. See also my sol'n on github – mpag Jun 16 '16 at 19:40

Which answer is this based on? There are a lot of answers here, and it's a bit confusing. Thanks. – jpaugh Nov 1 '16 at 14:38

2

can select only required part from returning table:

```
CREATE FUNCTION dbo.splitstring ( @stringToSplit VARCHAR(M

RETURNS

 @returnList TABLE ([numOrder] [tinyint] , [Name] [nvarcha
BEGIN

 DECLARE @name NVARCHAR(255)

 DECLARE @pos INT

 DECLARE @orderNum INT

 SET @orderNum=0

 WHILE CHARINDEX('.', @stringToSplit) > 0

 BEGIN
    SELECT @orderNum=@orderNum+1;
  SELECT @pos  = CHARINDEX('.', @stringToSplit)
  SELECT @name = SUBSTRING(@stringToSplit, 1, @pos-1)

  INSERT INTO @returnList
  SELECT @orderNum,@name

  SELECT @stringToSplit = SUBSTRING(@stringToSplit, @pos+1
 END
    SELECT @orderNum=@orderNum+1;
 INSERT INTO @returnList
 SELECT @orderNum, @stringToSplit

 RETURN
END


Usage:


SELECT Name FROM
dbo.splitstring('ELIS.YD.CRP1.1.CBA.MDSP.T389.BT') WHERE
numOrder=5
```

answered Apr 17 '15 at 9:40

SNabi
38 5

2

If you need a quick ad-hoc solution for common cases with minimum code, then this recursive CTE two-liner will do it:

```
DECLARE @s VARCHAR(200) = ',1,2,,3,,,4,,,,5,'

;WITH
a AS (SELECT i=-1, j=0 UNION ALL SELECT j, CHARINDEX(',', (
i),
b AS (SELECT SUBSTRING(@s, i+1, IIF(j>0, j, LEN(@s)+1)-i-1
SELECT * FROM b
```

Either use this as a stand-alone statement or just add the above CTEs to any of your queries and you will be able to join the resulting table `b` with others for use in any further expressions.

edit (by Shnugo)

If you add a counter, you will get a position index together with the List:

```
DECLARE @s VARCHAR(200) = '1,2333,344,4'

;WITH
a AS (SELECT n=0, i=-1, j=0 UNION ALL SELECT n+1, j, CHARI
WHERE j > i),
b AS (SELECT n, SUBSTRING(@s, i+1, IIF(j>0, j, LEN(@s)+1)-
SELECT * FROM b;
```

```
n    s
1    1
2    2333
3    344
4    4
```

edited Oct 12 '18 at 17:10

Paul
1,963 1 10 18

answered Jun 5 '18 at 2:50

alkoln
464 5 12

I like this approach. I hope you don't mind, that I added some enhancement directly into your answer. Just feel free to edit this in any convenient way... – Shnugo Aug 14 '18 at 8:23

1

here is a version that can split on a pattern using patindex, a simple adaptation of the post above. I had a case where I needed to split a string that contained multiple separator chars.

```
alter FUNCTION dbo.splitstring ( @stringToSplit VARCHAR(10
)
RETURNS
 @returnList TABLE ([Name] [nvarchar] (500))
AS
BEGIN

 DECLARE @name NVARCHAR(255)
 DECLARE @pos INT
```

```
  SELECT @name = SUBSTRING(@stringToSplit, 1, @pos-1)

  INSERT INTO @returnList
  SELECT @name

  SELECT @stringToSplit = SUBSTRING(@stringToSplit, @pos+1
 END

 INSERT INTO @returnList
 SELECT @stringToSplit

 RETURN
END
select * from dbo.splitstring('stringa/stringb/x,y,z','%[/
```

result looks like this

stringa stringb x y z

answered Dec 5 '13 at 19:20

markgiaconia
2,800 3 13 34

1

Personnaly I use this function :

```
ALTER FUNCTION [dbo].[CUST_SplitString]
(
    @String NVARCHAR(4000),
    @Delimiter NCHAR(1)
)
RETURNS TABLE
AS
RETURN
(
    WITH Split(stpos,endpos)
    AS(
        SELECT 0 AS stpos, CHARINDEX(@Delimiter,@String) A
        UNION ALL
```

```
    )
    SELECT 'Id' = ROW_NUMBER() OVER (ORDER BY (SELECT 1)),
        'Data' = SUBSTRING(@String,stpos,COALESCE(NULLIF(e
stpos)
    FROM Split
)
```

answered Apr 24 '14 at 8:08

Kaly
44 8

1

I have developed a double Splitter (Takes two split characters) as requested Here. Could be of some value in this thread seeing its the most referenced for queries relating to string splitting.

```
CREATE FUNCTION uft_DoubleSplitter
(
    -- Add the parameters for the function here
    @String VARCHAR(4000),
    @Splitter1 CHAR,
    @Splitter2 CHAR
)
RETURNS @Result TABLE (Id INT,MId INT,SValue VARCHAR(4000)
AS
BEGIN
DECLARE @FResult TABLE(Id INT IDENTITY(1, 1),
                    SValue VARCHAR(4000))
DECLARE @SResult TABLE(Id INT IDENTITY(1, 1),
                    MId INT,
                    SValue VARCHAR(4000))
SET @String = @String+@Splitter1

WHILE CHARINDEX(@Splitter1, @String) > 0
    BEGIN
       DECLARE @WorkingString VARCHAR(4000) = NULL

       SET @WorkingString = SUBSTRING(@String, 1, CHARINDE
```

```
            SELECT CASE
                 WHEN @WorkingString = '' THEN NULL
                 ELSE @WorkingString
                 END

            SET @String = SUBSTRING(@String, LEN(@WorkingString

        END
    IF ISNULL(@Splitter2, '') != ''
        BEGIN
            DECLARE @OStartLoop INT
            DECLARE @OEndLoop INT

            SELECT @OStartLoop = MIN(Id),
                 @OEndLoop = MAX(Id)
            FROM @FResult

            WHILE @OStartLoop <= @OEndLoop
               BEGIN
                  DECLARE @iString VARCHAR(4000)
                  DECLARE @iMId INT

                  SELECT @iString = SValue+@Splitter2,
                        @iMId = Id
                  FROM @FResult
                  WHERE Id = @OStartLoop

                  WHILE CHARINDEX(@Splitter2, @iString) > 0
                     BEGIN
                         DECLARE @iWorkingString VARCHAR(4000)

                         SET @IWorkingString = SUBSTRING(@iStri
    @iString) - 1)

                         INSERT INTO @SResult
                         SELECT @iMId,
                              CASE
                              WHEN @iWorkingString = '' THEN NU
                              ELSE @iWorkingString
                              END

                         SET @iString = SUBSTRING(@iString, LEN
    LEN(@iString))

                     END
```

```
        SELECT MId AS PrimarySplitID,
             ROW_NUMBER() OVER (PARTITION BY MId ORDER BY M
             SValue
        FROM @SResult
    END
ELSE
    BEGIN
        INSERT INTO @Result
        SELECT Id AS PrimarySplitID,
             NULL AS SecondarySplitID,
             SValue
        FROM @FResult
    END
RETURN
```

Usage:

```
--FirstSplit
SELECT * FROM
uft_DoubleSplitter('ValueA=ValueB=ValueC=ValueD==ValueE&Va
```

```
--Second Split
SELECT * FROM
uft_DoubleSplitter('ValueA=ValueB=ValueC=ValueD==ValueE&Va
```

Possible Usage (Get second value of each split):

```
SELECT fn.SValue
FROM
uft_DoubleSplitter('ValueA=ValueB=ValueC=ValueD==ValueE&Va
 '&', '=')AS fn
WHERE fn.mid = 2
```

edited May 23 '17 at 12:26

Community ♦

Gouri Shankar Aechoor
1,430 1 5 8

1

The often used approach with XML elements breaks in case of forbidden characters. This is an approach to use this method with any kind of character, even with the semicolon as delimiter.

The trick is, first to use `SELECT SomeString AS [*] FOR XML PATH('')` to get all forbidden characters properly escaped. That's the reason, why I replace the delimiter to a *magic value* to avoid troubles with `;` as delimiter.

```
DECLARE @Dummy TABLE (ID INT, SomeTextToSplit NVARCHAR(MAX
INSERT INTO @Dummy VALUES
 (1,N'A&B;C;D;E, F')
,(2,N'"C" & ''D'';<C>;D;E, F');

DECLARE @Delimiter NVARCHAR(10)=';'; --special effort need
with "&code;")!

WITH Casted AS
(
    SELECT *
          ,CAST(N'<x>' + REPLACE((SELECT
REPLACE(SomeTextToSplit,@Delimiter,N'§§Split$me$here§§') A
PATH('')),N'§§Split$me$here§§',N'</x><x>') + N'</x>' AS XM
    FROM @Dummy
)
SELECT Casted.ID
      ,x.value(N'.',N'nvarchar(max)') AS Part
FROM Casted
CROSS APPLY SplitMe.nodes(N'/x') AS A(x)
```

The result

```
1   D
1   E, F
2   "C" & 'D'
2   <C>
2   D
2   E, F
```

answered Feb 2 '17 at 10:43

Shnugo
52k 7 27 76

0

This is more narrowly-tailored. When I do this I usually have a comma-delimited list of unique ids (INT or BIGINT), which I want to cast as a table to use as an inner join to another table that has a primary key of INT or BIGINT. I want an in-line table-valued function returned so that I have the most efficient join possible.

Sample usage would be:

```
DECLARE @IDs VARCHAR(1000);
SET @IDs = ',99,206,124,8967,1,7,3,45234,2,889,987979,';
SELECT me.Value
FROM dbo.MyEnum me
INNER JOIN dbo.GetIntIdsTableFromDelimitedString(@IDs) id
```

I stole the idea from http://sqlrecords.blogspot.com/2012/11/converting-delimited-list-to-table.html, changing it to be in-line table-valued and cast as INT.

```
create function dbo.GetIntIDTableFromDelimitedString
    (
    @IDs VARCHAR(1000)  --this parameter must start and en
```

```
RETURNS TABLE AS
 RETURN

SELECT
    CAST(SUBSTRING(@IDs,Nums.number + 1,CHARINDEX(',',@IDs
Nums.number - 1) AS INT) AS ID
FROM
     [master].[dbo].[spt_values] Nums
WHERE Nums.Type = 'P'
AND    Nums.number BETWEEN 1 AND DATALENGTH(@IDs)
AND    SUBSTRING(@IDs,Nums.number,1) = ','
AND    CHARINDEX(',',@IDs,(Nums.number+1)) > Nums.number;

GO
```

answered Nov 14 '13 at 21:11

Tom Regan

1,466 1 23 40

0

```
ALTER FUNCTION [dbo].func_split_string
(
    @input as varchar(max),
    @delimiter as varchar(10) = ";"

)
RETURNS @result TABLE
(
    id smallint identity(1,1),
    csv_value varchar(max) not null
)
AS
BEGIN
    DECLARE @pos AS INT;
    DECLARE @string AS VARCHAR(MAX) = '';

    WHILE LEN(@input) > 0
    BEGIN
        SELECT @pos = CHARINDEX(@delimiter,@input);
```

```
        IF(@pos <> LEN(@input))
            SELECT @string = SUBSTRING(@input, 1, @pos-1);
        ELSE
            SELECT @string = SUBSTRING(@input, 1, @pos);

        INSERT INTO @result SELECT @string

        SELECT @input = SUBSTRING(@input, @pos+len(@delimi
    END
    RETURN
END
```

edited Jan 2 '14 at 15:28

Community ♦

1 1

answered Oct 11 '13 at 12:28

MiH

1 1

0

You can Use this function:

```
CREATE FUNCTION SplitString
(
   @Input NVARCHAR(MAX),
   @Character CHAR(1)
  )
    RETURNS @Output TABLE (
    Item NVARCHAR(1000)
  )
AS
BEGIN

DECLARE @StartIndex INT, @EndIndex INT
SET @StartIndex = 1
IF SUBSTRING(@Input, LEN(@Input) - 1, LEN(@Input)) <
BEGIN
```

```
        WHILE CHARINDEX(@Character, @Input) > 0
        BEGIN
              SET @EndIndex = CHARINDEX(@Character, @Input)

              INSERT INTO @Output(Item)
              SELECT SUBSTRING(@Input, @StartIndex, @EndInde

              SET @Input = SUBSTRING(@Input, @EndIndex + 1,
        END

        RETURN
  END
  GO
```

edited Jul 1 '15 at 22:51

ZygD
4,414 11 30 54

answered Jul 1 '15 at 22:23

Abhinav
24 1 1 8

0

Here is an example that you can use as function or also you can put the same logic in procedure. -- SELECT * from [dbo].fn_SplitString ;

```
CREATE FUNCTION [dbo].[fn_SplitString]
(@CSV VARCHAR(MAX), @Delimeter VARCHAR(100) = ',')
       RETURNS @retTable TABLE
(

    [value] VARCHAR(MAX) NULL
)AS

BEGIN

DECLARE
```

```
IF @vDelimeter = ';'
BEGIN
    SET @vCSV = REPLACE(@vCSV, ';', '~!~#~');
    SET @vDelimeter = REPLACE(@vDelimeter, ';', '~!~#~');
END;

SET @vCSV = REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(@vCSV,
'>', '&gt;'), '''', '&apos;'), '"', '&quot;');

DECLARE @xml XML;

SET @xml = '<i>' + REPLACE(@vCSV, @vDelimeter, '</i><i>') +

INSERT INTO @retTable
SELECT
       x.i.value('.', 'varchar(max)') AS COLUMNNAME
  FROM @xml.nodes('//i')AS x(i);

 RETURN;
END;
```

answered Jun 24 '16 at 12:32

Surya
117 3

This approach will break if `@vCSV` contains forbidden characters... I just posted an answer to overcome this issue. – Shnugo Feb 2 '17 at 10:45

0

/*

Answer to T-SQL split string
Based on answers from Andy Robinson and AviG
Enhanced functionality ref: LEN function not including trailing spaces in SQL Server
This 'file' should be valid as both a markdown file and

```
```

*/

```
CREATE FUNCTION dbo.splitstring ( --CREATE OR ALTER
    @stringToSplit NVARCHAR(MAX)
) RETURNS @returnList TABLE ([Item] NVARCHAR (MAX))
AS BEGIN
    DECLARE @name NVARCHAR(MAX)
    DECLARE @pos BIGINT
    SET @stringToSplit = @stringToSplit + ','
that end with a `,` to have a blank value in that "column"
    WHILE ((LEN(@stringToSplit+'_') > 1)) BEGIN
trimming terminal spaces. See URL referenced above
        SET @pos = COALESCE(NULLIF(CHARINDEX(',',
@stringToSplit),0),LEN(@stringToSplit+'_')) -- COALESCE gr
        SET @name = SUBSTRING(@stringToSplit, 1, @pos-1)
nvarchar is 4000
        SET @stringToSplit = SUBSTRING(@stringToSplit, @po
fn (MS web): "If start is greater than the number of chara
a zero-length expression is returned."
        INSERT INTO @returnList SELECT @name --additional
be added
        -- + ' pos:' + CAST(@pos as nvarchar) + ' remain:'
CAST(LEN(@stringToSplit+'_')-1 as nvarchar) + ')'
    END
    RETURN
END
GO
```

/*

```
```

Test cases: see URL referenced as "enhanced functionality" above

```
SELECT *,LEN(Item+'_')-1 'L' from splitstring('a,,b')
```

```
Item | L
---  | ---
a    | 1
```

```
SELECT *,LEN(Item+'_')-1 'L' from splitstring('a,,')


 Item | L
 ---  / ---
 a    | 1
      | 0
      | 0


SELECT *,LEN(Item+'_')-1 'L' from splitstring('a,, ')


 Item | L
 ---  / ---
 a    | 1
      | 0
      | 1


SELECT *,LEN(Item+'_')-1 'L' from splitstring('a,, c ')


 Item | L
 ---  / ---
 a    | 1
      | 0
  c   | 3

*/
```

edited May 23 '17 at 11:47

Community ♦
1 1

answered Aug 11 '16 at 20:57

mpag
211 2 14

0

```
declare @T table (iden int identity, col1 varchar(100));
insert into @T(col1) values
       ('ROOT/South America/Lima/Test/Test2')
     , ('ROOT/South America/Peru/Test/Test2')
     , ('ROOT//South America/Venuzuala ')
     , ('RtT/South America / ')
     , ('ROOT/South Americas// ');
declare @split char(1) = '/';
select @split as split;
with cte as
(  select t.iden, case when SUBSTRING(REVERSE(rtrim(t.col1
LTRIM(RTRIM(t.col1)) else LTRIM(RTRIM(t.col1)) + @split en
, 1 as cnt
   from @T t
   union all
   select t.iden, t.col1
, charindex(@split, t.col1, t.pos + 1), cnt + 1
   from cte t
   where charindex(@split, t.col1, t.pos + 1) > 0
)
select t1.*, t2.pos, t2.cnt
     , ltrim(rtrim(SUBSTRING(t1.col1, t1.pos+1, t2.pos-t1.
from cte t1
join cte t2
  on t2.iden = t1.iden
 and t2.cnt  = t1.cnt+1
 and t2.pos > t1.pos
order by t1.iden, t1.cnt;
```

answered Mar 13 '18 at 21:21

paparazzo
38.1k 17 78 141

With all due respect to @AviG this is the bug free version of function deviced by him to return all the tokens in full.

0

```
IF EXISTS (SELECT * FROM sys.objects WHERE type = 'TF' AND
```

```
-- ============================================
-- Author:  AviG
-- Amendments:  Parameterize the delimeter and included th
- Gemunu Wickremasinghe
-- Description: Tabel valued function that Breaks the deli
delimeter and returns a tabel having split results
-- Usage
-- select * from   [dbo].[TF_SplitString]('token1,token2,,
-- 969 items should be returned
-- select * from   [dbo].[TF_SplitString]('4672978261,4672
-- 2 items should be returned
-- ============================================
CREATE FUNCTION dbo.TF_SplitString
( @stringToSplit VARCHAR(MAX) ,
  @delimeter char = ','
)
RETURNS
 @returnList TABLE ([Name] [nvarchar] (500))
AS
BEGIN

    DECLARE @name NVARCHAR(255)
    DECLARE @pos INT

    WHILE LEN(@stringToSplit) > 0
    BEGIN
        SELECT @pos  = CHARINDEX(@delimeter, @stringToSpli


        if @pos = 0
        BEGIN
            SELECT @pos = LEN(@stringToSplit)
            SELECT @name = SUBSTRING(@stringToSplit, 1, @p
        END
        else
        BEGIN
            SELECT @name = SUBSTRING(@stringToSplit, 1, @p
        END

        INSERT INTO @returnList
        SELECT @name

        SELECT @stringToSplit = SUBSTRING(@stringToSplit, 
@pos)
    END
```

edited Jun 23 '18 at 9:24

answered Jun 23 '18 at 9:16

Gemunu R Wickremasinghe

99 5

0

This is based on Andy Robertson's answer, I needed a delimiter other than comma.

```
CREATE FUNCTION dbo.splitstring ( @stringToSplit nvarchar(
RETURNS
 @returnList TABLE ([value] [nvarchar] (MAX))
AS
BEGIN

 DECLARE @value NVARCHAR(max)
 DECLARE @pos INT

 WHILE CHARINDEX(@delim, @stringToSplit) > 0
 BEGIN
  SELECT @pos  = CHARINDEX(@delim, @stringToSplit)
  SELECT @value = SUBSTRING(@stringToSplit, 1, @pos - 1)

  INSERT INTO @returnList
  SELECT @value

  SELECT @stringToSplit = SUBSTRING(@stringToSplit, @pos +
LEN(@stringToSplit) - @pos)
 END

 INSERT INTO @returnList
 SELECT @stringToSplit

 RETURN
END
```

And to use it:

```
SELECT * FROM dbo.splitstring('test1 test2 test3', ' ');
```

(Tested on SQL Server 2008 R2)

EDIT: correct test code

answered Jan 25 at 21:00

Nunzio Tocci
178 1 9

-3

The easiest way:

1. Install SQL Server 2016
2. Use STRING_SPLIT https://msdn.microsoft.com/en-us/library/mt684588.aspx

It works even in express edition :).

answered Mar 8 '16 at 21:17

Jovan MSFT
7,550 2 25 34

Don't forget to set "Compatibility level" to SQL Server 2016 (130) - in management studio, right click on database, properties / options / compatibility level. – Tomino Nov 9 '16 at 10:17

The original post said for SQL 2008 R2. Installing SQL 2016 may not be an option – Shawn Gavett Jan 9 '17 at 17:34

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?