Menu

Home
Articles
Tips
Blogs
Quiz
Scripts
Links

Converting row values in a table to a single concatenated string

Category : Articles User Rating : Views :

I recently had to look at a problem that required converting the column values from a set of rows into a single comma separated string. On the face of it this is quite straightforward, but as with many things SQL Server there is more than one way to solve the problem. I've presented a few possible solutions here, no doubt there are more.

Let's get down to the problem. I'll create a table which can be used to demonstrate this (for the purposes of simplicity I've removed most of the columns).

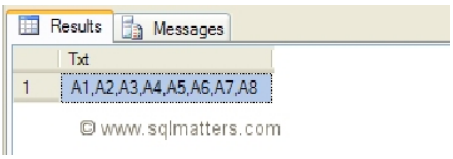
Here is the table creation script :

```
CREATE TABLE ConcatenationDemo
(
 RowID    INT PRIMARY KEY
,Txt      VARCHAR(MAX)
)
```

and here is the data (as a SQL script) :

```
INSERT INTO ConcatenationDemo
(RowID,Txt)
      SELECT 1,'A1'
UNION SELECT 2,'A2'
UNION SELECT 3,'A3'
UNION SELECT 4,'A4'
UNION SELECT 5,'A5'
UNION SELECT 6,'A6'
UNION SELECT 7,'A7'
UNION SELECT 8,'A8'
```

The requirement is to create a query that will concatenate the values in the 'Txt' column into a comma separated list. The query needs to return one row and one column so that the results pane in SQL Server Management Studio looks like this :



Solution 1 : Using a Recursive CTE

SQL Server 2005 opened up the possibility of using recursion within a SQL statement. Here is a recursive CTE that produces the required result :

```
;WITH CTE_Concatenated AS
    (
    SELECT  RowID,
            Txt
    FROM    ConcatenationDemo
    WHERE   RowID = 1
    UNION ALL
    SELECT  ConcatenationDemo. RowID
           ,CTE_Concatenated.Txt + ',' + ConcatenationDemo.Txt
    FROM    CTE_Concatenated
    JOIN    ConcatenationDemo
    ON      ConcatenationDemo. RowID =CTE_Concatenated.RowID + 1
    )
```

```
    SELECT  Txt
    FROM    CTE_Concatenated
    WHERE   RowID = (SELECT MAX(RowID) FROM ConcatenationDemo)
```

If I run this query then I get the result I want : The first part of the CTE is the anchor that sets the initial condition and the second part (after the UNION ALL) is the recursive part. I won't explain this further here, however there is an excellent article about this subject on MSDN titled **'Recursive Queries Using Common Table Expressions'**

Solution 2 : Using PIVOT

The first instinct when confronted with a problem which involves converting rows to columns may be to use PIVOT, and indeed the following code uses the PIVOT operator. It returns the same result as that above :

```
SELECT [1]+','+[2]+','+[3]+','+[4]+','+[5]+','+[6]+','+[7]+','+[8] AS Txt
FROM  (SELECT 'Total' AS AC,[1],[2],[3],[4],[5],[6],[7],[8]
       FROM
          (SELECT RowID,Txt FROM ConcatenationDemo) AS B
       PIVOT (MAX(Txt) FOR RowID IN ([1],[2],[3],[4],[5],[6],[7],[8])
             ) AS A
      ) AS C
```

I've used the PIVOT command to convert the rows to an equivalent number of columns and then concatenated the result together. Normally we would have some type of aggregation as the rows are pivoted to columns, hence the use of the MAX() here. Note that in the this example no aggregation is necessary so this could equally well be a MIN(). One thing that you may have noticed is that the code is specific for the number of rows in the table – the code would have to be modified if there was an extra row. This is a significant limitation of this approach.

Solution 3 : Using a WHILE loop or Cursor

In the procedural world the chosen solution would probably be some sort of looping construct. We can do the same thing in SQL Server using either a while loop or cursor. The following code uses a WHILE loop (this could be rewritten to use a cursor) :

```
DECLARE @MaxCount INTEGER
DECLARE @Count INTEGER
DECLARE @Txt VARCHAR(MAX)
SET @Count = 1
SET @Txt = ''
SET @MaxCount = (SELECT MAX(RowID) FROM ConcatenationDemo)
WHILE @Count<=@MaxCount
    BEGIN
    IF @Txt!=''
        SET @Txt=@Txt+',' + (SELECT Txt FROM ConcatenationDemo WHERE RowID=@Count)
    ELSE
        SET @Txt=(SELECT Txt FROM ConcatenationDemo WHERE RowID=@Count)
    SET @Count=@Count+1
    END
SELECT @Txt AS Txt
```

Generally looping constructs in SQL are to be avoided as they can perform badly. However if the number of rows is small, as it is here, then this can be a useful approach. This solution is perhaps the easiest to understand and the most flexible, where performance isn't an issue.

Solution 4 : Using SQL Concatenation

This solution is perhaps the most surprising, in that at first glance you might not expect it to work. The SQL below simply creates a variable then concatenates the value of the row. SQL iterates around each row in the table to produce the result. I adapted this solution from some code I found on a newsgroup a while ago, and was astonished that it worked. This behaviour is certainly unexpected, and also appears to be undocumented. As such it comes with a "health warning" as it maybe that Microsoft will remove the ability to do this in the future without warning. However for non-production code it's a very simple solution.

```
DECLARE @Txt1 VARCHAR(MAX)
SET @Txt1=''

SELECT  @Txt1 = @Txt1 + Txt +','
FROM    ConcatenationDemo
SELECT  LEFT(@Txt1,LEN(@Txt1)-1) AS Txt
```

Solution 5 : Using FOR XML PATH

Some of the XML statements introduced in SQL Server 2005 had to implement a means of looping around data in order to produce XML. This solution takes advantage of this, but strips out the XML specific parts to produce the comma separated list.

```
SELECT STUFF((SELECT ',' + Txt
            FROM ConcatenationDemo
            FOR XML PATH('')) ,1,1,'') AS Txt
```

Solution 6 : Using the CLR

SQL Server 2005 introduced the ability to write logic using procedural code in C# or other dot net languages using the CLR. As the dot net framework provides richer capabilities for implementing logic such as string handling this is another possible way of implementing a solution.

Conclusion

I've given six possible solutions here. Each has its own merits and may be useful in differing circumstances. It's likely that a real life requirement would be more complex than the simple example given here, so it wouldn't be sensible to give a recommendation for which approach to adopt as this will vary according to the precise requirements. However the solution chosen is likely to be based on the performance required, code clarity and maintainability.

Related Articles

The following articles may also be of interest :

- **Converting row values in a table to a single concatenated string (this article)**
- **Using SQL to convert row values to a single concatenated string**

Link back to this article : **http://www.sqlmatters.com/Articles/Converting row values in a table to a single concatenated string.aspx**

Keywords

TSQL,Row concatenation,table,pivot

Comments

Post by Deep on Thu 18 Oct 2012 13:40. Report Inappropriate Post

Hi, I have a requirement like.... Table : Employee Row [0] : column [0] : has value = "Server" Do something so in Table it will insert like S e r v e r

Post by Bob Murray on Wed 24 Oct 2012 15:15. Report Inappropriate Post

This post is fantastic - thank you!

Post by Nike on Thu 10 Jan 2013 16:51. Report Inappropriate Post

Hi, i want to get same multiple row from 1 row in table sql. table "schedule" Name Hospital Visit ---------- Ana RSIA 3 Budi RSIB 2 i want to create a query to get the result like : Ana RSIA 3 Ana RSIA 3 Ana RSIA 3 Budi RSIB 2 Budi RBIB 2 Need your help to create a query in sql sever. Thanks

Post by wajira on Fri 19 Apr 2013 10:05. Report Inappropriate Post

Thank you very much. Saved my day

Post by Bob on Wed 09 Oct 2013 21:26. Report Inappropriate Post

Exceptional article that continues to save some of us overworked developers TONS of time! Thanks So Much!

Post by Daniel on Wed 23 Oct 2013 11:27. Report Inappropriate Post

Excelent article!!! Thanks a lot!!!

Post by Peter Sidi on Fri 10 Jan 2014 20:59. Report Inappropriate Post

Way to go!!!

Post a comment No login required !

Enter your comment here

Name : *Your name* *Will be displayed alongside your comment*

Email : *Your email address* *Not displayed*

Website : *e.g. http://www.mysite.com* *Optional, but displayed if entered*

Submit Comment

Notify me if more comments are added