Convert multiple rows into one with comma as separator [duplicate]

Ask Question

68

26

This question already has an answer here:
How to concatenate text from multiple rows into a single text string in SQL server? *45 answers*

If I issue `SELECT username FROM Users` I get this result:

```
username
--------
Paul
John
Mary
```

but what I really need is *one* row with all the values separated by comma, like this:

```
Paul, John, Mary
```

How do I do this?

tsql rows csv

edited May 20 '09 at 16:07

rjstelling
17.7k 26 97 173

3,271 6 31 47

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

marked as duplicate by Community ♦ Oct 23 '15 at 5:56

This question has been asked before and already has an answer. If those answers do not fully address your question, please ask a new question.

10 Answers

80

This should work for you. Tested all the way back to SQL 2000.

```
create table #user (username varchar(25))

insert into #user (username) values ('Paul')
insert into #user (username) values ('John')
insert into #user (username) values ('Mary')

declare @tmp varchar(250)
SET @tmp = ''
select @tmp = @tmp + username + ', ' from #user

select SUBSTRING(@tmp, 0, LEN(@tmp))
```

answered May 20 '09 at 12:44

mwigdahl
13.6k 4 44 60

1 +1, but `select SUBSTRING(@tmp, 0, LEN(@tmp))` looks incorrect (to me) while apparently working (I tried it). The...turgid...prose of the MSDN page on `substring` fails to clarify why it works, but I guess the end point is `start_expression + length_expression` without correcting

`SUBSTRING(@tmp, 1, LEN(@tmp) - 1)` instead, though. – T.J. Crowder Aug 7 '11 at 11:53

Yeah, apparently, since `select SUBSTRING('testing', -2, 5)` gives us `'te'` (e.g., exactly what `select SUBSTRING('testing', 1, 2)` would give us), as in both cases the resulting (exclusive) end index is `3`. Not behavior I'd want to rely on. Is there some specific reason you do? – T.J. Crowder Aug 7 '11 at 11:57

No special reason, just the fact that my roots are in C++ so I'm used to zero-offset arithmetic... – mwigdahl Aug 7 '11 at 14:36

1 @BoratSagdiyev -- You are correct, that's exactly what it's doing. That syntax expands into what is effectively a mini-cursor. In general, the FOR XML PATH solution Hogan provides below is faster for large cases and if you don't care about XML escaping of your special characters. – mwigdahl Apr 18 '14 at 20:11

3 SQL server 2017/ SQL Azure supports STRING_AGG for the purpose - docs.microsoft.com/en-us/sql/t-sql/functions/… stackoverflow.com/a/42778050/1545569 – Chirag Rupani Dec 25 '17 at 11:13

104

```
select
  distinct
   stuff((
       select ',' + u.username
       from users u
       where u.username = username
       order by u.username
       for xml path('')
   ),1,1,'') as userlist
from users
group by username
```

had a typo before. the above works

Hogan
56.1k 9 67 103

2 You are a genius. Also, this puts one space in front of the list. Make the STUFF options 1, 2 and it'll remove that space. I need to figure out how the hell this works, now. – Jared Mar 28 '13 at 12:58

3 Disclaimer: this **will not work** for data that has special characters such as `<` or `&` in it. – Kehlan Krumme Apr 17 '14 at 19:19

6 @BoratSagdiyev - kehrk is wrong, `FOR XML` auto encodes, see sqlfiddle.com/#!6/b824a/1 – Hogan Apr 20 '14 at 19:21

4 @kehrk - ummm... no. Works fine. sqlfiddle.com/#!6/b824a/1 – Hogan Apr 20 '14 at 19:22

1 Should be the accepted answer – user9168386 Nov 29 '18 at 13:53

42

good review of several approaches:

http://blogs.msmvps.com/robfarley/2007/04/07/coalesce-is-not-the-answer-to-string-concatentation-in-t-sql/

Article copy -

Coalesce is not the answer to string concatentation in T-SQL I've seen many posts over the years about using the COALESCE function to get string concatenation working in T-SQL. This is one of the examples here (borrowed from Readifarian Marc Ridey).

```
DECLARE @categories varchar(200)
SET @categories = NULL
```

```
SELECT @categories
```

This query can be quite effective, but care needs to be taken, and the use of COALESCE should be properly understood. COALESCE is the version of ISNULL which can take more than two parameters. It returns the first thing in the list of parameters which is not null. So really it has nothing to do with concatenation, and the following piece of code is exactly the same - without using COALESCE:

```
DECLARE @categories varchar(200)
SET @categories = ''

SELECT @categories = @categories + ',' + Name
FROM Production.ProductCategory

SELECT @categories
```

But the unordered nature of databases makes this unreliable. The whole reason why T-SQL doesn't (yet) have a concatenate function is that this is an aggregate for which the order of elements is important. Using this variable-assignment method of string concatenation, you may actually find that the answer that gets returned doesn't have all the values in it, particularly if you want the substrings put in a particular order. Consider the following, which on my machine only returns ',Accessories', when I wanted it to return ',Bikes,Clothing,Components,Accessories':

```
DECLARE @categories varchar(200)
SET @categories = NULL

SELECT @categories = COALESCE(@categories + ',','') + Name
FROM Production.ProductCategory
ORDER BY LEN(Name)
```

Far better is to use a method which does take order into consideration, and which has been included in SQL2005 specifically for the purpose of string concatenation - FOR XML PATH('')

```
SELECT ',' + Name
FROM Production.ProductCategory
ORDER BY LEN(Name)
FOR XML PATH('')
```

In the post I made recently comparing GROUP BY and DISTINCT when using subqueries, I demonstrated the use of FOR XML PATH(''). Have a look at this and you'll see how it works in a subquery. The 'STUFF' function is only there to remove the leading comma.

```
USE tempdb;
GO
CREATE TABLE t1 (id INT, NAME VARCHAR(MAX));
INSERT t1 values (1,'Jamie');
INSERT t1 values (1,'Joe');
INSERT t1 values (1,'John');
INSERT t1 values (2,'Sai');
INSERT t1 values (2,'Sam');
GO

select
    id,
    stuff((
        select ',' + t.[name]
        from t1 t
        where t.id = t1.id
        order by t.[name]
        for xml path('')
    ),1,1,'') as name_csv
from t1
group by id
;
```

FOR XML PATH is one of the only situations in which you

tags. This does mean that the strings will be HTML Encoded, so if you're concatenating strings which may have the < character (etc), then you should maybe fix that up afterwards, but either way, this is still the best way of concatenating strings in SQL Server 2005.

edited Oct 31 '16 at 20:22

Brian
18.6k 13 66 152

answered May 20 '09 at 16:06

A-K
14.1k 5 46 66

1 Alex, I added the whole article in case original link goes dead. I hope you will accept the changes. Thanks. Chenqui. – Borat Sagdiyev Apr 18 '14 at 18:46

1 @AlexKuznetsov - FYI - This article is over 7 years old and contains information which has not been true since SQL 2008 came out. – Hogan Apr 20 '14 at 19:27

@Hogan the answer is very old as well, and it was written at the time when 2008 was not widely adopted at all. I see no value in keeping this answer up-to-date. – A-K Apr 21 '14 at 15:46

@AlexKuznetsov the last one worked me to use in an inline query. thanks for the details answer. – Ram May 30 '14 at 10:36

1 +1 for your use of the FOR XML('') version, I was struggling to get aggregated rows partitioned by the id. Your example not only fixed it, but I understand how it works. – Morvael Jun 2 '17 at 11:02

10

```
group, csv
'group1', 'paul, john'
'group2', 'mary'

    --drop table #user
create table #user (groupName varchar(25), username varcha

insert into #user (groupname, username) values ('apostles'
insert into #user (groupname, username) values ('apostles'
insert into #user (groupname, username) values ('family','

select
    g1.groupname
    , stuff((
        select ', ' + g.username
        from #user g
        where g.groupName = g1.groupname
        order by g.username
        for xml path('')
    ),1,2,'') as name_csv
from #user g1
group by g1.groupname
```

answered Jul 17 '15 at 14:56

Tom McDonald
487 6 20

6

```
DECLARE @EmployeeList varchar(100)

SELECT @EmployeeList = COALESCE(@EmployeeList + ', ', '') +
   CAST(Emp_UniqueID AS varchar(5))
FROM SalesCallsEmployees
WHERE SalCal_UniqueID = 1

SELECT @EmployeeList
```

source: http://www.sqlteam.com/article/using-coalesce-to-

edited Sep 17 '11 at 10:35

LPL
14.6k 5 34 73

answered Jan 12 '11 at 18:50

vinay
61 1 1

6

You can use this query to do the above task:

```
DECLARE @test NVARCHAR(max)
SELECT @test = COALESCE(@test + ',', '') + field2 FROM #te
SELECT field2 = @test
```

For detail and step by step explanation visit the following link http://oops-solution.blogspot.com/2011/11/sql-server-convert-table-column-data.html

edited Nov 4 '11 at 22:22

LPL
14.6k 5 34 73

answered Nov 4 '11 at 11:54

Rashmi Kant
143 1 2

4

In SQLite this is simpler. I think there are similar implementations for MySQL, MSSql and Orable

```
CREATE TABLE Beatles (id integer, name string );
```

```
INSERT INTO Beatles VALUES (4, "George");
SELECT GROUP_CONCAT(name, ',') FROM Beatles;
```

answered Feb 13 '12 at 7:41

elcuco
5,928 7 34 58

3

A clean and flexible solution in MS SQL Server 2005/2008 is to create a CLR Agregate function.

You'll find quite a few articles (with code) on google.

It looks like this article walks you through the whole process using C#.

answered May 20 '09 at 19:15

Arjan Einbu
11.9k 1 48 56

2

you can use stuff() to convert rows as comma separated values

```
select
EmployeeID,
stuff((
  SELECT ',' + FPProjectMaster.GroupName
      FROM     FPProjectInfo AS t INNER JOIN
              FPProjectMaster ON t.ProjectID = FPProjectMa
      WHERE  (t.EmployeeID = FPProjectInfo.EmployeeID)
              And t.STatusID = 1
              ORDER BY t.ProjectID
       for xml path('')
       ),1,1,'') as name_csv
from FPProjectInfo
```

Thanks @AlexKuznetsov for the reference to get this answer.

answered May 30 '14 at 10:44

Ram

10.4k 3 28 33

If you're executing this through PHP, what about this?

-4

```
$hQuery = mysql_query("SELECT * FROM users");
while($hRow = mysql_fetch_array($hQuery)) {
    $hOut .= $hRow['username'] . ", ";
}
$hOut = substr($hOut, 0, strlen($hOut) - 1);
echo $hOut;
```

answered May 20 '09 at 12:49

James Brooks

703 3 14 28

Oh my bad, it seems your running this through the console. – James Brooks May 20 '09 at 12:50

I need this done in sql, not in php or whatever (I'm using c# actually) – Pavel Bastov May 21 '09 at 2:23

Yeah I noticed it wasn't PHP. – James Brooks May 21 '09 at 9:37