**Grouped Concatenation in SQL Server**

August 6, 2014 by [Aaron Bertrand](https://sqlperformance.com/author/abertrand) in [T-SQL Queries](https://sqlperformance.com/category/t-sql-queries) | [22 Comments](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comments)

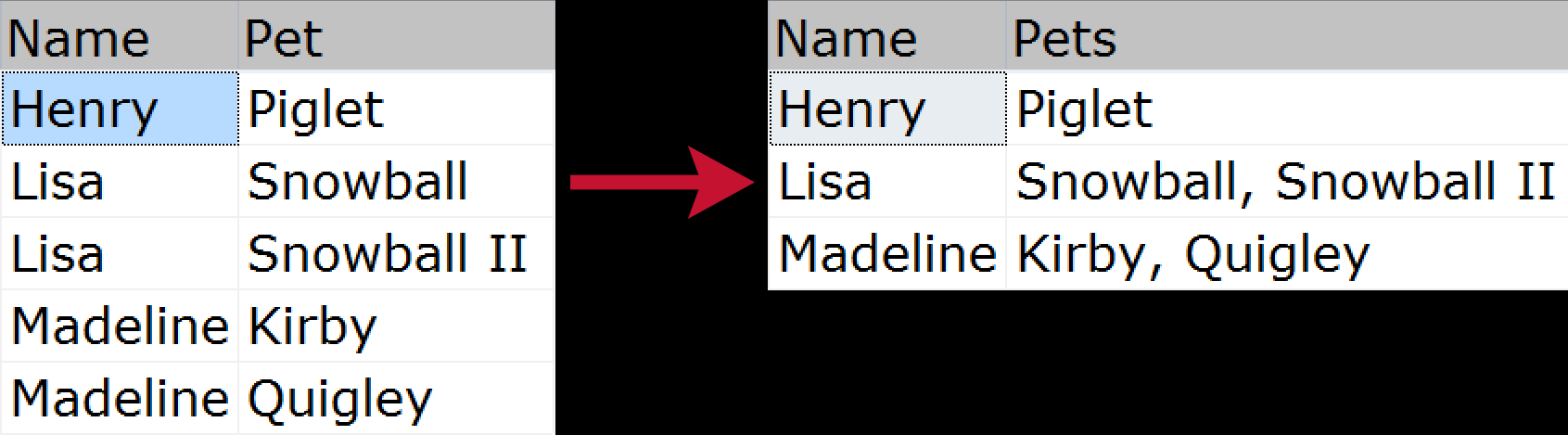
Grouped concatenation is a common problem in SQL Server, with no direct and intentional features to support it (like XMLAGG in Oracle, STRING\_AGG or ARRAY\_TO\_STRING(ARRAY\_AGG()) in PostgreSQL, and GROUP\_CONCAT in MySQL). It has been requested, but no success yet, as evidenced in these Connect items:

* [Connect #247118 : SQL needs version of MySQL group\_Concat function](http://connect.microsoft.com/SQLServer/feedback/details/247118/sql-needs-version-of-mysql-group-concat-function) (Postponed)
* [Connect #728969 : Ordered Set Functions – WITHIN GROUP Clause](http://connect.microsoft.com/SQLServer/feedback/details/728969/feature-request-ordered-set-functions-within-group-clause) (Closed as Won't Fix)

**\*\* UPDATE January 2017 \*\***: STRING\_AGG() will be in SQL Server 2017; read about it [here](https://sqlperformance.com/2016/12/sql-performance/sql-server-v-next-string_agg-performance), [here](https://sqlperformance.com/2017/01/sql-performance/sql-server-v-next-string_agg-performance-part-2), and [here](https://www.mssqltips.com/sqlservertip/5275/solve-old-problems-with-sql-servers-new-stringagg-and-stringsplit-functions/).

**What is Grouped Concatenation?**

For the uninitiated, grouped concatenation is when you want to take multiple rows of data and compress them into a single string (usually with delimiters like commas, tabs, or spaces). Some might call this a "horizontal join." A quick visual example demonstrating how we would compress a list of pets belonging to each family member, from the normalized source to the "flattened" output:

[](https://sqlperformance.com/wp-content/uploads/2014/08/gcat1.png)

There have been many ways to solve this problem over the years; here are just a few, based on the following sample data:

CREATE TABLE dbo.FamilyMemberPets

(

Name SYSNAME,

Pet SYSNAME,

PRIMARY KEY(Name,Pet)

);

INSERT dbo.FamilyMemberPets(Name,Pet) VALUES

(N'Madeline',N'Kirby'),

(N'Madeline',N'Quigley'),

(N'Henry', N'Piglet'),

(N'Lisa', N'Snowball'),

(N'Lisa', N'Snowball II');

I am not going to demonstrate an exhaustive list of every grouped concatenation approach ever conceived, as I want to focus on a few aspects of my recommended approach, but I do want to point out a few of the more common ones:

**Scalar UDF**

CREATE FUNCTION dbo.ConcatFunction

(

@Name SYSNAME

)

RETURNS NVARCHAR(MAX)

WITH SCHEMABINDING

AS

BEGIN

DECLARE @s NVARCHAR(MAX);

SELECT @s = COALESCE(@s + N', ', N'') + Pet

FROM dbo.FamilyMemberPets

WHERE Name = @Name

ORDER BY Pet;

RETURN (@s);

END

GO

SELECT Name, Pets = dbo.ConcatFunction(Name)

FROM dbo.FamilyMemberPets

GROUP BY Name

ORDER BY Name;

Note: there is a reason we don't do this:

SELECT DISTINCT Name, Pets = dbo.ConcatFunction(Name)

FROM dbo.FamilyMemberPets

ORDER BY Name;

With DISTINCT, the function is run for every single row, then duplicates are removed; with GROUP BY, the duplicates are removed first.

**Common Language Runtime (CLR)**

This uses the GROUP\_CONCAT\_S function found at <http://groupconcat.codeplex.com/>:

SELECT Name, Pets = dbo.GROUP\_CONCAT\_S(Pet, 1)

FROM dbo.FamilyMemberPets

GROUP BY Name

ORDER BY Name;

**Recursive CTE**

There are several variations on this recursion; this one pulls out a set of distinct names as the anchor:

;WITH x as

(

SELECT Name, Pet = CONVERT(NVARCHAR(MAX), Pet),

r1 = ROW\_NUMBER() OVER (PARTITION BY Name ORDER BY Pet)

FROM dbo.FamilyMemberPets

),

a AS

(

SELECT Name, Pet, r1 FROM x WHERE r1 = 1

),

r AS

(

SELECT Name, Pet, r1 FROM a WHERE r1 = 1

UNION ALL

SELECT x.Name, r.Pet + N', ' + x.Pet, x.r1

FROM x INNER JOIN r

ON r.Name = x.Name

AND x.r1 = r.r1 + 1

)

SELECT Name, Pets = MAX(Pet)

FROM r

GROUP BY Name

ORDER BY Name

OPTION (MAXRECURSION 0);

**Cursor**

Not much to say here; cursors are usually not the optimal approach, but this may be your only choice if you are stuck on SQL Server 2000:

DECLARE @t TABLE(Name SYSNAME, Pets NVARCHAR(MAX),

PRIMARY KEY (Name));

INSERT @t(Name, Pets)

SELECT Name, N''

FROM dbo.FamilyMemberPets GROUP BY Name;

DECLARE @name SYSNAME, @pet SYSNAME, @pets NVARCHAR(MAX);

DECLARE c CURSOR LOCAL FAST\_FORWARD

FOR SELECT Name, Pet

FROM dbo.FamilyMemberPets

ORDER BY Name, Pet;

OPEN c;

FETCH c INTO @name, @pet;

WHILE @@FETCH\_STATUS = 0

BEGIN

UPDATE @t SET Pets += N', ' + @pet

WHERE Name = @name;

FETCH c INTO @name, @pet;

END

CLOSE c; DEALLOCATE c;

SELECT Name, Pets = STUFF(Pets, 1, 1, N'')

FROM @t

ORDER BY Name;

GO

**Quirky Update**

Some people \*love\* this approach; I don't comprehend the attraction at all.

DECLARE @Name SYSNAME, @Pets NVARCHAR(MAX);

DECLARE @t TABLE(Name SYSNAME, Pet SYSNAME, Pets NVARCHAR(MAX),

PRIMARY KEY (Name, Pet));

INSERT @t(Name, Pet)

SELECT Name, Pet FROM dbo.FamilyMemberPets

ORDER BY Name, Pet;

UPDATE @t SET @Pets = Pets = COALESCE(

CASE COALESCE(@Name, N'')

WHEN Name THEN @Pets + N', ' + Pet

ELSE Pet END, N''),

@Name = Name;

SELECT Name, Pets = MAX(Pets)

FROM @t

GROUP BY Name

ORDER BY Name;

**FOR XML PATH**

Quite easily my preferred method, at least in part because it is the only way to \*guarantee\* order without using a cursor or CLR. That said, this is a very raw version that fails to address a couple of other inherent problems I will discuss further on:

SELECT Name, Pets = STUFF((SELECT N', ' + Pet

FROM dbo.FamilyMemberPets AS p2

WHERE p2.name = p.name

ORDER BY Pet

FOR XML PATH(N'')), 1, 2, N'')

FROM dbo.FamilyMemberPets AS p

GROUP BY Name

ORDER BY Name;

I've seen a lot of people mistakenly assume that the new [CONCAT() function introduced in SQL Server 2012](http://msdn.microsoft.com/en-us/library/hh231515(v=sql.110).aspx) was the answer to these feature requests. That function is only meant to operate against columns or variables in a single row; it cannot be used to concatenate values across rows.

**More on FOR XML PATH**

FOR XML PATH('') on its own is not good enough – it has known problems with XML entitization. For example, if you update one of the pet names to include an HTML bracket or an ampersand:

UPDATE dbo.FamilyMemberPets

SET Pet = N'Qui>gle&y'

WHERE Pet = N'Quigley';

These get translated to XML-safe entities somewhere along the way:

Qui&gt;gle&amp;y

So I always use PATH, TYPE).value(), as follows:

SELECT Name, Pets = STUFF((SELECT N', ' + Pet

FROM dbo.FamilyMemberPets AS p2

WHERE p2.name = p.name

ORDER BY Pet

FOR XML PATH(N''), TYPE).value(N'.[1]', N'nvarchar(max)'), 1, 2, N'')

FROM dbo.FamilyMemberPets AS p

GROUP BY Name

ORDER BY Name;

I also always use NVARCHAR, because you never know when some underlying column will contain Unicode (or later be changed to do so).

You may see the following varieties inside .value(), or even others:

... TYPE).value(N'.', ...

... TYPE).value(N'(./text())[1]', ...

These are interchangeable, all ultimately representing the same string; the performance differences between them (more below) were negligible and possibly completely nondeterministic.

Another issue you may come across is certain ASCII characters that are not possible to represent in XML; for example, if the string contains the character 0x001A (CHAR(26)), you will get this error message:

Msg 6841, Level 16, State 1, Line 51  
FOR XML could not serialize the data for node 'NoName' because it contains a character (0x001A) which is not allowed in XML. To retrieve this data using FOR XML, convert it to binary, varbinary or image data type and use the BINARY BASE64 directive.

This seems pretty complicated to me, but hopefully you don't have to worry about it because you're not storing data like this or at least you're not trying to use it in grouped concatenation. If you are, you may have to fall back to one of the other approaches.

**Performance**

The above sample data makes it easy to prove that these methods all do what we expect, but it is hard to compare them meaningfully. So I populated the table with a much larger set:

TRUNCATE TABLE dbo.FamilyMemberPets;

INSERT dbo.FamilyMemberPets(Name,Pet)

SELECT o.name, c.name

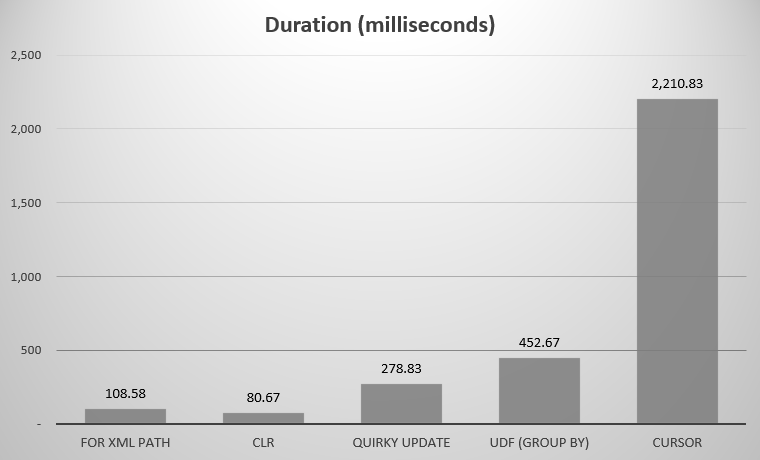
FROM sys.all\_objects AS o

INNER JOIN sys.all\_columns AS c

ON o.[object\_id] = c.[object\_id]

ORDER BY o.name, c.name;

For me, this was 575 objects, with 7,080 total rows; the widest object had 142 columns. Now again, admittedly, I did not set out to compare every single approach conceived in the history of SQL Server; just the few highlights I posted above. Here were the results:

[](https://sqlperformance.com/wp-content/uploads/2014/08/gcat2.png)

You may notice a couple of contenders missing; the UDF using DISTINCT and the recursive CTE were so off the charts that they would skew the scale. Here are the results of all seven approaches in tabular form:

|  |  |
| --- | --- |
| **Approach** | **Duration (milliseconds)** |
| FOR XML PATH | 108.58 |
| CLR | 80.67 |
| Quirky Update | 278.83 |
| UDF (GROUP BY) | 452.67 |
| UDF (DISTINCT) | 5,893.67 |
| Cursor | 2,210.83 |
| Recursive CTE | 70,240.58 |

*Average duration, in milliseconds, for all approaches*

Also note that the variations on FOR XML PATH were tested independently but showed very minor differences so I just combined them for the average. If you really want to know, the .[1] notation worked out fastest in my tests; YMMV.

**Conclusion**

If you are not in a shop where CLR is a roadblock in any way, and especially if you're not just dealing with simple names or other strings, you should definitely consider the CodePlex project. Don't try and re-invent the wheel, don't try unintuitive tricks and hacks to make CROSS APPLY or other constructs work just a little faster than the non-CLR approaches above. Just take what works and plug it in. And heck, since you get the source code too, you can improve upon it or extend it if you like.

If CLR is an issue, then FOR XML PATH is likely your best option, but you'll still need to watch out for tricky characters. If you are stuck on SQL Server 2000, your only feasible option is the UDF (or similar code not wrapped in a UDF).

**Next Time**

A couple of things I want to explore in a follow-on post: [removing duplicates from the list](https://www.mssqltips.com/sqlservertip/4140/removing-duplicates-from-strings-in-sql-server/), ordering the list by something other than the value itself, cases where putting any of these approaches into a UDF can be painful, and [practical use cases for this functionality](https://blogs.sentryone.com/aaronbertrand/grouped-concatenation-use-cases/).

* [Facebook4](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?share=facebook&nb=1)
* [Twitter](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?share=twitter&nb=1)
* [LinkedIn](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?share=linkedin&nb=1)

22 thoughts on “Grouped Concatenation in SQL Server”

1. *Joe Celko* says:

[August 9, 2014 at 11:35 AM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-95343)

Why would anyone want to destroy First Normal Form (1NF) in the database tier? Are columns with strings 'a,b,c' and 'c,a,b' equal? why or why not? It gets to be even more of a mess as you get into it.

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-95343)

1. *Data Minor* says:

[August 9, 2014 at 11:53 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-95360)

@Joe I suspect your argument is a straw man. I don't think Aaron wants to STORE the data in that form at all. At some point though the story told by the data has to be conveyed to the eyes of the beholder. Whilst we'd all love our client apps to do this work for us, on rare occasions they don't and that might well be beyond our control. I think this is probably just a sometimes-necessary evil hack to get around client-side limitations. It would have been better had Aaron made that point up front because beginners will read articles like this and apply the advice to no end of inappropriate uses.

The only other comment I'd add to an otherwise excellent article is that I'd love Aaron to expand a bit on the quirky update remarks. Performance aside, what are the issues?

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-95360)

* 1. *Aaron Bertrand* says:

[August 10, 2014 at 12:31 AM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-95363)

Right, what I see a lot of is: "Pointy-haired boss specified these requirements: output from SQL Server as CSV." As academic and purist as we might want to be about it, sometimes people just need a solution. And there are several practical use cases I'll talk about next time that don't necessarily violate relational principles. As for my concerns about quirky update, it's just not supported or documented syntax, and there's no way to \*guarantee\* the order of the output (which you can with FOR XML PATH). Sorry, I just don't trust it. Some discussions about quirky update either in posts or comments:

<http://sqlblog.com/blogs/jamie_thomson/archive/2013/05/28/increment-a-variable-in-the-set-clause-of-an-update-statement-t-sql.aspx>

<http://sqlperformance.com/2012/07/t-sql-queries/running-totals>

<http://sqlperformance.com/2013/07/t-sql-queries/performance-palooza>

<https://www.simple-talk.com/sql/learn-sql-server/robyn-pages-sql-server-cursor-workbench/>

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-95363)

1. *Mark* says:

[October 22, 2015 at 2:11 AM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-112641)

Okay, but what if you're updating a VIEW for example, and you don't have the luxury to place a where clause inside the inner query, and must rely on implicit wheres instead? Then what?

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-112641)

* 1. [*Aaron Bertrand*](https://sentryone.com/AaronBertrand) says:

[October 22, 2015 at 12:43 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-112660)

Hi Mark, I'm having a hard time understanding your scenario. Could you post schema, sample data, and desired results somewhere?

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-112660)

1. *Pravin* says:

[January 15, 2016 at 2:12 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-115515)

Thanx a lot for this article. I helped me a long way to get the SAP text extracted correctly from the database. I'm posting this as few have doubt of where it would practically be handy. SAP text is a long description about the product and stored in a table but in different lines. So in order get the exact text we need to do a group concatenate all the lines to arrive at the desired result. After coding as indicated in the article I faced issues with the HTML Bracket and ampersand signs getting converted, which was not desired. This article really proved handy to get rid of that issue. I'm hereby attaching the code I used for your referral.

select distinct t2.TDNAME, t2.TDSPRAS, t2.TDID,

stuff(( select ' '+T1.tdline

from [dbo].[stTEXT\_SAP\_EQUI\_SAP\_TEXT\_SAP\_EQUI] t1

where t1.tdname=t2.tdname AND t1.tdSPRAS=t2.tdSPRAS

AND T1.TDLINE IS NOT NULL AND LEN(T1.TDLINE)0

order by [counter]

for xml path(''),type).value('.[1]','nvarchar(max)') ,1,1,'') as TDLINE

from [dbo].[stTEXT\_SAP\_EQUI\_SAP\_TEXT\_SAP\_EQUI] t2

group by TDNAME, TDSPRAS, TDID

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-115515)

1. *Charles Burns* says:

[February 21, 2016 at 12:49 AM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-116727)

Thank you for this excellent, thorough article on the matter.  
Most arguments against this functionality (and other missing features such as GREATEST/LEAST) focus on theoretical elegance and rigid adherence to the relational model.  
These complaints are in the vein of those that look down upon use of materialized views and SELECT \*, and those that criticize C for disallowing function definitions within functions (to make the language truly procedural).  
For those that need to get work done in the real world, all of the above database features have their use in production code, even if rarely.  
Group concatenation may not map beautifully to relational calculus, but when working in a limited system such as a simple reporting framework, the feature does map beautifully to what humans can easily understand.

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-116727)

1. *Kathi K* says:

[February 24, 2016 at 9:42 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-116860)

I use the "For XML PATH(")" all the time in 2008 / 2008R2 SQL server – it works great – never had any issues.  
I'm trying to rewrite my query in SQL Server 2014 – the query runs great – will return 1700 rows of data in under 1 second…but if I put in my code that uses the "For XML PATH("), the report runs for 10 plus MINUTES before returning any data – is there something on SQL server 2014 I need to set so this will run? An example of my code is:

STUFF((SELECT ', ' + x.field\_a

FROM dbo.tbl\_a AS x2

WHERE x1.ID = x2.ID

ORDER BY x2.date

FOR XML PATH('')), 1, 2, '')

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-116860)

* 1. [*Aaron Bertrand*](https://sentryone.com/AaronBertrand) says:

[February 24, 2016 at 9:53 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-116861)

No, this doesn't sound normal (though tough to know with just that query fragment what else might be going on). I suspect it has nothing to do with 2014, but rather just different optimizer decisions based on different hardware, different statistics, different plan cache. Try updating relevant statistics and adding OPTION (RECOMPILE) to the query to make sure you are getting a new plan. Also go through [this post](https://blogs.sentryone.com/aaronbertrand/backtobasics-the-runaway-query/) to make sure you aren't categorizing some other symptom as a 10+ minute query. It could be something else entirely. You could also check to see if running under trace flag 9481 (which forces the old estimator) has any impact.

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-116861)

1. *Mir* says:

[June 6, 2016 at 10:19 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-120710)

Hello Aaron, thanks for posting a very well explained article. Can you please guide me in the direction where I can read up about the syntax of XML path and understand what the .value(N'.[1]' stands for also what is the 1,2, N ' ') for? I cant seem to understand what is the reason to use 1,2 or 1,1 in the end.

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-120710)

* 1. [*Aaron Bertrand*](https://sentryone.com/AaronBertrand) says:

[June 6, 2016 at 10:38 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-120712)

Hi Mir,

The N'.[1]' is just telling the .value() directive to take the first value (which is the only one, but is necessary in the syntax). There are other ways to do that, e.g. N'.' or N'/'. But if you don't use a proper XQuery expression there to ensure a single scalar value comes out, you'll get an error (usually 6306 unless some other error happens first).

The other piece you're asking about has nothing to do with FOR XML PATH directly; it is because I am using STUFF() to strip off a single character – the leading comma. If you concatenate with a comma and a space, then it would require stripping off the first two characters. Try it:

DECLARE @x nvarchar(max) = N', foo, bar';

SELECT STUFF(@x, 1, 1, N''), STUFF(@x, 1, 2, N'');

The first one has a leading space, the second one doesn't. Just be careful because if you aren't using a comma and a space, you could lose the first character of the string.

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-120712)

* + 1. *Amir* says:

[June 7, 2016 at 10:11 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-120765)

Hello Aaron, thank you so very much once again. Your example on STUFF cleared things up for me but I am still a little unclear on the .VALUE() method.  
So below is what my understanding is and then what I am still not clear about:

a. basically using TYPE).value(N'.[1]' is same as using TYPE).value(N'.' , right?  
b. And when you say , N'.[1]' is just telling the .value() directive to take the first value (which is the only one, what do you mean? This is the part I am not clear on.

Sorry, if my question seems very silly but I am struggling to grasp the concept here.

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-120765)

* + - 1. [*Aaron Bertrand*](https://sentryone.com/AaronBertrand) says:

[June 7, 2016 at 10:38 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-120766)

It is just telling the syntax that it is not to expect a set within .value().

SELECT N',' + name FROM sys.databases FOR XML PATH, TYPE;

-- creates XML consisting of a set of &lt;row&gt;s

-- in order to get our string we need to

-- extract a concatenated VALUE from XML

-- however:

SELECT (SELECT N',' + name FROM sys.databases FOR XML PATH, TYPE).value();

-- generates an error message : value function requires 2 argument(s)

-- From -- https://msdn.microsoft.com/en-us/library/ms178030.aspx

-- we need to pass an XQuery expression and an output data type

-- "The XQuery must return at most one value."

-- So we limit that using '.' or '.[1]' or '/' or ...

SELECT (SELECT N',' + name FROM sys.databases FOR XML PATH, TYPE

).value(N'.', N'nvarchar(max)');

-- generates a single string: ,master,tempdb,model,msdb,...

SELECT STUFF((SELECT N',' + name FROM sys.databases FOR XML PATH, TYPE

).value(N'.', N'nvarchar(max)'), 1, 1, N'');

-- generates the same string but drops the leading comma

There are probably dozens of posts out there explaining FOR XML PATH and concatenation that can do a better job than I can. This post was meant to help you understand the most efficient approaches, even if I don't even understand the entire set of internal mechanics.

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-120766)

* + - * 1. *Amir* says:

[June 8, 2016 at 7:40 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-120812)

Hello Aaron, that was a fantastic explanation. I know there are a lot posts out there but in none of those posts has anyone explained the way you did and I know for a fact that there are many people out there who will really appreciate your explanation.

I for one really do appreciate the time and effort you have taken because I was struggling with this for a month now and I can finally rest a little. :-)

Thank you so much and God Bless!!

Amir

1. *Lester Chin* says:

[July 27, 2016 at 5:57 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-122236)

Hello Aaron,

Just wanted to thank you for your helpful article. I used your recursive CTE technique in a current project and it worked perfectly in grouping and concatenating very long strings of data.

Your excellent examples are very much appreciated.

Lester Chin

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-122236)

1. Pingback: [SQL LISTAGG concatenates values of multiple rows into an delimited string such CSV. – Cloud Data Architect](http://www.dataarchitect.cloud/sql-listagg-concatenates-values-of-multiple-rows-into-an-delimited-string-such-csv/)
2. Pingback: [SQL LISTAGG concatenates values of multiple rows into an delimited string such as CSV. – Cloud Data Architect](http://www.dataarchitect.cloud/sql-listagg-concatenates-values-of-multiple-rows-into-an-delimited-string-such-as-csv/)
3. [*Wesley Mota DBA SQL Server*](https://dbasqlserverbr.com.br/) says:

[July 15, 2017 at 8:20 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-132112)

Great job!

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-132112)

1. *Wayne Erfling* says:

[August 1, 2017 at 1:30 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-133121)

Do you have any insight into what kind of test I would have to devise to stress a c# SQL CLR aggregate in a way that might cause SQL Server to present it with data out of the expected order?

For ten years I've been using the SQL CLR routine "Concatenate()" derived from an MSDN article, with a bit of added functionality and attributes such as "IsInvariantToOrder".

One of my colleagues steadfastly refused to use this routine because "the fields could be concatenated out of order", instead insisting on a version of the XML PATH technique considerably more verbose than those in your examples and significantly affecting readability.

Even though IsInvariantToOrder is still documented as "for future use" (<https://msdn.microsoft.com/en-us/library/microsoft.sqlserver.server.sqluserdefinedaggregateattribute.isinvarianttoorder.aspx>), and even though the sub-query (SELECT TOP … ORDER BY … ) is not supposed to order its values in a way that would allow aggregates to take advantage of the ORDER BY, I have never observed an instance where this c# SQL CLR aggregate failed to order its components as they would have been had they been allowed to be returned in rows.

While I've used this aggregate in queries from large tables and in complex queries with parallel elements, none of the databases with which I've worked has used partitioning, which I thought might also influence aggregate calls.

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-133121)

* 1. *Aaron Bertrand* says:

[August 2, 2017 at 7:25 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-133245)

The fact that order isn't guaranteed is enough for me. I don't know how much effort it's worth contriving some case where it might fail – even if you do that, that might not be the only case that will ever fail, so I'm not sure what it would gain. This to me is a lot like trying to disprove there are no poisonous snakes in your yard.

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-133245)

1. *Wayne Erfling* says:

[August 3, 2017 at 11:57 AM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-133294)

Bertrand, thanks for the REPLY. I'm hoping to find somebody in the community with perhaps a connection to the SQL Server team that could update my understanding not only of the internals of aggregation but also of SQL CLR, which hasn't had documentation updates in years (I'm old fashioned and believe in up-to-date docs). I was in similar territory a few years ago in trying to learn about the physical locations of the b-tree index blocks in large tables (we had about 7 levels in the tree).

I hesitate to point out that one poisonous snake would be enough to disprove [that] there are no poisonous snakes in my yard ;)

Double negatives notwithstanding, I don't agree that proving a negative is a good analogy for understanding an algorithm that just doesn't happen to be published.

Beyond that, of course the unpublished algorithm could always change. At present I prefer SQL CLR before SQL 2017 as long as one doesn't require ordering, which will not fail due to an XML serialization issue and is faster to execute.

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-133294)

1. *Michael Hagesfeld* says:

[November 16, 2017 at 1:27 PM](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation#comment-139866)

This. Is Fantastic. I had NO clue how I was going to accomplish what I needed, and this was perfect. Thanks SO MUCH!!

[Reply](https://sqlperformance.com/2014/08/t-sql-queries/sql-server-grouped-concatenation?fbclid=IwAR0TQB1hVQZsUT0a-WigOqLynvGyOMh8DBJ_TnP4tyWK1L6bP7asV_KpeXo#comment-139866)

表單的頂端

表單的底部