

Return a list of related records in SQL Server

Posted At : October 21, 2020 10:45 AM | Posted By : Steve

Related Categories: SQL

I absolutely love SQL (especially SQL Server) and I love how I continually find new ways to solve problems with it. I wish I could remember where I first found this tip to give proper credit, but I really love it.

Have you ever wanted to return a comma-delimited list of records in SQL Server? It turns out, that it is actually pretty easy to do.

For example, if you have users that can work in multiple regions and you need a list of which regions they work in.

If you are using SQL Server 2017 or later, check out how to [Return a list of related records in SQL Server 2017](#)

```
SELECT  UserID,
        STUFF (
            SELECT  ', ' + CONVERT(varchar,RegionID)
            FROM    userregions
            WHERE   userregions.UserID = users.UserID
            FOR XML PATH('')
            ,1,1, '') AS Regions
FROM    users
WHERE   1 = 1
```

You will get back recordset of all of your users and each row will have a comma-delimited list of regions for that user. It will work for any subquery that returns a single varchar field.

That's really all you need. If you are interested in how it works, read on!

FOR XML

The FOR XML functionality of SQL Server is pretty neat by itself.

```
SELECT  WidgetID,WidgetName
FROM    widgets
FOR XML PATH('widget')
```

This will return a query with a really weird column name with the following content:

```
<widget>
<WidgetID>1</WidgetID>
<WidgetName>Apple</WidgetName>
</widget>
<widget>
<WidgetID>2</WidgetID>
<WidgetName>Banana</WidgetName>
</widget>
<widget>
<WidgetID>3</WidgetID>
<WidgetName>Cherry</WidgetName>
</widget>
<widget>
<WidgetID>4</WidgetID>
```

```
<WidgetName>Date</WidgetName>
</widget>
```

If you wanted to make the query useful, then you could just give the resulting column a proper name.

```
SELECT (
    SELECT WidgetID,WidgetName
    FROM widgets
    FOR XML PATH('widget')
) AS widgets
```

If you ditched the string passed in to FOR XML PATH, then you would end up with the following XML:

```
<WidgetID>1</WidgetID>
<WidgetName>Apple</WidgetName>
<WidgetID>2</WidgetID>
<WidgetName>Banana</WidgetName>
<WidgetID>3</WidgetID>
<WidgetName>Cherry</WidgetName>
<WidgetID>4</WidgetID>
<WidgetName>Date</WidgetName>
```

As XML, this isn't worth much but it helps us understand how this works.

What does SQL Server do, however, if it doesn't have a column name? (note the lack of "AS" for each column)

```
SELECT (
    SELECT WidgetID + 0,
           WidgetName + ''
    FROM widgets
    FOR XML PATH('widget')
) AS widgets
```

Then it just returns a messy string.

```
1Apple2Banana3Cherry4Date
```

That is useless, but doesn't have to be.

```
SELECT (
    SELECT CONVERT(varchar,WidgetID) + ': ',
           WidgetName + ';'
    FROM widgets
    FOR XML PATH('')
) AS damages
```

This returns the following:

```
1:Apple;2:Banana;3:Cherry;4:Date;
```

Which, honestly, isn't that bad by itself.

Let's try it with just one field and see what kind of list we get.

```
SELECT (
    SELECT CONVERT(varchar,WidgetID) + ','
    FROM widgets
    FOR XML PATH('')
) AS damages
```

That returns the following list for our example.

```
1,2,3,4,
```

If you don't mind the trailing comma, this could actually work.

STUFF

STUFF is a really handy function with the following method signature:

```
STUFF ( input_string , start_position , length , replace_with_substring )
```

So, if you had numbers representing a time string and wanted to inject a colon in them to make it more readable, you could do the following:

```
SELECT STUFF ( '1230' , 3 , 0 , ':' )
```

(Example from SQLServerTutorial.net)

In this case, as the "length" argument is zero, STUFF didn't replace anything and instead just inserted a character.

Putting it Together

So, to put this all together, all you need to do is move the comma to the start of the string and then use STUFF to remove it.

```
SELECT STUFF (
    SELECT CONVERT(varchar,WidgetID) + ','
    FROM widgets
    FOR XML PATH('')
    ,1,1,'') AS damages
```

In this case, you start with the string "1,2,3,4" and then tell stuff to replace the first character with an empty string.

Enjoy and let me know if you have any questions!