

Re: using OpenXML in T-sql?

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.programming/2004-06/5778.html>

From: Tom Moreau (*tom_at_dont.spam.me.cips.ca*)

Date: 06/24/04

Date: Thu, 24 Jun 2004 19:27:29 -0400

Based on what I see, he didn't represent the true picture. What I see here are code snippets and not a complete set of working code. The @iDoc was never populated, so the code doesn't have a hope of working.

You may want to check out SQL Pro at:

www.pinnaclepublishing.com/sql

--

Tom

Thomas A. Moreau, BSc, PhD, MCSE, MCDBA
SQL Server MVP
Columnist, SQL Server Professional
Toronto, ON Canada
www.pinnaclepublishing.com/sql

"Ed" <anonymous@discussions.microsoft.com> wrote in message
news:215ad01c45a28\$3f18fb50\$a001280a@phx.gbl...

Thanks. I did read your other article on MSDN. Very insightful. However, of the journals I subscribe to (Sql Server Solutions from elementKjournals) I currently do not have a subscription with Sql Server Professional and thus cannot access your article there.

While I'm at it, here is the entire article I was referring too (actually, just a tip) at the beginning of my post. The author makes a reference to using comma delimited strings and opts for the xml string instead. I just couldn't follow the solution, but it sounds pretty nifty. My confusion is if the guy inadvertently left out `sp_xml_preparedocument`

or he is actually not using it. Here is his whole tip.

>>

ADVISOR TIP

SQL Server and Arrays

Explore these workarounds to a common problem.

By Russ Nemhauser

One of the most frequent questions I receive about SQL Server development has to do with passing an array (or collection) of values to a stored procedure as an input parameter. Quite often, I've had to return rows from a table based on selections the user made. There's obviously a performance hit if you retrieve one row at a time because you can only specify the row's ID.

microsoft.public.sqlserver.programming: Re: using OpenXML in T-sql?

Developers have discovered a few workarounds for cases like this. Some create a slew of input parameters with default values (which makes them optional):

```
CREATE PROCEDURE GetRecords
    @ID1 INT,
    @ID2 INT,
    @ID3 INT,
    @ID4 INT
AS ...
```

With an approach like this, you're betting you'll never have to specify more values than the number of input parameters you create. In addition, inserting all these input parameters into a temporary table can cause a noticeable decrease in performance if you're dealing with a lot of values.

Another approach is to create one large VARCHAR input parameter, and then attempt to parse it:

```
CREATE PROCEDURE GetRecords
    @Params VARCHAR(8000)
AS ...
```

Your stored procedure will have to retrieve the values from the @Params parameter and insert them into a temporary table so you can do things such as joins or work with the IN statement. You can separate each value in the parameter by a comma or other delimiter:

```
EXEC GetRecords '4,19,42'
```

One problem with this approach is T-SQL isn't a programming language. It's a data access language, and working with strings isn't necessarily ideal-especially where performance is concerned.

SQL Server 2000 offers you the ability to work with XML in your stored procedures. If you stick with the concept of using a large VARCHAR parameter, you can pass in an XML string that represents the ID numbers with which you have to work:

```
<IDNumbers>
  <ID value="4"/>
  <ID value="19"/>
  <ID value="42"/>
</IDNumbers>
```

Using T-SQL's OPENXML, you can insert these values (no matter how many there are) into a temporary table with ease:

```
DECLARE @iDoc INT
SELECT Value
INTO #tmp
FROM OPENXML(@iDoc, '/IDNumbers/ID', 1)
WITH ([Value] INT)
```

Now that the values exist in the temporary table, you can use them in any way you'd use values in any table.

Furthermore, you can create the temporary table manually using the CREATE TABLE statement if you have to get fancy with indexes or keys.

<<

>-----Original Message-----

>I have written an article on passing arrays to stored procs via

>comma-delimited strings. If you subscribe to SQL Server Professional, check

>out:

>

><http://www.pinpub.com/html/main.isx?sub=64&story=819>

microsoft.public.sqlserver.programming: Re: using OpenXML in T-sql?

```
>
>Essentially, I create a UDF to translate a comma-
delimited string to an XML
>doc:
>
>create function dbo.ArrayToXML
>(
> @InputStr varchar (8000)
>, @Delim     varchar (5)
>)
>returns varchar (8000)
>as
>begin
> return ('<ROOT><Worktable Value="'
>         + replace (@InputStr
>                   , @Delim
>                   , '"/><Worktable Value="' )
>         + '"/></ROOT>')
>end
>go
>Regardless, you still need to go with
sp_xml_preparedocument and OPENXML()
>to make use of it.
>
>
>--
>Tom
>
>-----
-----
>Thomas A. Moreau, BSc, PhD, MCSE, MCDBA
>SQL Server MVP
>Columnist, SQL Server Professional
>Toronto, ON Canada
>www.pinnaclepublishing.com/sql
>
>
>"Ed" <anonymous@discussions.microsoft.com> wrote in
message
>news:20b3b01c45a22$297b9790$a601280a@phx.gbl...
>Yes. I am aware of this. According to the examples in
>Books Online for OpenXml the examples all called
>sp_xml_prepareDocument. But in this article, the author
>did not call sp_xml_prepareDocument. I was just checking
>if he found a way to not have to do that. The actual
>article was about passing an array to an SP. He suggests
>you can pass an xml string to the sp (as opposed to an xml
>doc - values from array would be in an xml string) and use
>Openxml to retrieve the values in one call rather than
>looping through an array. Any thoughts on that?
>
>Thanks,
>Ed
>
>
>>-----Original Message-----
>>You did not run sp_xml_prepare_document or OPENXML().
>>Check out:
>>
>>http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsqpro01/html/sql01c5.asp
>>
```

microsoft.public.sqlserver.programming: Re: using OpenXML in T-sql?

```
>>--
>>Tom
>>
>>-----
-
>-----
>>Thomas A. Moreau, BSc, PhD, MCSE, MCDBA
>>SQL Server MVP
>>Columnist, SQL Server Professional
>>Toronto, ON Canada
>>www.pinnaclepublishing.com/sql
>>
>>
>>"Ed" <anonymous@discussions.microsoft.com> wrote in
>message
>>news:210d101c45ala$ffe60250$a101280a@phx.gbl...
>>Hello,
>>
>>I stumble onto an article which explains how you can pass
>>data to a T-sql statement in the form of an xml string
>>but
>>I got lost on how the xml string is invoked into the
>>Select statement. Here is the meat of the article:
>>
>>>>
>>SQL Server 2000 offers you the ability to work with XML
>>in
>>your stored procedures. If you stick with the concept of
>>using a large VARCHAR parameter, you can pass in an XML
>>string that represents the ID numbers with which you have
>>to work:
>>
>><IDNumbers>
>> <ID value="4"/>
>> <ID value="19"/>
>> <ID value="42"/>
>></IDNumbers>
>>
>>Using T-SQL's OPENXML, you can insert these values (no
>>matter how many there are) into a temporary table with
>>ease:
>>
>>DECLARE @iDoc INT
>>SELECT Value
>>INTO #tmp
>>FROM OPENXML(@iDoc, '/IDNumbers/ID', 1)
>> WITH ([Value] INT)
>>
>><<
>>
>>I am thinking like this:
>>
>>Declare @xml varchar(8000)
>>Set @xml='<IDNumbers>
>> <ID value="4"/>
>> <ID value="19"/>
>> <ID value="42"/>
>></IDNumbers>'
>>
>>So could anyone explain where @xml gets used in the
>>example from the article above?
>>
```

microsoft.public.sqlserver.programming: Re: using OpenXML in T-sql?

```
>>Thanks (again),  
>>Ed  
>>  
>>.  
>>  
>  
>.  
>
```