Some Usages for XML

Passing a rowset to a stored procedure:

Many times you need to pass a table result to a stored procedure. Despite the fact that SQL Server 2000 and SQL Server 2005 have a Data Type which is called table variable, you can not use it as a parameter in stored procedures. From SQL 2000 onward Microsoft has supported XML in a great way. I know most of you have used XML or at least have heard a lot about XML; however, I do not want to dig into XML, but I want to show you some of its usages. For those of you who are not familiar with XML the following link is a good point to start: http://www.topxml.com/sql/learn_sql_server_xml_tutorial.asp

Let's get started. We have the following table:

```
CREATE TABLE Employee (empid INT,
name VARCHAR (20),
salary MONEY NOT NULL
)
```

And we want to insert the following list to the employee's table:

Empid	Name	Salary
1	John	$10000.00
2	Joseph	$3000.00
3	Melissa	$2500.00
4	Dan	$2000.00

To do so, we create a stored procedure to process the XML. A new data type is introduced In SQL 2005 which is the most suitable for holding XML values you might also use VARCHAR (MAX) or if you use SQL 2000, you can use VARCHAR (8000) instead.

```
Create PROC usp_ProcessRowset @p XML
AS
SET NOCOUNT ON
```

```
DECLARE @ih INT

EXEC sp_XML_preparedocument @ih output, @p

INSERT Employee
SELECT *
FROM OPENXML(@ih, 'data/emp')
WITH Employee

EXEC sp_XML_removedocument @ih
```

The next step is to change the input date to valid XML, simply declare a variable of XML and assign the desired XML data, and then execute the above stored procedure:

```
declare @x XML

set @x=' <data>
 <emp empid="1" name="John" salary="10000" />
 <emp empid="2" name="Joseph" salary="3000" />
 <emp empid="3" name="Melissa" salary="2500" />
 <emp empid="4" name="Dan" salary="2000" />
 </data>'
exec usp_ProcessRowset @x
--Just to make sure it works
select * from      Employee
```

Here is the result:

```
empid       name                 salary
----------- -------------------- ---------------------
1           John                 10000.00
2           Joseph               3000.00
3           Melissa              2500.00
4           Dan                  2000.00

(4 row(s) affected)
```

You can collect the user inputs in a XML file and then in off time pass it to the stored procedure. By using this technique, you will minimize the call of stored procedure, resulting in high performance.

Extracting values from comma delimited list (CSV)

Several times in forums I have read that new SQL developers ask how they can pass an array to a procedure. IMHO, the only answer to that question is comma delimited lists; however changing it back to rowset is a pain. Some developers will use loop structure to extract values from the list. In this article I will show you the new way to solve this problem.

The loop solution:

```
create  proc usp_LoopVersion @CSV varchar(max)
AS

SET NOCOUNT ON
```

```
declare @res table (val  varchar(100))
 declare @pos int
set @pos=CHARINDEX(',',@CSV,1)

while @pos > 0
begin
  insert @res
  values(ltrim(substring(@CSV,1,@pos-1)))

  set @CSV=right(@CSV,len(@CSV)-@pos)

  set @pos=CHARINDEX(',',@CSV,1)

end

insert @res
values(ltrim(@CSV))

select *
 from @res
```

It simply finds the position of the first comma, then extracts and adds the first value of the list to the @res table variable from the beginning of the list to the position of the foundation comma, after that it removes the added value from the list. It repeats the loop until there is no comma found in the list.

The XML solution:

```
CREATE PROC usp_LoopVersion @CSV VARCHAR(MAX)
AS
SET NOCOUNT ON
DECLARE @idoc INT
--SECTION A
SET @CSV ='<root><x val="'+replace(@CSV,',','" /><x val="')+'" /></root>'

EXEC sp_XML_preparedocument @idoc OUTPUT, @CSV

SELECT *
FROM OPENXML (@idoc, '/root/x',1)
      WITH (val VARCHAR (100) )

EXEC sp_XML_removedocument @idoc
```

I replaced commas with " /><x val=", so in SECTION A the @CSV will fall into a valid XML expression. After that I invoked sp_XML_preparedocument to obtain a document handler which is passed to OPENXML. Calling sp_XML_removedocument is a crucial step. Failure to do so leaves the document in memory.

To test the stored procedures run the following snippet code:

```
DECLARE @i int
DECLARE @CSV varchar(max)

set @i=0
while @i < 2
begin
  SET @CSV=isnull(@CSV+',test','test '  )
  set @i=@i+1
```

```
end

print     @CSV
exec usp_LoopVersion       @CSV
exec usp_LoopVersion   @CSV
```

Note: you can use any delimiter rather than comma or if you like you can parameterize the delimiter that I left it for you. You can download the stored procedure's code from here:code.txt

Performance:

On my machine with a 10,000 comma separated list, the XML version ran 3 seconds faster than the loop version.

Conclusion:

Microsoft has developed XML in SQL Server 2005 and has added new features such as XML data type and the MAX specifier. I tried to show you some of the usages of XML that might be practical in the real world. I hope that this article has enlightened you.