Iterating over JSON and XML values

One handy thing when working with JSON or XML is to take part of the data and represent it as tabular data, since everything in SQL Server works better as a table, pulling values out and representing them as a table can be handy.

Going back to our original data as shown below from my original blog post that kicked off JSON and XML month at SteveStedman.com we will be using the same XML and JSON. The goal here would be to pull out each employees first name from the XML and JSON and represent it as a table.

One option (the hard way) would be to use the JSON_VALUE and xmlData.Value functions to pull

out each value, then join them into a table with a UNION ALL, somthing like this:

```
SELECT JSON_VALUE(JsonData, '$.employees[0].firstName') as FirstName   FR
OM XmlAndJson  WHERE id = 2  UNION ALL  SELECT JSON_VALUE(JsonData, '$.em
ployees[1].firstName') as FirstName   FROM XmlAndJson  WHERE id = 2  UNIO
N ALL  SELECT JSON_VALUE(JsonData, '$.employees[2].firstName') as FirstNa
me   FROM XmlAndJson  WHERE id = 2  ;    SELECT XmlData.value('(/employee
s/employee/firstName)[1]', 'varchar(max)') as FirstName   FROM XmlAndJson
  WHERE id = 2   UNION ALL  SELECT XmlData.value('(/employees/employee/fi
rstName)[2]', 'varchar(max)') as FirstName   FROM XmlAndJson  WHERE id =
2  UNION ALL  SELECT XmlData.value('(/employees/employee/firstName)[3]',
'varchar(max)') as FirstName   FROM XmlAndJson  WHERE id = 2  ;
```

Which as I mention is the hard way, but it does produce the following results.

The results are what we are looking for in this specific example, but where they break down is when there are more employees represented in the XML, for each employee we need to add another UNION to bring the results together. That is not very iterative and since the title of this post includes the word iterating, we need to focus on how to do that.

Now we introduce the CROSS APPLY functionality that can be used like a JOIN to take a value from one result set (table) and apply it to a function that gets called once for each row. You can reference my [JOIN TYPES poster for using CROSS APPLY](). Something like this:

```
 -- JSON Example SELECT id, JSON_VALUE(x.[value], '$.firstName') as first
Name   FROM XmlAndJson   CROSS APPLY OPENJSON(JSON_QUERY(JsonData, '$.emp
loyees')) AS x  WHERE id = 2;  -- XML Exmaple SELECT id, otherXml.value('
(./text())[1]', 'VARCHAR(10)')  as firstName   FROM XmlAndJson  CROSS APP
LY XmlData.nodes('/employees/employee/firstName') as MyDerivedTable (othe
rXml)  WHERE id = 2;
```

Which produces similar results to the previous example, but with much more flexible code. Now we can try it with a bigger data set. Not big data, just bigger data.

```
GO -- bigger dataset DECLARE @XmlData as XML; set @XmlData = '        Ste
ve  Stedman           Derrick  Bovenkamp           Bill  Simpkins
   Beth  Allen           John  Jones           Mary  Smith    ';  DECLARE
 @JsonData as VARCHAR(MAX); set @JsonData = ' {"employees":[   { "firstNa
me":"Steve", "lastName":"Stedman" },   { "firstName":"Derrick", "lastName
":"Bovenkamp" },   { "firstName":"Bill", "lastName":"Simpkins" },   { "fi
rstName":"Beth", "lastName":"Allen" },   { "firstName":"John", "lastName"
:"Jones" },   { "firstName":"Mary", "lastName":"Smith" } ]}';   INSERT IN
TO XmlAndJson(XmlData, JsonData) VALUES (@XmlData, @JsonData );
```

Then we run the same query to access the data, just changing the row id that we are looking for.

```
 -- JSON Example SELECT id, JSON_VALUE(x.[value], '$.firstName') as first
Name   FROM XmlAndJson   CROSS APPLY OPENJSON(JSON_QUERY(JsonData, '$.emp
loyees')) AS x  WHERE id = 11;  -- XML Exmaple SELECT id, otherXml.value(
'(./text())[1]', 'VARCHAR(10)')  as firstName   FROM XmlAndJson  CROSS AP
PLY XmlData.nodes('/employees/employee/firstName') as MyDerivedTable (oth
erXml)  WHERE id = 11;
```

From the results you can see we get all the employees first names back in this case.

So by using the cross apply we can effectively iterate over each row in a chunk of XML or JSON quickly and easily.

Related Links:

- Getting started with XML and JSON in SQL Server.
- JSON and XML Month at SteveStedman.com
- SQL 2019 Page and Row Compression with XML and JSON
- Accessing XML and JSON content