BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

DATA WAREHOUSING SS-G515

LAB-7[Working with XML data in SQL server]

DATE: 25/05/2021 **TIME: 02 Hours**

XML (eXtensible Markup Language) is one of the most common formats used to share information between different platforms. Owing to its simplicity and readability, it has become the de-facto standard for data sharing. In addition, XML is easily extendable.

In this lab , we will see how we can work with XML in SQL Server. We will see how to convert tables in SQL into XML, how to load XML documents into SQL Server and how to create SQL tables from XML documents.

Let's first generate some dummy data. We will use this data to create XML documents. Execute the following script:

```
CREATE DATABASE Showroom
 Use Showroom
CREATE TABLE Car
(
  CarId int identity(1,1) primary key,
  Name varchar(100),
  Make varchar(100),
  Model int ,
  Price int ,
  Type varchar(20)
)
```

```
insert into Car( Name, Make,  Model , Price, Type)
VALUES ('Corrolla','Toyota',2015, 20000,'Sedan'),
('Civic','Honda',2018, 25000,'Sedan'),
('Passo','Toyota',2012, 18000,'Hatchback'),
('Land Cruiser','Toyota',2017, 40000,'SUV'),
('Corrolla','Toyota',2011, 17000,'Sedan'),
('Vitz','Toyota',2014, 15000,'Hatchback'),
('Accord','Honda',2018, 28000,'Sedan'),
('7500','BMW',2015, 50000,'Sedan'),
('Parado','Toyota',2011, 25000,'SUV'),
('C200','Mercedez',2010, 26000,'Sedan'),
('Corrolla','Toyota',2014, 19000,'Sedan'),

('Civic','Honda',2015, 20000,'Sedan')
```

In the script above, we created a Showroom database with one table Car. The Car table has five attributes: CarId, Name, Make, Model, Price, and Type. Next, we added 12 dummy records in the Car table.

Converting into XML from SQL tables

The simplest way to convert data from SQL tables into XML format is to use the FOR XML AUTO and FOR XML PATH clauses.

FOR XML AUTO in SQL SERVER

The FOR XML AUTO clause converts each column in the SQL table into an attribute in the corresponding XML document.

Execute the following script:

```
1 USE Showroom
2 SELECT * FROM Car
3 FOR XML AUTO
```

In the console output you will see the following:

Click on the link and you will see the following document in a new query window of SQL Server management studio:

```
<Car CarId="1" Name="Corrolla" Make="Toyota" Model="2015" Price="20000" Type="Sedan" />
<Car CarId="2" Name="Civic" Make="Honda" Model="2018" Price="25000" Type="Sedan" />
<Car CarId="3" Name="Passo" Make="Toyota" Model="2012" Price="18000" Type="Hatchback" />
<Car CarId="4" Name="Land Cruiser" Make="Toyota" Model="2017" Price="40000" Type="SUV" />
<Car CarId="5" Name="Corrolla" Make="Toyota" Model="2011" Price="17000" Type="Sedan" />
<Car CarId="6" Name="Vitz" Make="Toyota" Model="2014" Price="15000" Type="Hatchback" />
<Car CarId="7" Name="Accord" Make="Honda" Model="2018" Price="28000" Type="Sedan" />
<Car CarId="8" Name="7500" Make="BMW" Model="2015" Price="50000" Type="Sedan" />
<Car CarId="9" Name="Parado" Make="Toyota" Model="2011" Price="25000" Type="SUV" />
<Car CarId="10" Name="C200" Make="Mercedez" Model="2010" Price="26000" Type="Sedan" />
<Car CarId="11" Name="Corrolla" Make="Toyota" Model="2014" Price="19000" Type="Sedan" />
<Car CarId="12" Name="Civic" Make="Honda" Model="2015" Price="20000" Type="Sedan" />
```

You can see that for each record an element Car has been created in the XML document, and for each column, an attribute with the same name has been added to each element in the XML document.

FOR XML PATH in SQL SERVER

The FOR XML AUTO class creates an XML document where each column is an attribute. On the other hand, the FOR XML PATH will create an XML document where each record is an element and each column is a nested element for a particular record. Let's see this in action:

```
1 USE Showroom
2 SELECT * FROM Car
3 FOR XML PATH
```

A snapshot of the output is as follows:

```
<row>
  <CarId>1</CarId>
  <Name>Corrolla</Name>
  <Make>Toyota</Make>
  <Model>2015</Model>
  <Price>20000</Price>
  <Type>Sedan</Type>
</row>
<row>
  <CarId>2</CarId>
  <Name>Civic</Name>
  <Make>Honda</Make>
  <Model>2018</Model>
  <Price>25000</Price>
  <Type>Sedan</Type>
</row>
<row>
  <CarId>3</CarId>
  <Name>Passo</Name>
  <Make>Toyota</Make>
  <Model>2012</Model>
  <Price>18000</Price>
  <Type>Hatchback</Type>
</row>
<row>
  <CarId>4</CarId>
  <Name>Land Cruiser</Name>
  <Make>Toyota</Make>
  <Model>2017</Model>
  <Price>40000</Price>
  <Type>SUV</Type>
</row>
```

In the output, you will see a total of 12 elements (the screenshot shows only the first 4). You can see that each column name has been converted to an element. However, there is one problem; by default, the parent element name is “row”. We can change that using the following query:

```
USE Showroom
SELECT * FROM Car
FOR XML PATH ('Car')
```

```
<Car>
  <CarId>1</CarId>
  <Name>Corrolla</Name>
  <Make>Toyota</Make>
  <Model>2015</Model>
  <Price>20000</Price>
  <Type>Sedan</Type>
</Car>
<Car>
  <CarId>2</CarId>
  <Name>Civic</Name>
  <Make>Honda</Make>
  <Model>2018</Model>
  <Price>25000</Price>
  <Type>Sedan</Type>
</Car>
<Car>
  <CarId>3</CarId>
  <Name>Passo</Name>
  <Make>Toyota</Make>
  <Model>2012</Model>
  <Price>18000</Price>
  <Type>Hatchback</Type>
</Car>
<Car>
  <CarId>4</CarId>
  <Name>Land Cruiser</Name>
  <Make>Toyota</Make>
  <Model>2017</Model>
  <Price>40000</Price>
  <Type>SUV</Type>
</Car>
```

In the output, you can see Car as the parent element for each sub-element. However, the document is not well-formed as there is no root element in the document. To add a root element, we need to execute the following script:

```
1 USE Showroom
2 SELECT * FROM Car
3 FOR XML PATH ('Car'), ROOT('Cars')
```

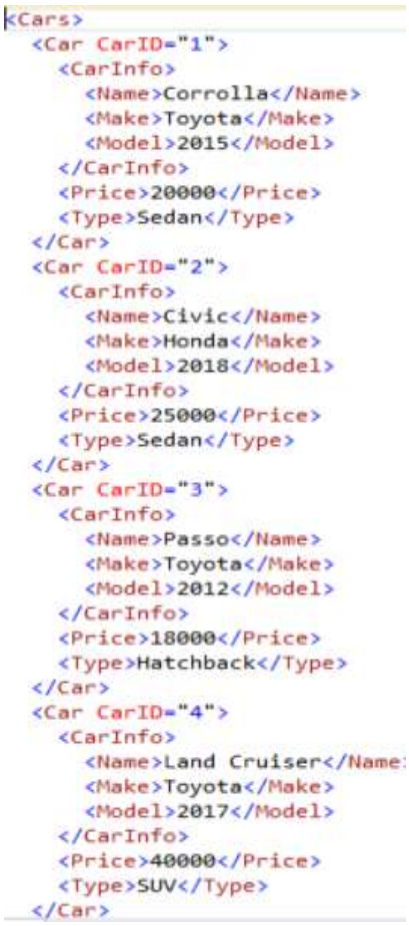
In the output, you should see “Cars” as the root element as shown below:

```
<Cars>
  <Car>
    <CarId>1</CarId>
    <Name>Corrolla</Name>
    <Make>Toyota</Make>
    <Model>2015</Model>
    <Price>20000</Price>
    <Type>Sedan</Type>
  </Car>
  <Car>
    <CarId>2</CarId>
    <Name>Civic</Name>
    <Make>Honda</Make>
    <Model>2018</Model>
    <Price>25000</Price>
    <Type>Sedan</Type>
  </Car>
  <Car>
    <CarId>3</CarId>
    <Name>Passo</Name>
    <Make>Toyota</Make>
    <Model>2012</Model>
    <Price>18000</Price>
    <Type>Hatchback</Type>
  </Car>
  <Car>
    <CarId>4</CarId>
    <Name>Land Cruiser</Name>
    <Make>Toyota</Make>
    <Model>2017</Model>
    <Price>40000</Price>
    <Type>SUV</Type>
  </Car>
```

Now suppose you want that Car-id should be the attribute of the Car element rather than an element. We can add further nesting levels to an XML document. For instance, if we want Name, Make and Model elements to be nested inside another element Car-Info we can do so with the following script:

```
USE Showroom
SELECT  CarId as [@CarID],
   Name  AS [CarInfo/Name],
   Make [CarInfo/Make],
   Model [CarInfo/Model],
   Price,
   Type
FROM Car
FOR XML PATH ('Car'), ROOT('Cars')
```

The output looks like this:



Finally, if you want to convert the elements Name and Make into an attribute of element CarInfo, you can do so with the following script:

```
USE Showroom
SELECT  CarId as [@CarID],
   Name  AS [CarInfo/@Name],
   Make [CarInfo/@Make],
   Model [CarInfo/Model],
   Price,
   Type
FROM Car
FOR XML PATH ('Car'), ROOT('Cars')
```

The output looks like this:

```
<Cars>
  <Car CarID="1">
    <CarInfo Name="Corrolla" Make="Toyota">
      <Model>2015</Model>
    </CarInfo>
    <Price>20000</Price>
    <Type>Sedan</Type>
  </Car>
  <Car CarID="2">
    <CarInfo Name="Civic" Make="Honda">
      <Model>2018</Model>
    </CarInfo>
    <Price>25000</Price>
    <Type>Sedan</Type>
  </Car>
  <Car CarID="3">
    <CarInfo Name="Passo" Make="Toyota">
      <Model>2012</Model>
    </CarInfo>
    <Price>18000</Price>
    <Type>Hatchback</Type>
  </Car>
  <Car CarID="4">
    <CarInfo Name="Land Cruiser" Make="Toyota">
      <Model>2017</Model>
    </CarInfo>
    <Price>40000</Price>
    <Type>SUV</Type>
  </Car>
```

Save the above XML document with the name Cars.xml. In the next section, we will load this XML script into the SQL Server and will see how to create a table from the XML Document.

Creating a SQL table from an XML document

In the previous section, we saw how to create an XML document from the SQL table. In this section, we will see how to do the reverse i.e. we will create a table in SQL using XML documents.

The document we will use is the document that we created in the last section. One node of the document looks like this:

```
<Cars>
  <Car CarID="1">
    <CarInfo Name="Corrolla" Make="Toyota">
      <Model>2015</Model>
    </CarInfo>
    <Price>20000</Price>
    <Type>Sedan</Type>
  </Car>
```

Creating SQL table using XML attributes

Let's first see how we can create an SQL table using attributes. Suppose we want to create a table with two columns that contain the values from the Name and Make attributes from the CarInfo element. We can do so using the following script:

```
DECLARE @cars xml


SELECT @cars = C

FROM OPENROWSET (BULK 'D:\Cars.xml', SINGLE_BLOB) AS Cars(C)


SELECT @cars


DECLARE @hdoc int


EXEC sp_xml_preparedocument @hdoc OUTPUT, @cars

SELECT *

FROM OPENXML (@hdoc, '/Cars/Car/CarInfo' , 1)

WITH(

   Name VARCHAR(100),

   Make VARCHAR(100)

   )

EXEC sp_xml_removedocument @hdoc
```

In the script above we declare an XML type variable @cars. The variable stores the result returned by the OPENROWSET function which retrieves XML data in binary format. Next using the SELECT @Cars statement we print the contents of the XML file. At this point in time, the XML document is loaded into the memory.

Next, we create a handle for the XML document. To read the attributes and elements of the XML document, we need to attach the handle with the XML document. The sp_xml_preparedocument performs this task. It takes the handle and the document variable as parameters and creates an association between them.

Next, we use the OPENXML function to read the contents of the XML document. The OPENXML function takes three parameters: the handle to the XML document, the path of the node for which we want to retrieve the attributes or elements and the mode. The mode value of 1 returns the attributes only. Next, inside the WITH clause, we need to define the name and type of the attributes that you want returned. In our case the CarInfo element has two attributes Name, and Make, therefore we retrieve both.

As a final step, we execute the sp_xml_removedocument stored procedure to remove the XML document from the memory. In the output you will see values from the Name and Make attributes of the CarInfo element as shown below:

Results Messages

	(No column name)
1	<Cars><Car CarID="1"><CarInfo Name="Corrolla" Ma...

	Name	Make
1	Corrolla	Toyota
2	Civic	Honda
3	Passo	Toyota
4	Land Cruiser	Toyota
5	Corrolla	Toyota
6	Vitz	Toyota
7	Accord	Honda
8	7500	BMW
9	Parado	Toyota
10	C200	Merc...
11	Corrolla	Toyota
12	Civic	Honda

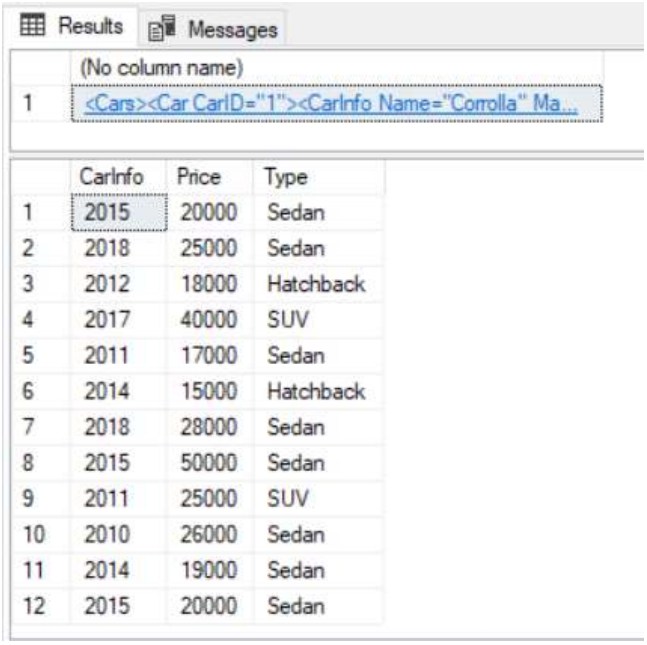
Creating a SQL table using XML elements

To create a SQL table using XML elements, all you have to do is to change the mode value of the OPENXML function to 2 and change the name of the attributes to the name of the element you want to retrieve.

Suppose we want to retrieve the values for the nested CarInfo, Price and Type elements of the parent Car element, we can use the following script:

```
DECLARE @cars xml

SELECT @cars = C
FROM OPENROWSET (BULK 'D:\Cars.xml', SINGLE_BLOB) AS Cars(C)

SELECT @cars

DECLARE @hdoc int

EXEC sp_xml_preparedocument @hdoc OUTPUT, @cars
SELECT *
FROM OPENXML (@hdoc, '/Cars/Car' , 2)
WITH(
   CarInfo INT,
   Price INT,
   Type VARCHAR(100)
   )


EXEC sp_xml_removedocument @hdoc
```

Output of the script above looks like this:



Results Messages

	(No column name)
1	<Cars><Car CarID="1"><CarInfo Name="Corrolla" Ma...

	CarInfo	Price	Type
1	2015	20000	Sedan
2	2018	25000	Sedan
3	2012	18000	Hatchback
4	2017	40000	SUV
5	2011	17000	Sedan
6	2014	15000	Hatchback
7	2018	28000	Sedan
8	2015	50000	Sedan
9	2011	25000	SUV
10	2010	26000	Sedan
11	2014	19000	Sedan
12	2015	20000	Sedan

XML is one of the most popular data formats for information exchange. So this is the way to create a document using XML from a SQL table and how to import into a table in SQL from an XML document.

Exercise:

download the sample data here.

a. retrieve all the customer information(Customer ID,name,address)
b. retrieve all the customer information along with OrderID and OrderDate
c. customer information and their orders along with ProductID and Quantity from each order placed

Now in order to import data from the XML file to a table in SQL Server, use the OPENROWSET function.In the script below, First create a table with a column of data type XML and then reading the XML data from the file using the OPENROWSET function by specifying the file location and name of the XML file as you can see below:

```
CREATE DATABASE OPENXMLTesting
GO
USE OPENXMLTesting
GO

CREATE TABLE XMLwithOpenXML
(
Id INT IDENTITY PRIMARY KEY,
XMLData XML,
LoadedDateTime DATETIME
)

INSERT INTO XMLwithOpenXML(XMLData, LoadedDateTime)
SELECT CONVERT(XML, BulkColumn) AS BulkColumn, GETDATE()
FROM OPENROWSET(BULK 'D:\OpenXMLTesting.xml', SINGLE_BLOB) AS x;
```

```
SELECT * FROM XMLwithOpenXML
```