**Лекция 17:**

**XML** (e**X**tensible **M**arkup **L**anguage) – расширенный язык разметки. Широко распространен, так как является платформонезависимым способом представления данных. Полезен, например, для обмена информацией между слабосвязанными областями.

Примеры из википедии (xml и json):

Пример XML:

<person>

<firstName>Иван</firstName>

<lastName>Иванов</lastName>

<address>

<streetAddress>Московское ш., 101, кв.101</streetAddress>

<city>Ленинград</city>

<postalCode>101101</postalCode>

</address>

<phoneNumbers>

<phoneNumber>812 123-1234</phoneNumber>

<phoneNumber>916 123-4567</phoneNumber>

</phoneNumbers>

</person>

Или:

<person firstName="Иван" lastName="Иванов">

**<address** streetAddress="Московское ш., 101, кв.101" city="Ленинград" postalCode="101101" **/>**

<phoneNumbers>

<phoneNumber>812 123-1234</phoneNumber>

<phoneNumber>916 123-4567</phoneNumber>

</phoneNumbers>

</person>

Пример JSON (*JavaScript Object Notation*). Структура хранения информации: {ключ1:значение1, ключ2: [значение21, значение22]}

{

"firstName": "Иван",

"lastName": "Иванов",

"address": {

"streetAddress": "Московское ш., 101, кв.101",

"city": "Ленинград",

"postalCode": 101101

},

"phoneNumbers": [

"812 123-1234",

"916 123-4567"

]

}

**Как работать с данными в формате XML:**

С SQL 2005 есть встроенный тип данных xml. Можно заводить переменные типа xml или столбцы в колонках такого же типа. Обратите внимание на то, что имена элементов и атрибутов являются регистрозависимыми.

Для работы с xml есть специальные методы:

query() – осуществляет запросы к xml данным, то есть получает нужные фрагменты из xml структуры;

value() – позволяет извлекать значения атрибутов из xml элемента;

exist() – проверяет, существуют ли результаты запроса (узел или атрибут);

nodes() – извлекает поддерево из структуры xml;

modify() – изменяет xml данные.

Методы принимают на вход выражение XPath или запрос XQuery.

Для начала давайте обсудим **методы конструирования XML**:

Xml constructions: ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de\_7xqury/html/a6330b74-4e52-42a4-91ca-3f440b3223cf.htm

**Direct:**

DECLARE @x xml

SET @x=

'<Empl EmplID = "2">

Name and Features Empl

<Name>Ivanov Ivan</Name>

<Features>

<Dep>IT</Dep>

<Salary>50000</Salary>

</Features>

</Empl>'

SELECT @x

<Empl EmplID="2">

Name and Features Empl

<Name>Ivanov Ivan</Name>

<Features>

<Dep>IT</Dep>

<Salary>50000</Salary>

</Features>

</Empl>

Здесь у элемента Empl есть 3 потомка:

- текстовый узел – значение «Name and Features Empl»

- два узла-элемента «Name» и «Features»:

- узел «Name» имеет одного потомка: узел-значение «Ivanov Ivan»

- узел «Features» имеет двух потомков: узел-элемент Dep и узел-элемент Salary, каждый из которых имеет свое заданное значение.

Кроме того, у элемента Empl есть атрибут EmplID, имеющий значение «2».

При помощи XQuery можно генерировать текст на основе данных из БД, используя {}. В этом случае в результирующем XML выражение будет заменено на его значение.

Сгенерируем новый узел-элемент:

DECLARE @x xml

SET @x='<DepID>5</DepID><XID>8</XID>'

SELECT @x.query('<Empl><Features><Dep> {/DepID} </Dep></Features></Empl>')

Результат: <Empl><Features><Dep**><DepID>5</DepID>**</Dep></Features></Empl>

Можно, воспользовавшись функцией data() получить только значение, без названия свойства:

DECLARE @x xml

SET @x='<DepID>5</DepID><XID>8</XID>'

SELECT @x.query('<Empl><Features><Dep> { data(/DepID) } </Dep></Features></Empl>')

Результат: <Empl><Features><Dep>**5**</Dep></Features></Empl>

Комментарий: Если Вы захотите в тексте использовать {}, то нужно прописать каждую из них дважды: {{ и }}.

Для создания атрибута у узла нужно воспользоваться следующим запросом:

DECLARE @x xml

SET @x='<EmplID>5</EmplID>'

SELECT @x.query('<Empl EmplID="{ data(/EmplID) }" ></Empl>')

Или таким:

DECLARE @x xml

SET @x='<EmplID>5</EmplID>'

SELECT @x.query('<Empl EmplID="{ /EmplID }" ></Empl>')

Результат: <Empl EmplID="5" />

Обратите внимание, что здесь, несмотря на то, что нам нужно значение, а не весь узел, функцию data() можно не использовать, так как она неявно сама применится.

Пример создания одного узла из нескольких в исходном xml:

DECLARE @x xml

SET @x = '

<News>

<page>First page of news</page>

<page>Second page of news</page>

<page>Third page of news</page>

</News>'

select @x.query('

<FullNews>

{ string(/News[1]/page[1]) }

{ string(/News[1]/page[2]) }

{ string(/News[1]/page[3]) }

</FullNews>')

Другой вариант:

DECLARE @x xml

SET @x = '

<News>

<page>First page of news</page>

<page>Second page of news</page>

<page>Third page of news</page>

</News>'

select @x.query('

<FullNews>

{

for $i in /News[1]/page

return string($i)

}

</FullNews>')

Обратите внимание на то, что добавились пробелы.

**Constructed xml:**

Можно явно сконструировать xml:

DECLARE @x xml

SET @x=''

SELECT @x.query(

'element Empl

{

attribute EmplID { 2 },

text {"Name and Features Empl"},

element Name

{

text {"Ivanov Ivan"}

},

element Features

{

element Dep

{

text{"IT"}

},

element Salary

{

text{"50000"}

}

}

}'

)

Результат:

<Empl EmplID="2">

Name and Features Empl

<Name>Ivanov Ivan</Name>

<Features>

<Dep>IT</Dep>

<Salary>50000</Salary>

</Features>

</Empl>

**Работа с XML:**

Давайте на следующем примере научимся получать нужные данные из переменной или столбца типа xml.

Пример данных (из WorldDataBank):

CREATE TABLE #Countries (id int primary key, name nvarchar(100), xmlData XML)

INSERT INTO #Countries (id, name, xmlData) VALUES

(1, 'Russian Federation',

'<indicators>

<indicator name = "Population">

<comments>

<comment>Total population</comment>

</comments>

<IndicatorCode>SP.POP.TOTL</IndicatorCode>

<data year = "2011" size = "142956460"/>

<data year = "2012" size = "143178000"/>

<data year = "2013" size = "143499861"/>

</indicator>

<indicator name = "UnemploymentRate">

<comments>

<comment>Unemployment, total (% of total labor force) (modeled ILO estimate)</comment>

</comments>

<IndicatorCode>SL.UEM.TOTL.ZS</IndicatorCode>

<data year = "2011" size = "6.5"/>

<data year = "2012" size = "5.5"/>

<data year = "2013" size = "5.6"/>

</indicator>

</indicators>

')

Сейчас в таблице данные только по России и только по 2-м показателям за 3 года, Вы можете самостоятельно либо дополнить эту структуру данными, или скачать данные с сайта WorldDataBank в формате xml и разобрать xml в БД.

**Метод query():**

В общем случае метод query() принимает на вход выражение XPath и возвращает **новую XML строку** по запросу.

Выражение XPath '/indicators/indicator[2]/data' указывает, что мы хотим получить всю информацию из data для второго показателя (indicator). При этом метод query() возвращает фрагмент XML документа, содержащий всё между начальным и конечным тегами элемента "data", включая и сам элемент.

SELECT name, xmlData.query('/indicators/indicator[2]/data') AS Xml1

FROM #Countries;

|  |  |
| --- | --- |
| **Name** | **Xml1** |
| Russian Federation | <data year="2011" size="6.5" /><data year="2012" size="5.5" /><data year="2013" size="5.6" /> |

В случае XPath '/indicators/indicator[2]/data[2]' мы уже конкретно указываем, что нам нужен второй элемент из data:

|  |  |
| --- | --- |
| **name** | **Xml2** |
| Russian Federation | <data year="2012" size="5.5" /> |

Теперь мы хотим получить фрагменты xml, которые содержат слово «total», видим, что это слово содержится в двух фрагментах комментариев. Давайте попробуем такой запрос:

SELECT name,

xmlData.query ('/indicators/indicator/comments[comment[contains(., "total")]]') AS ContainsComment

FROM #Countries

Здесь нужно найти такие комментарии, где есть слово «total». «.» означает обращение к содержимому текущего элемента «comment».

Но в результате получаем только один фрагмент:

|  |  |
| --- | --- |
| **name** | **ContainsComment** |
| Russian Federation | <comments><comment>Unemployment, total (% of total labor force) (modeled ILO estimate) </comment></comments> |

Почему? Потому что xml является регистрочувствительным. Чтобы не учитывать чувствительность к регистру сделаем все буквы маленькими (приведем обрабатываемый текст к нижнему регистру):

SELECT name, xmlData.query('/indicators/indicator/comments[comment[contains(lower-case(.), "total")]]') AS ContainsComment

FROM #Countries

Теперь результат учитывает оба комментария:

|  |  |
| --- | --- |
| **name** | **ContainsComment** |
| Russian Federation | <comments><comment>Total population</comment></comments><comments><comment>Unemployment, total (% of total labor force) (modeled ILO estimate)</comment></comments> |

Теперь давайте получим xml-фрагменты, в которых значение атрибута равно заданному значению:

SELECT name,

xmlData.query('/indicators/indicator/data/.[@year = "2012"]') as Data

FROM #Countries

|  |  |
| --- | --- |
| **name** | **Data** |
| Russian Federation | <data year="2012" size="143178000" /><data year="2012" size="5.5" /> |

**Метод value():**

Метод value() позволяет извлекать из XML документа содержимое единичного элемента или атрибута с указанием его типа данных.

Мы хотим получить название первого индикатора. Давайте напишем для этого запрос:

Два следующих запроса равносильны:

SELECT name, xmlData.value('(/indicators/indicator[2]/IndicatorCode/text())[1]', 'varchar(100)') as val

FROM #Countries

SELECT name, xmlData.value('/indicators[1]/indicator[2]/IndicatorCode[1]/text()[1]', 'varchar(100)') as val

FROM #Countries

|  |  |
| --- | --- |
| **name** | **val** |
| Russian Federation | SL.UEM.TOTL.ZS |

Обратите внимание на то, что нужно явно указать, какой элемент требуется, даже если он один.

Теперь давайте получим значение конкретного атрибута, например значение атрибута size для первого индикатора и второго узла data:

SELECT name, xmlData.value('(/indicators/indicator[1]/data[2]/@size)[1]', 'int') AS size

FROM #Countries

|  |  |
| --- | --- |
| **name** | **size** |
| Russian Federation | 143178000 |

**Метод exist():**

Данный метод используется только для проверки, существует ли указанное выражение XPath.

Проиллюстрируем, проверим, есть ли в data для второго индикатора определенное значение:

SELECT name, xmlData.exist('/indicators/indicator[2]/data[@size="142956460"]') AS SizeExists

FROM #Countries

На выходе будет 0.

Можно проверить конкретное значение элемента:

SELECT name, xmlData.exist('/indicators/indicator[1]/data[1][@size = 142956460 ]') AS YearExists1

FROM #Countries

Можно сравнивать и неравенства:

SELECT name, xmlData.exist('/indicators/indicator[2]/data[@size > "45.4"]') AS SizeExists

FROM #Countries

Ответ: 1, хотя нет ни одного size больше 45.4.

Чтобы понять, что произошло, подумайте, какой будет результат выполнения следующего запроса:

SELECT CASE

WHEN '5.45' > '45.4' THEN 1

WHEN '5.45' < '45.4' THEN -1

WHEN '5.45' = '45.4' THEN 0

ELSE NULL END

Ответ: 1. Понятно ли, почему? Вспомните, как сравниваются строки.

Вариант решения проблемы:

- либо сравнивать с числом:

SELECT name, xmlData.exist('/indicators/indicator[2]/data[1]

[@size > 45.4]') AS SizeExists

FROM #Countries

- либо явно приводить тип:

SELECT name, xmlData.exist('/indicators/indicator[2]/data[1]

[@size > xs:float("45.4")]') AS SizeExistsType1

FROM #Countries

SELECT name, xmlData.exist('/indicators/indicator[2]/data[1]

[@size[xs:float(.) > xs:float("45.4")]]') AS SizeExistsType2

FROM #Countries

SELECT name, xmlData.exist('/indicators/indicator[2]/data[1]

/@size[xs:float(.) > xs:float("45.4")]') AS SizeExistsType3

FROM #Countries

**Метод modify():**

Позволяет изменять значения сразу в XML-документе на сервере.

Давайте для простоты покажем на более наглядном примере:

CREATE TABLE #test\_mdf (name nvarchar(100), xmlData xml)

INSERT INTO #test\_mdf

VALUES

('test',

'

<Empl>

<data>

<num>1</num>

<val>15</val>

</data>

<data>

<num>2</num>

<val>20</val>

</data>

</Empl>

'

)

Для начала поменяем значение 15 для num = 1 на 10.

UPDATE #test\_mdf

SET xmlData.modify

('replace value of (/Empl/ data[num[text()="1"]] /val/text())[1] with "10"')

Результат: <Empl><data><num>1</num><val>10</val></data><data><num>2</num><val>20</val></data></Empl>

Покажем, как поменять строки местами:

1-й элемент скопировать в конец:

UPDATE #test\_mdf

SET xmlData.modify('insert (/Empl/data[1]) as last into (/Empl)[1]')

Результат: <Empl><data><num>1</num><val>10</val></data><data><num>2</num><val>20</val></data>

<data><num>1</num><val>10</val></data></Empl>

Удалить 1-й элемент:

UPDATE #test\_mdf

SET xmlData.modify('delete (/Empl/data[1])')

Результат:

<Empl><data><num>2</num><val>20</val></data><data><num>1</num><val>10</val></data></Empl>

Задание: подумайте, как изменить значение атрибута?

Пример данных:

CREATE TABLE #test\_modify (name nvarchar(100), xmlData xml)

INSERT INTO #test\_modify

VALUES

('test',

'

<Empl>

<data num = "1" val = "15"/>

<data num = "2" val = "20"/>

</Empl>

'

)

Ответ:

UPDATE #test\_modify

SET xmlData.modify('replace value of

(/Empl/ data[num[text()="1"]] /val/text())[1] with "10"')

Давайте теперь добавим атрибут:

UPDATE #test\_modify

SET xmlData.modify('

insert

(

attribute EmplID { "8" }

)

into (/Empl)[1]

')

Результат: <Empl EmplID="8">

<data><num>2</num><val>20</val></data><data><num>1</num><val>10</val></data></Empl>

**Метод nodes():**

Разбивает XML на несколько поддеревьев, в соответствии с указанным XPath.

Пример:

SELECT name, T.query('.') AS val, T.value('@year', 'int') Year, T.value('@size', 'float') Pop

FROM #Countries C

CROSS APPLY C.xmlData.nodes('/indicators/indicator/data') data(T)

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **val** | **Year** | **Pop** |
| Russian Federation | <data year="2011" size="142956460" /> | 2011 | 142956460 |
| Russian Federation | <data year="2012" size="143178000" /> | 2012 | 143178000 |
| Russian Federation | <data year="2013" size="143499861" /> | 2013 | 143499861 |
| Russian Federation | <data year="2011" size="6.5" /> | 2011 | 6.5 |
| Russian Federation | <data year="2012" size="5.5" /> | 2012 | 5.5 |
| Russian Federation | <data year="2013" size="5.6" /> | 2013 | 5.6 |

На сервере данные XML могут быть сгенерированы из таблиц и результатов запросов при помощи выражения FOR XML в команде SQL. Такой способ хорошо подходит для обмена данными в приложениях веб-сервисов.

Обратной операцией для FOR XML является функция генератора реляционного набора строка OpenXML, которая заносит значения из данных XML в столбцы набора строк.

Sp\_prepare….

**xQuery expressions:**

ms-help://MS.SQLCC.v10/MS.SQLSVR.v10.en/s10de\_7xqury/html/a6150940-4d1c-4bff-a8f9-eda667fc45eb.htm