OTUS

Онлайн образование

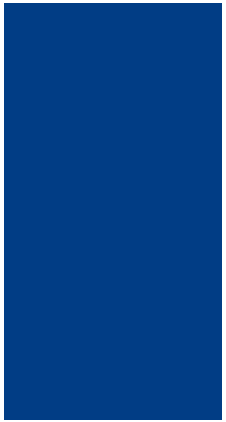
otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Тема вебинара

XML и JSON в MS SQL Server



Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в telegram

Задаем вопрос
в чат или голосом

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Маршрут вебинара

- XML и JSON что это?
- Как использовать?
- Выборка в формате XML и JSON
- Выборка из XML и JSON
- Рефлексия

Цели вебинара

После занятия вы сможете

1. Уметь выбирать тип XML и JSON
2. Выгружать данны в XML и JSON

Смысл

Зачем вам это уметь?

1. Сможете выбирать данные из XML и JSON полей

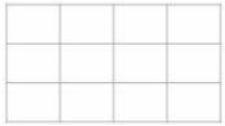
2. Формировать правильный XML и JSON

XML и JSON

Виды данных

Структурированные

- Реляционная модель
- SQL (**structured** query language)
- Определяем схему, колонки, типы данных и тп
- Информация, уже подготовленная к анализу



Неструктурированные

- Не имеет заранее определенной структуры данных
- Данные на естественном языке, текст, статьи
- Логи

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut non ex eu diam commodo ornare. Etiam magna ipsum, viverra et sollicitudin vitae, iaculis a ex. Suspendisse convallis placerat leo, quis lacinia orci faucibus id.

Полуструктурированнные (слабоструктурированные)

- Данные, понятные для машинного распознавания, но все еще требующие неких преобразований для получения конкретной информации из неё
- Нет схемы*
- Имеют некоторые организационные свойства, облегчающие их анализ
- JSON
- XML

```
{
   "firstName": "Иван",
   "lastName": "Иванов",
   "address": {
       "streetAddress": "Московское ш., 101, кв.101",
       "city": "Ленинград",
       "postalCode": "101101"
   },
   "phoneNumbers": [
       "812 123-1234",
       "916 123-4567"
   ]
}
```

Что такое XML, JSON

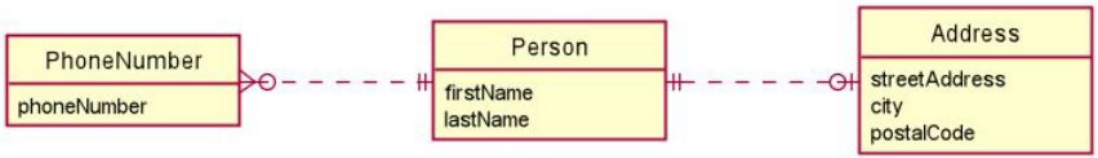
Полуструктурированные данные (semi-structured data), иерархические

XML — eXtensible Markup Language

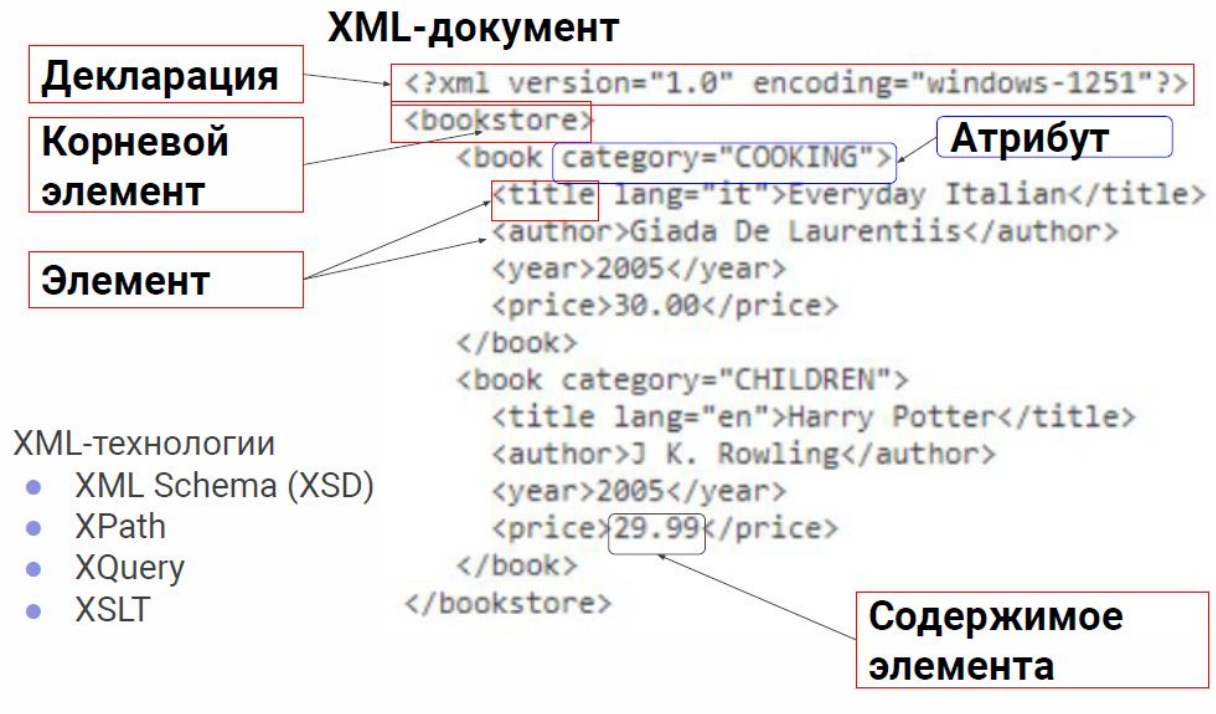
JSON — JavaScript Object Notation

```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>
```

```json
{
    "firstName": "Иван",
    "lastName": "Иванов",
    "address": {
        "streetAddress": "Московское ш., 101, кв.101",
        "city": "Ленинград",
        "postalCode": "101101"
    },
    "phoneNumbers": [
        "812 123-1234",
        "916 123-4567"
    ]
}
```



XML



https://xml.readthedocs.io/xml-intro.html

Валидный XML

Основные правила

- Должен быть корневой элемент
- Все теги должны быть закрыты

`<title></title>` или `<title/>`

- Должна быть корректная вложенность тегов

```
<book><title></book></title>
```

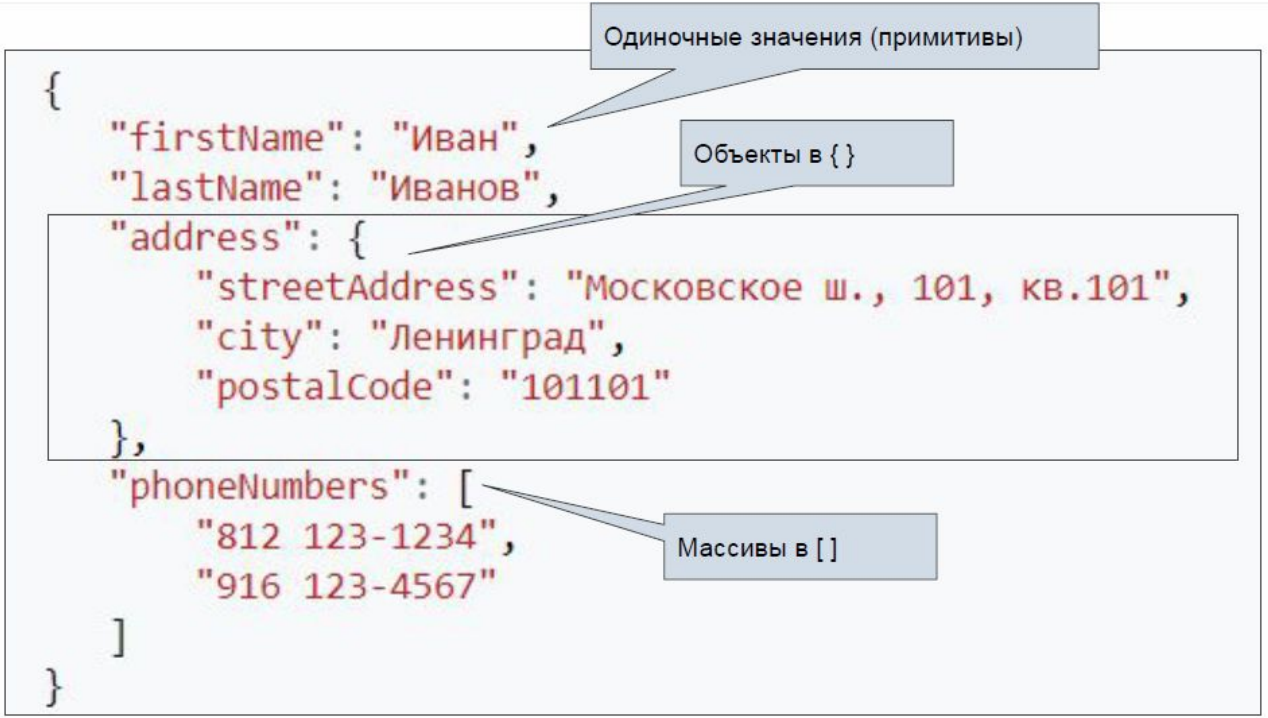
- Имена тегов регистрозависимы

```
<book></Book>
```

- Значения атрибутов в кавычках
- Спецсимволы (сущности):

< &lth;	>> &gth;	& &

JSON



Типы значений: число, строка, boolean, массив, объект, null

Что такое XML, JSON

XML - eXtensible Markup Language

https://en.wikipedia.org/wiki/SQL/XML
Стандарт SQL\XML (2003)
Документация SQL Server

```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>
```

JSON - JavaScript Object Notation

https://www.json.org/json-en.html
https://tools.ietf.org/html/rfc7159
Стандарт SQL/JSON (2016)
Документация SQL Server

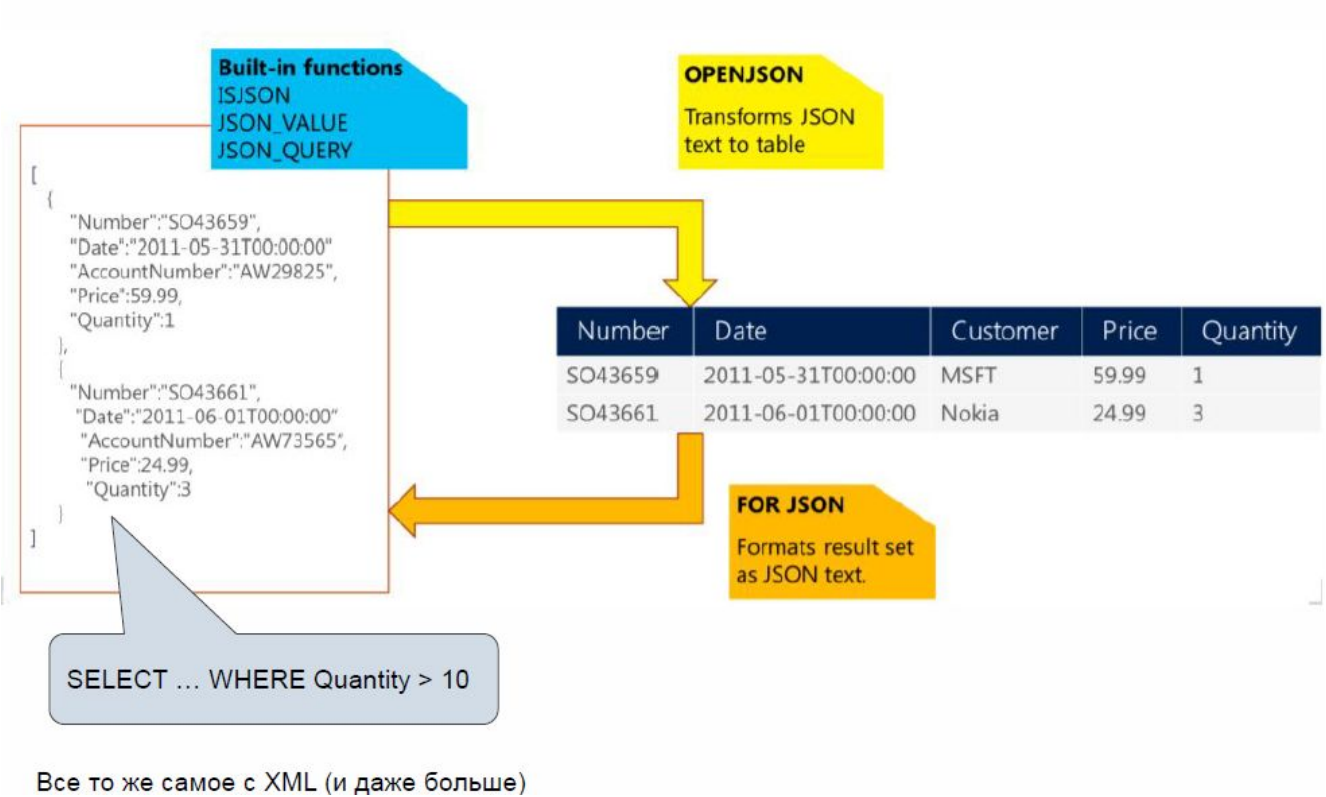
```json
{
   "firstName": "Иван",
   "lastName": "Иванов",
   "address": {
       "streetAddress": "Московское ш., 101, кв.101",
       "city": "Ленинград",
       "postalCode": "101101"
   },
   "phoneNumbers": [
       "812 123-1234",
       "916 123-4567"
   ]
}
```

Для чего используется XML и JSON?

Для чего используется XML и JSON?

- экспорт / импорт данных, интеграция, SOAP
- сразу отдавать JSON на frontend, REST
- хранение настроек, конфигурации
- хранение не структурированных (полуструктурированных) данных, денормализация, NoSQL
- генерирование документов (отчетов) XML + XSLT
- ...

XML и JSON в MS SQL



Все то же самое с XML (и даже больше)

https://learn.microsoft.com/en-us/sql/relational-databases/json/json-data-sql-server?view=sql-server-ver16

XML и JSON в MS SQL

XML

- **SQL Server 2000**
  - SELECT … FOR XML
  - OPENXML
- **SQL Server 2005**
  - Тип XML
  - Схемы (XML Schema)
  - Поддержка XQuery

JSON

- **SQL Server 2016**

До 2016:

- Парсить вручную с помощью строковых функций (если JSON простой и постоянный)
- CLR (C#)
- FUNCTION dbo.parseJSON –

https://www.red-gate.com/simple-talk/sql/t-sql-programming/consuming-json-strings-in-sql-server/

Table(query) → XML/JSON

Table в xml/json

Id	SupplierInfo.Name	SupplierInfo.Category	Contact.Primary	Contact.Alternate	WebsiteURL	CityName
1	A Datum Corporation	Novelty Goods Supplier	Reio Kabin	Oliver Kivi	http://www.adatum.com	Zionsville
2	Contoso, Ltd.	Novelty Goods Supplier	Hanna Mihhailov	Paulus Lippmaa	http://www.contoso.com	Greenbank

```xml
<Suppliers>
    <Supplier Id="1">
        <SupplierInfo>
            <Name>A Datum Corporation</Name>
            <Category>Novelty Goods Supplier</Category>
        </SupplierInfo>
        <Contact>
            <Primary>Reio Kabin</Primary>
            <Alternate>Oliver Kivi</Alternate>
        </Contact>
        <WebsiteURL>http://www.adatum.com</WebsiteURL>
        <CityName>Zionsville</CityName>
    </Supplier>
    <Supplier Id="2">
        <SupplierInfo>
            <Name>Contoso, Ltd.</Name>
            <Category>Novelty Goods Supplier</Category>
        </SupplierInfo>
        <Contact>
            <Primary>Hanna Mihhailov</Primary>
            <Alternate>Paulus Lippmaa</Alternate>
        </Contact>
        <WebsiteURL>http://www.contoso.com</WebsiteURL>
        <CityName>Greenbank</CityName>
    </Supplier>
</Suppliers>
```

```json
{
    "Suppliers": [
        {
            "Id": 1,
            "SupplierInfo": {
                "Name": "A Datum Corporation",
                "Category": "Novelty Goods Supplier"
            },
            "Contact": {
                "Primary": "Reio Kabin",
                "Alternate": "Oliver Kivi"
            },
            "WebsiteURL": "http:\/\/www.adatum.com",
            "CityName": "Zionsville"
        },
        {
            "Id": 2,
            "SupplierInfo": {
                "Name": "Contoso, Ltd.",
                "Category": "Novelty Goods Supplier"
            },
            "Contact": {
                "Primary": "Hanna Mihhailov",
                "Alternate": "Paulus Lippmaa"
            },
            "WebsiteURL": "http:\/\/www.contoso.com",
            "CityName": "Greenbank"
        }
    ]
}
```

Table в xml/json

```
SELECT … FROM …
FOR XML  PATH | AUTO | RAW | EXPLICIT
FOR JSON PATH | AUTO [WITHOUT_ARRAY_WRAPPER, INCLUDE_NULL_VALUES]
```

XML/JSON

- PATH

  Создает структуру на основе псевдонимов колонок

- AUTO

  Создает структуру на основе иерархии таблиц (join)

XML

- RAW

  Каждая строка - элемент с атрибутами или вложенными элементами.

- EXPLICIT

  ElementName!TagNumber!AttributeName!Directive

XML/JSON → Table

XPATH / JSONPATH
Xquery

Xml/JSON в Table

Преобразование XML/JSON в табличное представление

```xml
<Suppliers>
    <Supplier Id="1">
        <SupplierInfo>
            <Name>A Datum Corporation</Name>
            <Category>Novelty Goods Supplier</Category>
        </SupplierInfo>
        <Contact>
            <Primary>Reio Kabin</Primary>
            <Alternate>Oliver Kivi</Alternate>
        </Contact>
        <WebsiteURL>http://www.adatum.com</WebsiteURL>
        <CityName>Zionsville</CityName>
    </Supplier>
    <Supplier Id="2">
        <SupplierInfo>
            <Name>Contoso, Ltd.</Name>
            <Category>Novelty Goods Supplier</Category>
        </SupplierInfo>
        <Contact>
            <Primary>Hanna Mihhailov</Primary>
            <Alternate>Paulus Lippmaa</Alternate>
        </Contact>
        <WebsiteURL>http://www.contoso.com</WebsiteURL>
        <CityName>Greenbank</CityName>
    </Supplier>
</Suppliers>
```

```json
{
    "Suppliers": [
        {
            "Id": 1,
            "SupplierInfo": {
                "Name": "A Datum Corporation",
                "Category": "Novelty Goods Supplier"
            },
            "Contact": {
                "Primary": "Reio Kabin",
                "Alternate": "Oliver Kivi"
            },
            "WebsiteURL": "http:\/\/www.adatum.com",
            "CityName": "Zionsville"
        },
        {
            "Id": 2,
            "SupplierInfo": {
                "Name": "Contoso, Ltd.",
                "Category": "Novelty Goods Supplier"
            },
            "Contact": {
                "Primary": "Hanna Mihhailov",
                "Alternate": "Paulus Lippmaa"
            },
            "WebsiteURL": "http:\/\/www.contoso.com",
            "CityName": "Greenbank"
        }
    ]
}
```

Id	SupplierInfo.Name	SupplierInfo.Category	Contact.Primary	Contact.Alternate	WebsiteURL	CityName
1	A Datum Corporation	Novelty Goods Supplier	Reio Kabin	Oliver Kivi	http://www.adatum.com	Zionsville
2	Contoso, Ltd.	Novelty Goods Supplier	Hanna Mihhailov	Paulus Lippmaa	http://www.contoso.com	Greenbank

Openxml

Конвертирование XML в таблицу
(можно и через XQuery - об этом далее)

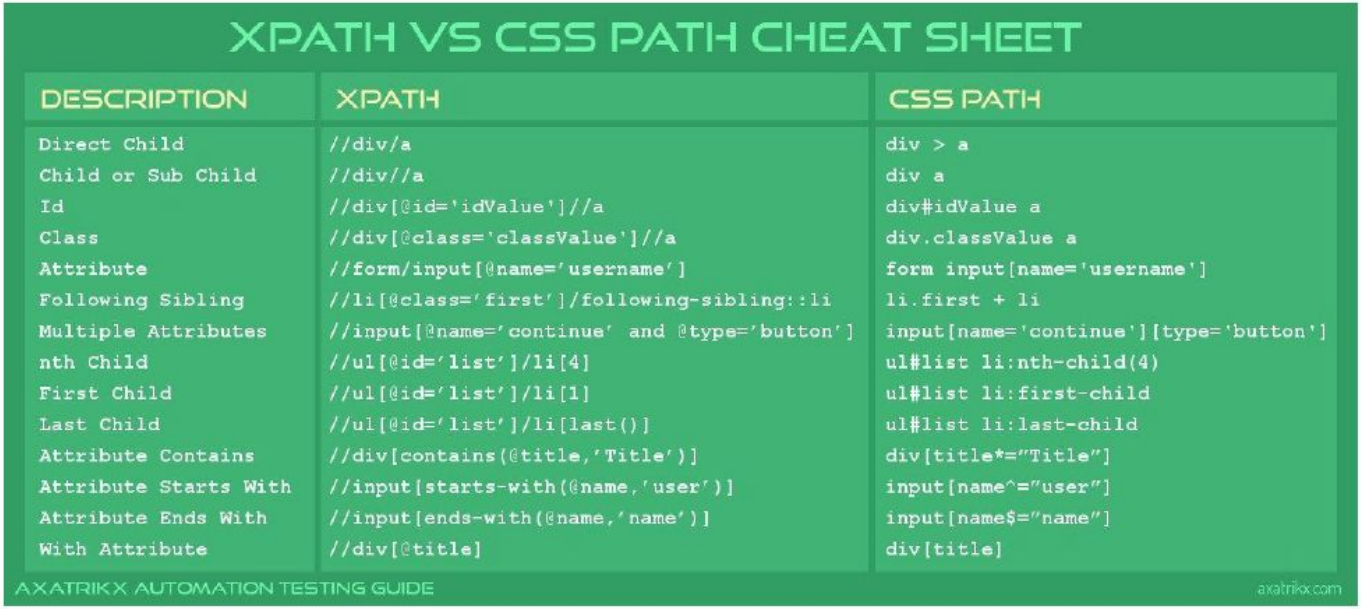
```
EXEC sp_xml_preparedocument @handle OUTPUT, @xml

SELECT *
FROM OPENXML (@handle,'/Customer/Order', 2)
WITH (
  OrderID     int          '../@OrderID',
  CustomerID  varchar(10)  '../@CustomerID',
  OrderDate   datetime     '../@OrderDate',
  ProdID      int          '@ProductID',
  Qty         int          'Quantity');


EXEC sp_xml_removedocument @handle
```

OrderID	CustomerID	OrderDate	ProdID	Qty
10248	VINET	1996-07-04 00:00:00.000	11	12
10248	VINET	1996-07-04 00:00:00.000	42	10
10283	LILAS	1996-08-16 00:00:00.000	72	3

Xpath

XPath (XML Path Language) — язык запросов к XML-документам

```
/tag1/tag2[@attribute="value"]
```



XPATH VS CSS PATH CHEAT SHEET

DESCRIPTION	XPATH	CSS PATH
Direct Child	//div/a	div > a
Child or Sub Child	//div//a	div a
Id	//div[@id='idValue']//a	div#idValue a
Class	//div[@class='classValue']//a	div.classValue a
Attribute	//form/input[@name='username']	form input[name='username']
Following Sibling	//li[@class='first']/following-sibling::li	li.first + li
Multiple Attributes	//input[@name='continue' and @type='button']	input[name='continue'][type='button']
nth Child	//ul[@id='list']/li[4]	ul#list li:nth-child(4)
First Child	//ul[@id='list']/li[1]	ul#list li:first-child
Last Child	//ul[@id='list']/li[last()]	ul#list li:last-child
Attribute Contains	//div[contains(@title,'Title')]	div[title*="Title"]
Attribute Starts With	//input[starts-with(@name,'user')]	input[name^="user"]
Attribute Ends With	//input[ends-with(@name,'name')]	input[name$="name"]
With Attribute	//div[@title]	div[title]

AXATRIKX AUTOMATION TESTING GUIDE axatrikx.com

Xpath

Выражение XPath	Результат
book	Все элементы “book”
/bookstore	Корневой элемент “bookstore”
/bookstore/book/author	Все author внутри book внутри bookstore
/bookstore//author	Все author внутри любых подэлементов bookstore

```
<?xml version="1.0" encoding="windows-1251"?>
<bookstore>
   <book category="COOKING">
     <title lang="it">Everyday Italian</title>
     <author>Giada De Laurentiis</author>
     <year>2005</year>
     <price>30.00</price>
   </book>
   <book category="CHILDREN">
     <title lang="en">Harry Potter</title>
     <author>J K. Rowling</author>
     <year>2005</year>
     <price>29.99</price>
   </book>
</bookstore>
```

Xpath

Выражение XPath	Результат
/bookstore/book[1]	Первая книга
/bookstore/book[last()]	Последняя книга
//book[@category='COOKING']	book с категорией COOKING
//book[year > 2000]	book, где year больше 2000
//book[year > 2000]/title	Выбирает все элементы title элементов book, где year больше 2000

```
<?xml version="1.0" encoding="windows-1251"?>
<bookstore>
   <book category="COOKING">
     <title lang="it">Everyday Italian</title>
     <author>Giada De Laurentiis</author>
     <year>2005</year>
     <price>30.00</price>
   </book>
   <book category="CHILDREN">
     <title lang="en">Harry Potter</title>
     <author>J K. Rowling</author>
     <year>2005</year>
     <price>29.99</price>
   </book>
</bookstore>
```

XPath в примерах

http://www.zvon.org/xxl/XPathTutorial/General_rus/examples.html

Demo

OPENXML

Импорт из файла

```
DECLARE @xmlDocument  xml

SELECT @xmlDocument = BulkColumn
FROM OPENROWSET(BULK 'file.xml', SINGLE_CLOB) as data
```

Openjson

```
SELECT *
FROM OPENJSON ( @json )
WITH (
              Number   varchar(200)   '$.Order.Number',
              Date     datetime       '$.Order.Date',
              Customer varchar(200)   '$.AccountNumber',
              Quantity int            '$.Item.Quantity',
              [Order]  nvarchar(MAX)  AS JSON
 )
```

```
DECLARE @json NVARCHAR(MAX) = N'[
  {
    "Order": {
      "Number":"SO43659",
      "Date":"2011-05-31T00:00:00"
    },
    "AccountNumber":"AW29825",
    "Item": {
      "Price":2024.9940,
      "Quantity":1
    }
  },
  {
    "Order": {
      "Number":"SO43661",
      "Date":"2011-06-01T00:00:00"
    },
    "AccountNumber":"AW73565",
    "Item": {
      "Price":2024.9940,
      "Quantity":3
    }
  }
]'
```

Number	Date	Customer	Quantity	Order
SO43659	2011-05-31T00:00:00	AW29825	1	{"Number":"SO43659","Date":"2011-05-31T00:00:00"}
SO43661	2011-06-01T00:00:00	AW73565	3	{"Number":"SO43661","Date":"2011-06-01T00:00:00"}

JsonPath

JSON_QUERY(), JSON_VALUE()

Выражение JSON Path	Описание
$	Сам JSON-объект (корень, контекст)
$.property1	Свойство
$.property1[5]	Шестой элемент в массиве
$.property1.property2[5].property3	
lax$.property1	Если путь не найден, то возвращается пустое значение. По умолчанию.
strict$.property1	Если путь не найден, то ошибка

Demo

OPENJSON

Тип данных XML

Переменная

```
DECLARE @xml_doc XML
```

Таблица

```
CREATE TABLE table1 (
  id INTEGER,
  xml_col XML)
```

Хранимая процедура

```
CREATE PROCEDURE proc1
  (indoc XML, @outdoc XML OUTPUT)
```

Функция

```
CREATE FUNCTION func1 (@x NVARCHAR(max))
RETURNS XML
```

Тип данных XML

- Хранит валидный XML (можно со схемой
- Документы или фрагменты
- Разрешены узлы в виде простого текста
- Разрешены NULL и пустые строки
- Разрешены CDATA
- Поддержка XQuery и XPath 2.0

Тип данных XML

- Это не символьный тип
- Не поддерживает сравнения (кроме с NULL)
  - нет сравнения на равенство
  - нет ORDER BY, GROUP BY
- Не может использоваться в первичных ключах
- Не может использоваться в UNIQUE
- Не может быть объявлен с COLLATE
  - использует кодировку XML
  - храниться как UNICODE UCS-2

XML Data Type
JSON / XML validation

Методы типа данных XML

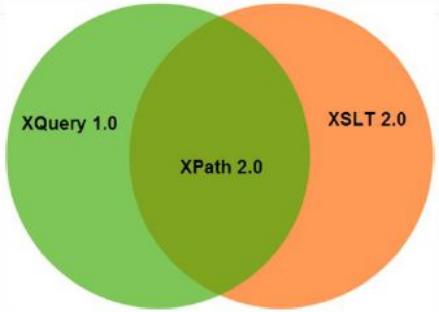
Методы в T-SQL для работы с XML:

- **query()** – извлечение XML фрагментов из XML документов;
- **value()** – извлечение значений конкретных узлов или атрибутов XML документов;
- **exist()** – проверки существования узла или атрибута. Возвращает 1, если узел или атрибут найден, и 0, если не найден;
- **modify()** – изменяет XML документ;
- **nodes()** – разделяет XML документ на несколько строк по узлам. Используется как вход в "XQuery-подзапросах"

Принимают на вход XPath или XQuery

XQuery

- Аналог SQL для XML
- Типизированный, декларативный, регистрозависимый
- Использует XPath
- Похож на SQL (FLWOR-выражения)
  for let where order by return
  - **for** – задает переменную для цикла
  - **let** – присваивание секвенции
  - **where** – задает фильтр для выбираемых данных
  - **order by** – указывает порядок сортировки
  - **return** – указывает выбираемые значения



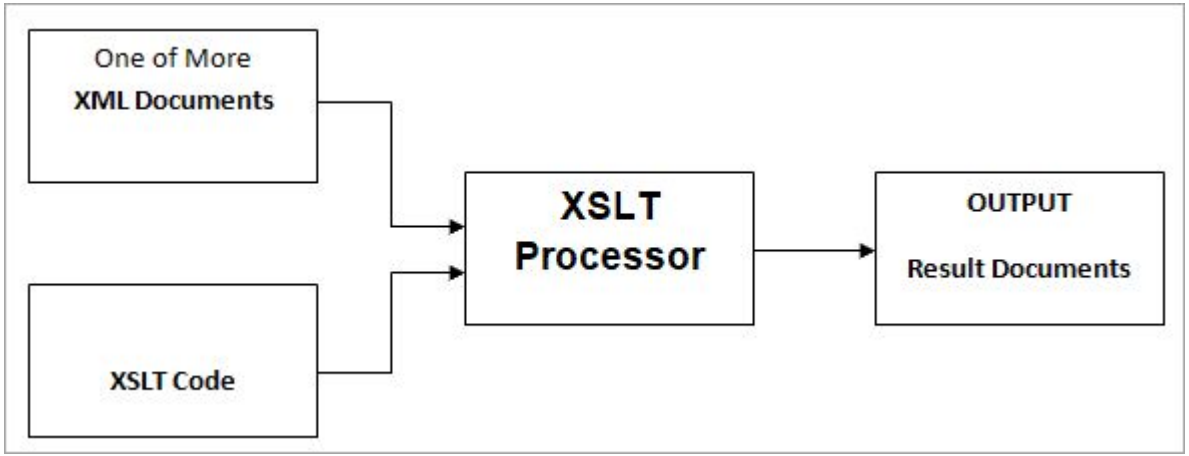

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

- https://ppt-online.org/23233
  (oracle, но хорошие простые примеры XQuery)

Demo
XML Data Type
XQuery

XSLT = XML Трасформация

XSLT - eXtensible Stylesheet Language Transformation



Demo XSLT

Функции JSON

Функции для работы с JSON

- **ISJSON()** проверяет соответствует ли строка формату JSON
- **JSON_ARRAY()** формирует JSON массив из списка значений
- **JSON_MODIFY()** меняет данные в JSONe и возвращает измененный JSON
- **JSON_OBJECT()** создает JSON объект из списка значений
- **JSON_PATH_EXISTS()** проверяет наличие пути в JSON с SQL Server 2022
- **JSON_QUERY()** получает JSON объект из JSON
- **JSON_VALUE()** получает значение из JSON объекта

Demo

json value
json query

Индексы

XML

- Должен быть кластеризованный индекс
- Используется в запросах XQuery

- Первичный xml-индекс (B+ дерево с разобранным xml)
- Вторичные индексы (PATH, PROPERTY, VALUE)

- Документация SQL Server — XML Indexes
- Статья "Indexing XML Data Stored in a Relational Database" (презентация)
- Getting Started With XML Indexes

JSON

- Специальных индексов нет

Индексы

XML

- Должен быть кластеризованный индекс
- Используется в запросах XQuery

- Первичный xml-индекс (B+ дерево с разобранным xml)
- Вторичные индексы (PATH, PROPERTY, VALUE)

- Документация SQL Server — XML Indexes
- Статья "Indexing XML Data Stored in a Relational Database" (презентация)
- Getting Started With XML Indexes

JSON

- Специальных индексов нет

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Рефлексия

Вопросы для проверки

1. Как хранится XML, JSON? Индексы на значения в XML, JSON?
2. Как получить результат запроса в виде XML, JSON?
3. Как выбрать поле из XML, JSON?
4. Как сделать валидацию схему?
5. Как преобразовать XML, JSON в таблицу?

Рефлексия

С какими основными мыслями
и инсайтами уходите с вебинара?

Как будете применять на практике то,
что узнали на вебинаре?

Заполните, пожалуйста, опрос о занятии по ссылке в чате