



ОНЛАЙН-ОБРАЗОВАНИЕ

Не забыть включить
запись!

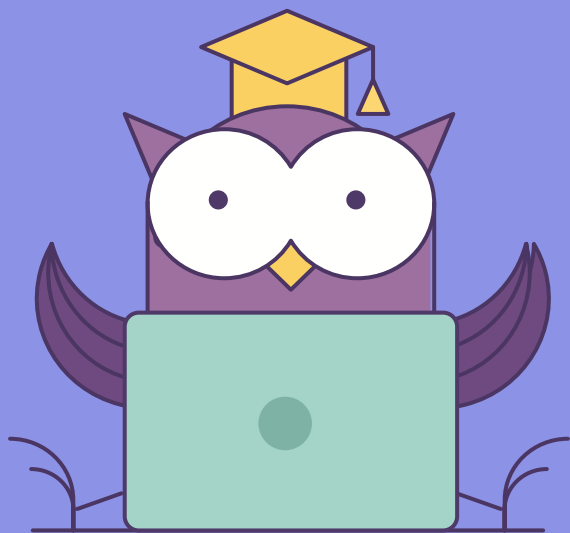


XML, JSON


Курс “MS SQL Server разработчик”
2022-03



Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте  если все хорошо
Или напишите, какие есть проблемы

01

XML
JSON

Напишите в чат знакомы ли вы с XML и JSON?

“-” - ни с чем не знакомы

“xml” - знакомы с xml

“json” - знакомы с json

“xml json” - знакомы и с xml и с json

“xml sql” - работали в SQL Server с xml

“json sql” - работали в SQL Server с json

“xml json sql” - работали в SQL Server и с xml и с json

Структурированные, полуструктурированные и неструктурированные данные

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.

Полуструктурированные (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"
  ]
}
```

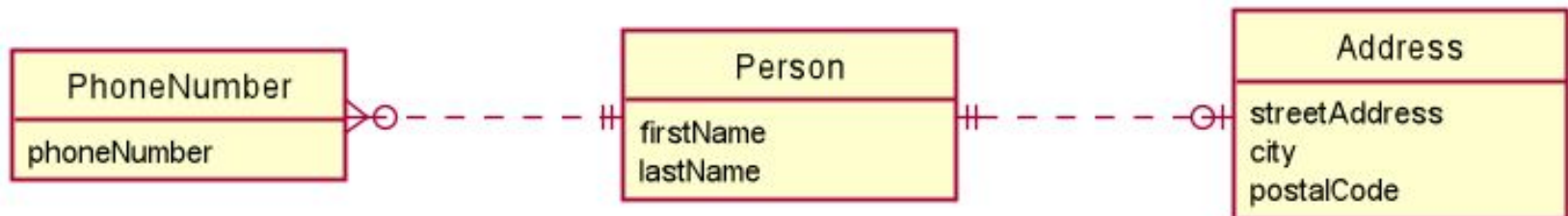
Полуструктурированные данные (semi-structured data)

XML — eXtensible Markup Language

JSON — JavaScript Object Notation

```
<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>
```

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



XML — eXtensible Markup Language

- <https://www.w3.org/TR/xml11/>
- Стандарт [SQL/XML](#) (SQL:2003 - ...)
- Документация SQL Server — [XML Data \(SQL Server\)](#)

JSON — JavaScript Object Notation

- <https://www.json.org>
- [RFC 7159](#) — JSON Data Interchange Format
- Стандарт [SQL/JSON](#) (SQL:2016)
- Документация SQL Server — [JSON data in SQL Server](#)

Для чего используется XML, JSON?
(да еще и в реляционной СУБД)



Для чего используется XML, 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
- ...

Built-in functionsISJSON
JSON_VALUE
JSON_QUERY**OPENJSON**

Transforms JSON text to table

```
[
  {
    "Number": "SO43659",
    "Date": "2011-05-31T00:00:00",
    "AccountNumber": "AW29825",
    "Price": 59.99,
    "Quantity": 1
  },
  {
    "Number": "SO43661",
    "Date": "2011-06-01T00:00:00",
    "AccountNumber": "AW73565",
    "Price": 24.99,
    "Quantity": 3
  }
]
```

Number	Date	Customer	Price	Quantity
SO43659	2011-05-31T00:00:00	MSFT	59.99	1
SO43661	2011-06-01T00:00:00	Nokia	24.99	3

FOR JSON

Formats result set as JSON text.

SELECT ... WHERE Quantity > 10

Все то же самое с XML (и даже больше)

<https://docs.microsoft.com/en-us/sql/relational-databases/json/json-data-sql-server>

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 — [Consuming JSON Strings in SQL Server](#)

XML-документ

Декларация

Корневой
элемент

Элемент

Атрибут

```
<?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>
```

Содержимое
элемента

XML-технологии

- XML Schema (XSD)
- XPath
- XQuery
- XSLT

Коротко об XML — <https://xml.readthedocs.io/xml-intro.html>

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):

<code><</code>	<code>></code>	<code>&</code>
<code>&lt;</code>	<code>&gt;</code>	<code>&amp;</code>

Одиночные значения (примитивы)

Объекты в { }

Массивы в []

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

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

02

Table (query) → XML/JSON

Получение данных из таблиц (запросов) в виде 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

```
<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>
```

```
{
  "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"
    }
  ]
}
```

ДЕМО

FOR 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

03


XML/JSON → Table

XPATH / JSONPATH

XQuery

Преобразование XML/JSON в табличное представление

```
<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>
```



```
{
  "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

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

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

XPATH VS CSS PATH CHEAT SHEET

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

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>
```

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>
```

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

XPath в примерах

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

ДЕМО

OPENXML



```
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"}

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

<https://docs.microsoft.com/ru-ru/sql/t-sql/functions/json-functions-transact-sql>

<https://docs.microsoft.com/ru-ru/sql/relational-databases/json/json-path-expressions-sql-server>

ДЕМО

OPENJSON



04

XML Data Type JSON / XML validation

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
```

- 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

- 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

ДЕМО

XML data type

XQuery



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

http://www.sql-tutorial.ru/ru/book_xml_data_type_methods/page1.html

- Аналог SQL для XML
- Типизированный, декларативный, регистрозависимый
- Использует XPath
- Похож на SQL (FLWOR-выражения)

for let where order by return

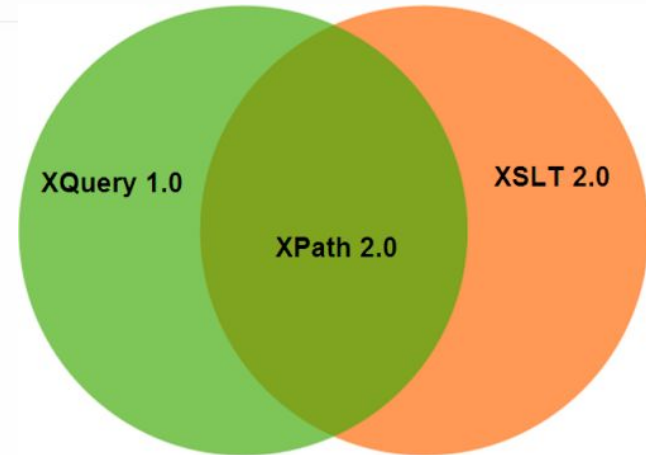
- **for** – задает переменную для цикла
- **let** – присваивание секвенции
- **where** – задает фильтр для выбираемых данных
- **order by** – указывает порядок сортировки
- **return** – указывает выбираемые значения

```
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)

- Специального типа данных нет
- Для валидации только функция ISJSON()
- JSON Schema в природе существует, но в SQL Server не поддерживается
- Примеры схем: <https://json-schema.org/learn/>

ДЕМО

json value

json query



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

05

Рефлексия

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

1. Как хранится XML, JSON? Индексы на значения в XML, JSON?

xml: Тип XML. Специальные индексы.

json: nvarchar, varchar

2. Как получить результаты запроса в виде XML, JSON?

xml: FOR XML RAW, FOR XML AUTO, FOR XML PATH

json: FOR JSON AUTO, FOR JSON PATH

3. Как выбрать поле из XML, JSON?

xml: query(), value()

json: JSON_VALUE, JSON_QUERY

4. Как сделать валидацию схемы?

xml: CREATE XML SCHEMA

json: Полноценной валидации нет. Только ISJSON

5. Как преобразовать XML, JSON в таблицу?

OPENXML, OPENJSON

XML

- [XML Best Practices for Microsoft SQL Server 2005](#)
- [Примеры массового импорта и экспорта XML-документов \(SQL Server\)](#)
- [Bulk Inserts via TSQL in SQL Server](#)

JSON

- [Работаем с JSON в SQL Server 2016](#)
- [SQL Server 2017 JSON](#)
- [Searching Complex JSON Data](#)
- [JSON Usage and Performance in SQL Server 2016 Bert Wagner](#)
- [Importing JSON Data from Web Services and Applications into SQL Server](#)

06

Домашнее задание

1. В личном кабинете есть файл StockItems.xml.

Это данные из таблицы Warehouse.StockItems.

Преобразовать эти данные в плоскую таблицу с полями, аналогичными Warehouse.StockItems.

Поля: StockItemName, SupplierID, UnitPackageID, OuterPackageID, QuantityPerOuter, TypicalWeightPerUnit, LeadTimeDays, IsChillerStock, TaxRate, UnitPrice

Загрузить эти данные в таблицу Warehouse.StockItems. Существующие записи в таблице обновить, отсутствующие добавить (сопоставлять записи по полю StockItemName).

2. Выгрузить данные из таблицы Warehouse.StockItems в xml-файл такой же структуры, как StockItems.xml.

Примечания к заданиям 1, 2:

- Если с выгрузкой в файл будут проблемы, то можно сделать просто SELECT с результатом в виде XML. Пример экспорта/импорта в файл можно посмотреть [здесь](#).
- Если у вас в проекте предусмотрен экспорт/импорт в XML, то можете взять свой XML и свои таблицы.
- Если с этим XML вам будет скучно, то можете взять любые открытые данные и импортировать их в таблицы (например, с <https://data.gov.ru>).

3. В таблице `Warehouse.StockItems` в колонке `CustomFields` есть данные в JSON.

Написать `SELECT` для вывода:

- `StockItemID`
- `StockItemName`
- `CountryOfManufacture` (из поля `CustomFields`)
- `FirstTag` (из поля `CustomFields` первое значение из массива `Tags`)

```
{
  "CountryOfManufacture": "China",
  "Tags": [
    "Limited Stock"
  ]
}
```

```
{
  "CountryOfManufacture": "China",
  "Tags": [],
  "Range": "Teens/Young Adult"
}
```

```
{
  "CountryOfManufacture": "China",
  "Tags": [
    "Radio Control",
    "Realistic Sound",
    "Vintage"
  ],
  "MinimumAge": "10"
}
```

StockItemID	StockItemName	CountryOfManufacture	FirstTag
64	RC vintage American toy coup...	China	Radio Control
93	"The Gu" red shirt XML tag t...	China	NULL
158	10 mm Double sided bubble wr...	China	Limited Stock

Выглядеть должно примерно так.

Записей должно быть больше.

4. Найти в таблице Warehouse.StockItems строки, где есть тэг "Vintage".

Вывести:

- StockItemID
- StockItemName
- (опционально) все теги (из CustomFields) через запятую в одном поле

StockItemID	StockItemName	Tags
64	RC vintage American toy coupe with remote control (Red) 1/50 scale	Radio Control,Realistic Sound,Vintage
65	RC vintage American toy coupe with remote control (Black) 1/50 scale	Radio Control,Realistic Sound,Vintage
73	Ride on vintage American toy coupe (Red) 1/12 scale	Vintage,So Realistic
74	Ride on vintage American toy coupe (Black) 1/12 scale	Vintage,So Realistic

Тэги искать в поле CustomFields, а не в Tags.

Запрос написать через функции работы с JSON.

Для поиска использовать равенство, использовать LIKE запрещено.

Должно быть в таком виде:

... where ... = 'Vintage'

Так принято не будет:

... where ... Tags like '%Vintage%'

... where ... CustomFields like '%Vintage%'

Выглядеть должно примерно так.

```
{
  "CountryOfManufacture": "China",
  "Tags": [
    "Radio Control",
    "Realistic Sound",
    "Vintage"
  ],
  "MinimumAge": "10"
}
```

**Пройдите, пожалуйста
опрос**

**Спасибо
за внимание!**

