

# Welcome to SQL Saturday Denmark

---



# JSON & XML

SQL Server 2016

since SQL 2016 it is important to weight respective advantages



# Thanks you our PLATINUM sponsors

---



# Thanks you our GOLD and SILVER sponsors

---



Microsoft



SQL POWERHOUSE



PURE STORAGE

by

CLOUDIO

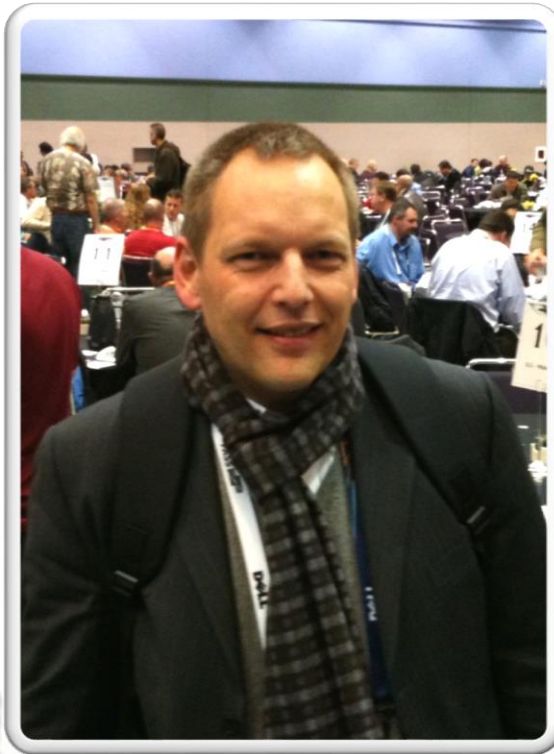
NEXTAGENDA  
BUSINESS INTELLIGENCE LØSNINGER



imshealth | Rehfeld 



# About me



Alexander Karl

.net - CDE 

SQL + BI Consultant

**Microsoft**  
CERTIFIED  
Trainer

**Microsoft**  
CERTIFIED  
Solutions Associate  
SQL Server 2012/2014

**Microsoft**  
CERTIFIED  
Solutions Expert  
Data Platform



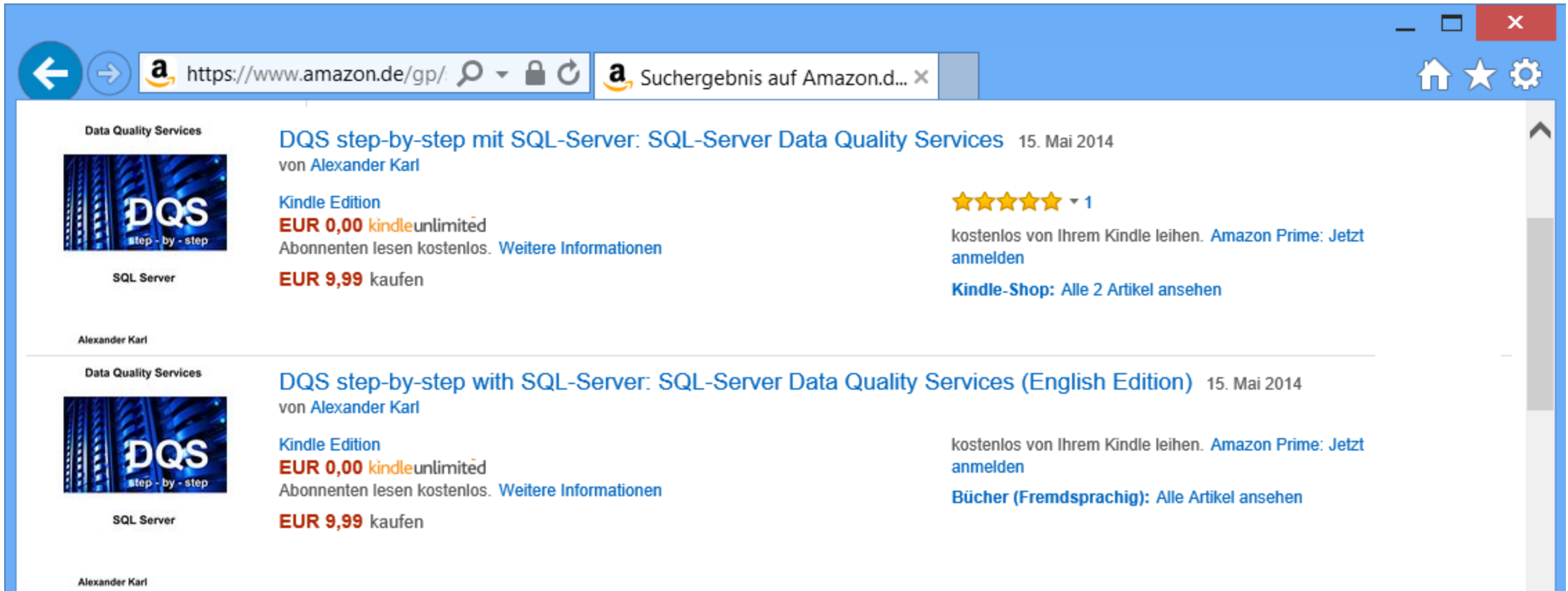
# publications



The screenshot shows a web browser window with the URL <https://www.video2brain.com/de/videotraining>. The page header includes the video2brain logo (a lynda.com brand) and navigation links: [Gratis Tutorials](#), [Abonnieren](#), [Trainings](#), [Trainer](#), [Lösungen](#), [Support](#), and [Apps](#). The main content area features a breadcrumb trail: [Alle Video-Trainings](#) » [IT](#) » [SQL Server](#). The title of the video is **SQL Server Integration Services – Grundlagen**, with a subtitle **Überblick und technischer Einstieg in den ETL-Prozess**. A video thumbnail shows a blue abstract graphic with the text **SQL Server Integration Services Grundlagen**. The description states: "Wenn Daten aus mehreren Datenquellen in eine Zieldatenbank, insbesondere in einem Datawarehouse zusammengeführt werden, nennt man diesen Prozess Extract-Transform-Load (ETL). Dafür gibt es im Microsoft SQL Server die Integration Services. Der Datenbank-Consultant und SQL-Entwickler Alexander Karl erläutert Ihnen in diesem Video-Training die Zusammenhänge und zeigt am Beispiel, wie Sie die SQL Server Integration Services (SSIS) erfolgreich einsetzen." The trainer is identified as **Alexander Karl**, the release date is **27.09.2013**, and the duration is **4 Std. 0 min**.



# publications



The screenshot shows a web browser window with the Amazon.de search results for the book 'DQS step-by-step mit SQL-Server: SQL-Server Data Quality Services' by Alexander Karl. The browser address bar shows the URL 'https://www.amazon.de/gp/'. The search results are displayed in a list format. The first result is the German edition, published on 15. Mai 2014. It is available as a Kindle Edition for EUR 0,00 (with Kindle Unlimited) and for EUR 9,99 (purchase). The book has a 5-star rating. The second result is the English edition, also published on 15. Mai 2014, available for EUR 0,00 (with Kindle Unlimited) and for EUR 9,99 (purchase). Both results include a 'Weitere Informationen' link and a 'Kindle-Shop' link.

← → a https://www.amazon.de/gp/ 🔍 🔒 ↻ a Suchergebnis auf Amazon.d... × 🏠 ★ ⚙️

Data Quality Services  
  
SQL Server

**DQS step-by-step mit SQL-Server: SQL-Server Data Quality Services** 15. Mai 2014  
von Alexander Karl


Kindle Edition  
**EUR 0,00** kindleunlimited  
Abonnenten lesen kostenlos. [Weitere Informationen](#)

**EUR 9,99** kaufen

★★★★★ 1  
kostenlos von Ihrem Kindle leihen. [Amazon Prime: Jetzt anmelden](#)  
[Kindle-Shop: Alle 2 Artikel ansehen](#)

Alexander Karl

---

Data Quality Services  
  
SQL Server

**DQS step-by-step with SQL-Server: SQL-Server Data Quality Services (English Edition)** 15. Mai 2014  
von Alexander Karl

Kindle Edition  
**EUR 0,00** kindleunlimited  
Abonnenten lesen kostenlos. [Weitere Informationen](#)

**EUR 9,99** kaufen

kostenlos von Ihrem Kindle leihen. [Amazon Prime: Jetzt anmelden](#)  
[Bücher \(Fremdsprachig\): Alle Artikel ansehen](#)

Alexander Karl

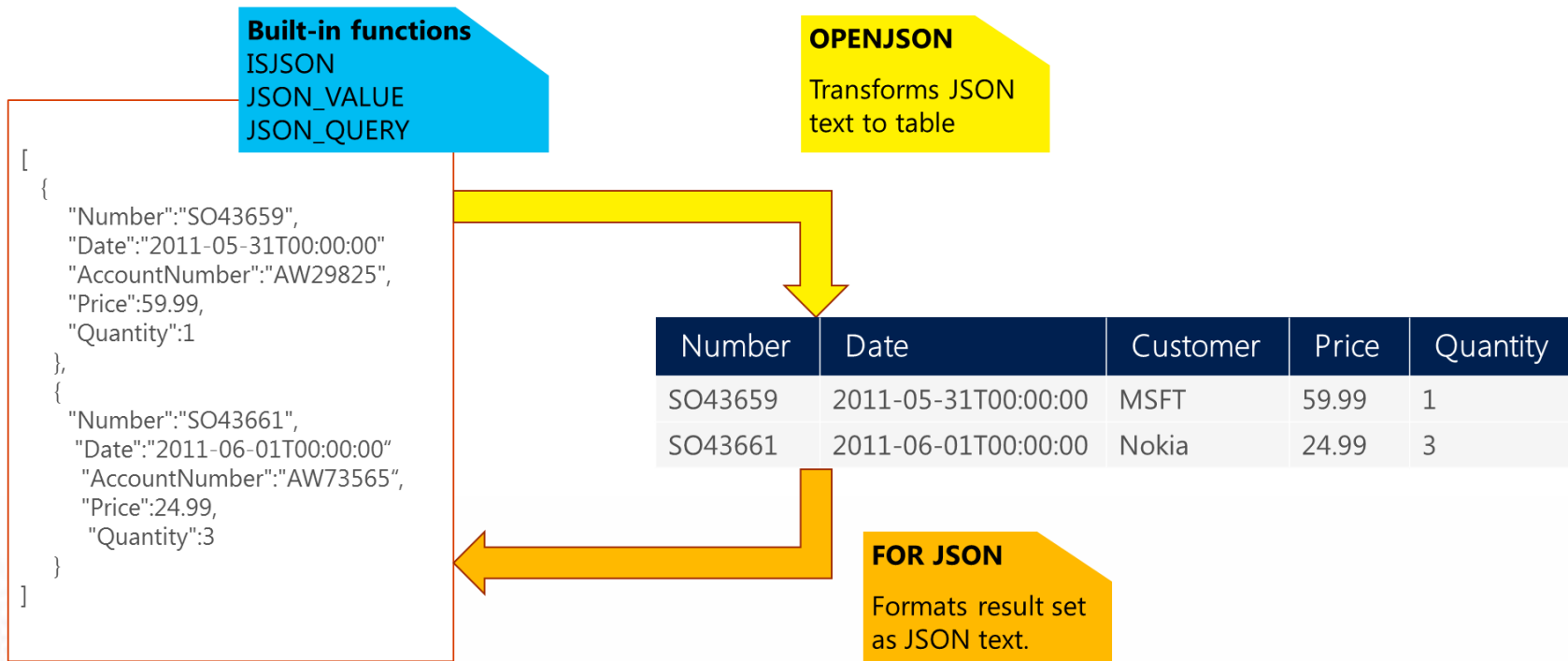
# Agenda

---

- JSON „new feature in SQL Server 2016“
- appropriate tools
- syntax For JSON / For XML
- syntax OpenJSON / OpenXML
- datatype ? JSON vs. XML
- indexes on datatype



# {JSON} „round trip“



<https://msdn.microsoft.com/en-us/library/dn921897.aspx>

# <xml /> tooling

The screenshot displays the SQL Server Enterprise Manager interface. On the left, the Object Explorer shows the database structure for 'dbo.CarManufacturer\_xml', including columns 'ID (PK, int, not null)', 'key (nvarchar(50), null)', and 'xmlValue (XML(,), null)'. The main window shows a SQL query in 'SQLQuery4.sql' that selects these columns from the table. Below the query, a 'Results' pane shows a table with three rows of data. A tooltip for the first row shows the XML content for 'ID=1', which is an XML document with a comment and a 'CompanyName' element.

```
1 SELECT [ID], [key], xmlValue
2 FROM dbo.CarManufacturer_xml
```

ID	key	xmlValue
1	Key01	<Customer CustomerID="1021"><!-- info 1021 custo...>
2	Key02	<Customer CustomerID="1022"><!-- info 1022 custo...>
3	Key03	<Customer CustomerID="1023"><!-- info 1023 custo...>

```
1 <Customer CustomerID="1021">
2   <!-- info 1021 customer since 2010-->
3   <CompanyName>Alfa Romeo</CompanyName>
4   <CustomerSince>2010</CustomerSince>
5 </Customer>
```

# {JSON}

## tooling

```
1 SELECT name          as 'name'  
2     , database_id    as 'database_id'  
3     , create_date    as 'create_date'  
4     , is_published   as 'replication.is_published'  
5     , is_subscribed  as 'replication.is_subscribed'  
6     , is_merge_published as 'replication.is_merge_published'  
7     , is_distributor as 'replication.is_distributor'  
8 FROM sys.databases  
9 FOR JSON PATH, ROOT('databases'), Include_NULL_Values;  
10
```

100 %

Results Messages

JSON\_F52E2B61-18A1-11d1-B105-00805F49916B

```
1 {"databases":[{"name":"master","database id":1,"create date":...
```

JSON\_F52E2B61-18...00805F49916B1.xml x demo01.sql

```
1 {"databases":[{"name":"master","database id":1,"create date":"2003-04-08T09:13:36.390"
```

Sign up for our free weekly Web Developer Newsletter.



ENTER THE DOLBY AUDIO CHALLENGE  
GRAND PRIZE \$10,000 USD  
CLICK HERE

Articles » Web Development » Client side scripting » General

- Tip/Trick
- Browse Code
- Stats
- Revisions (9)
- Alternatives
- Comments (11)
- Add your own alternative version

# View JSON in Internet Explorer

Coding 101, 21 May 2014 CPOOL  
★★★★★ 4.88 (17 votes)

Rate this: ★★★★★

A simple Registry change will enable IE to display JSON responses.

Need to view JSON responses in IE?

```
Windows Registry Editor Version 5.00;
; Tell IE 7,8,9,10,11 to open JSON documents in the browser on Windows XP and later.
; 25336920-03F9-11cf-8FD0-00AA00686F13 is the CLSID for the "Browse in place" .
;
[HKEY_CLASSES_ROOT\MIME\Database\Content Type\application/json]
"CLSID"="{25336920-03F9-11cf-8FD0-00AA00686F13}"
"Encoding"=hex:08,00,00,00
```

1. Open Notepad and paste the following:
2. Save document as *IE-Json.reg* and then run it.

- Tagged as
- Javascript
  - IE8
  - IE10
  - IE11
  - IE7
  - Browser

Internet Explorer browser window showing the URL `http://www.codeproject.com/Tips/216175/View-JSON-in-Internet-Explorer`. The page features a banner for Code Project and a promotion for the Dolby Audio Challenge with a grand prize of \$10,000 USD.

Internet Explorer browser window showing the URL `http://jsonview.com/example.json`. The page displays a JSON object:

```
{ "hey": "guy", "anumber": 243, "anobject": { "whoa": "nuts", "anarray": [1, 2, "three"], "more": "stuff" }, "awesome": true, "bogus": false, "meaning": null, "japanese": "明日がある。", "link": "http://jsonview.com", "notLink": "http://jsonview.com is great" }
```

Article content area with a 'Tagged as' section listing categories like Javascript, IE8, IE10, IE11, IE7, and Browser. A code block contains registry instructions for opening JSON files in IE:

```
Windows Registry Editor Version 5.00;  
; Tell IE 7,8,9,10,11 to open JSON documents in the browser on Windows XP and later.  
; 25336920-03F9-11cf-8FD0-00AA00686F13 is the CLSID for the "Browse in place" .  
;  
[HKEY_CLASSES_ROOT\MIME\Database\Content Type\application/json]  
"CLSID"="{25336920-03F9-11cf-8FD0-00AA00686F13}"  
"Encoding"=hex:08,00,00,00
```

Instructions:

1. Open Notepad and paste the following:
2. Save document as *IE-Json.reg* and then run it.



Firefox Add-ons Manager: about:addons

Erweiterungen

## JSONView 1.1.0

Von Ben Hollis

```
{
  hey: "guy",
  number: 242,
  - anobject: {
    whoa: "nuts",
    - anarray: [
      1,
      2,
      "thrchive"
    ],
    more: "stuff"
  },
  awesome: true,
  bogus: false,
  meaning: null,
  japanese: "明日があるさ。",
  link: http://jsonview.com,
  notLink: "http://jsonview.com is great"
}
```

JSON-Dokumente im Browser anzeigen.

Normalerweise bietet Firefox beim Aufrufen eines JSON-Dokuments (Content-Type "application/json") das Abspeichern der Datei an. Die JSONView-Erweiterung stellt JSON-Dokumente analog wie XML-Dokumente dar. Das Dokument verfügt über Formatierungen sowie Hervorhebungen und Felder und Objekte können erweitert und zusammengefasst werden. Enthält das JSON-Dokument Fehler, so zeigt JSONView nur den Quelltext an.

Um JSONView in Aktion zu sehen, rufen Sie nach der Installation <http://benhollis.net/software/jsonview/example.json> auf.

Automatische Updates:  Standard  Ein  Aus

Zuletzt aktualisiert: Sunday, February 21, 2016

Homepage: <http://jsonview.com/>

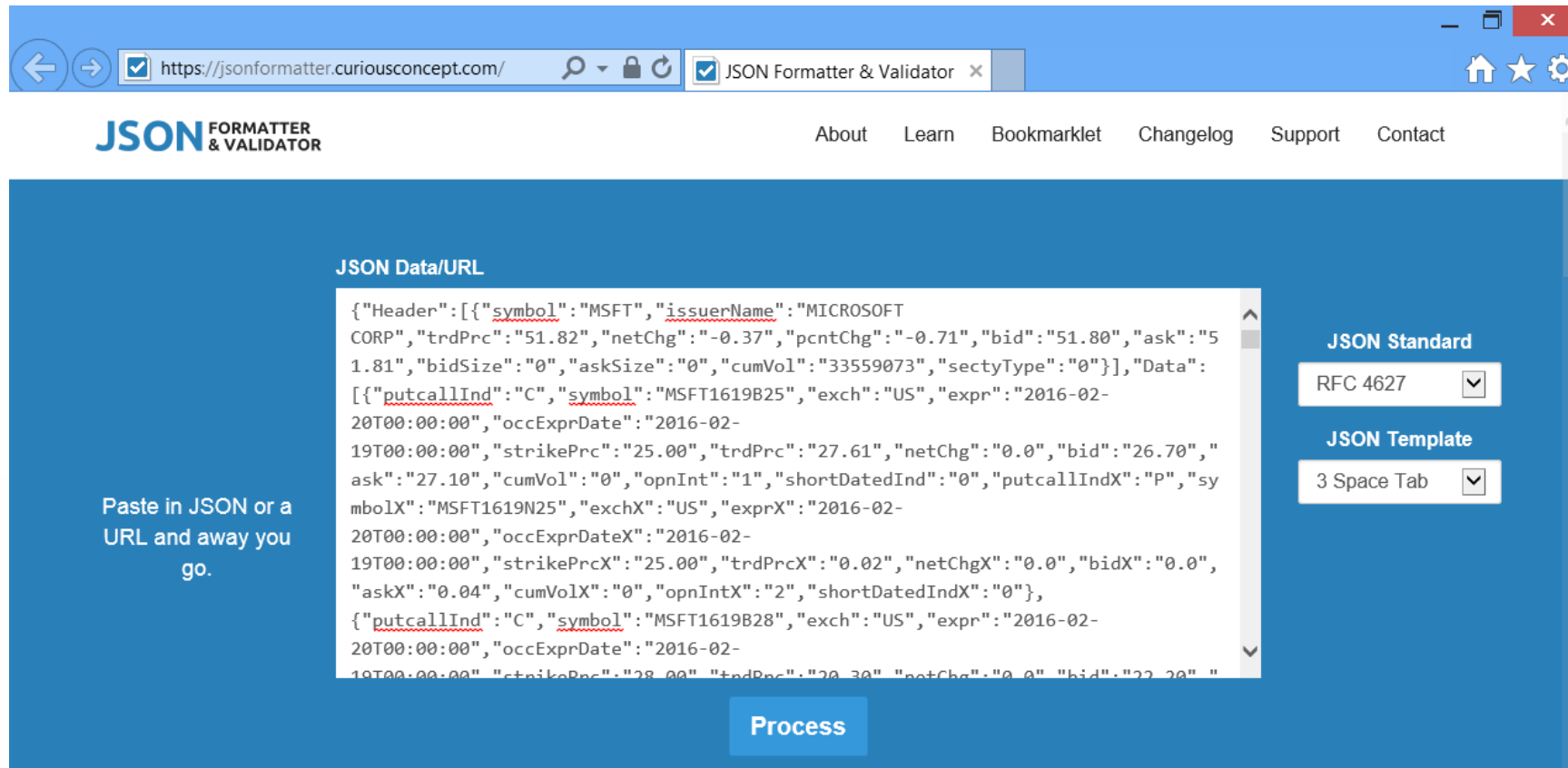
Bewertung: ★★★★★ 143 Bewertungen



The image shows a Firefox browser window with the Add-ons Manager open. The main window displays the 'about:addons' page. A secondary window is overlaid on top, showing a JSON viewer for the file 'http://jsonview.com/example.json'. The JSON content is as follows:

```
{
  hey: "guy",
  anumber: 243,
  anobject: {
    whoa: "nuts",
    anarray: [
      1,
      2,
      "thr<h1>ee"
    ],
    more: "stuff"
  },
  awesome: true,
  bogus: false,
  meaning: null,
  japanese: "明日がある。",
  link: http://jsonview.com,
  notLink: "http://jsonview.com is great"
}
```

# {JSON} Formatter & Validator

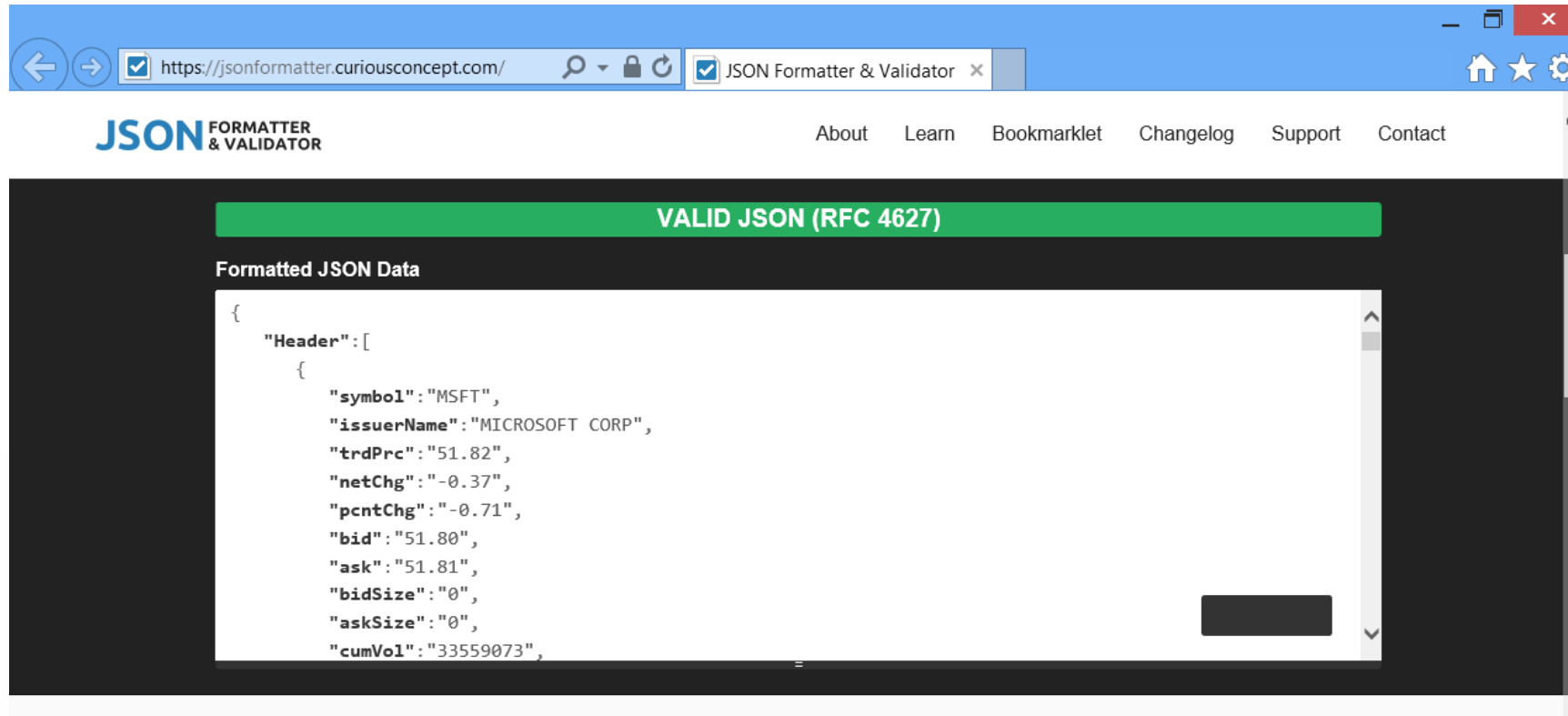


The screenshot shows a web browser window with the URL `https://jsonformatter.curiousconcept.com/`. The page title is "JSON FORMATTER & VALIDATOR". The navigation menu includes "About", "Learn", "Bookmarklet", "Changelog", "Support", and "Contact".

The main content area has a blue background. On the left, it says "Paste in JSON or a URL and away you go." In the center, there is a text area labeled "JSON Data/URL" containing a large JSON object. Below the text area is a blue button labeled "Process". On the right side, there are two dropdown menus: "JSON Standard" set to "RFC 4627" and "JSON Template" set to "3 Space Tab".

```
JSON Data/URL
{"Header":[{"symbol":"MSFT","issuerName":"MICROSOFT CORP","trdPrc":"51.82","netChg":"-0.37","pcntChg":"-0.71","bid":"51.80","ask":"51.81","bidSize":"0","askSize":"0","cumVol":"33559073","sectyType":"0"}],"Data":[{"putcallInd":"C","symbol":"MSFT1619B25","exch":"US","expr":"2016-02-20T00:00:00","occExprDate":"2016-02-19T00:00:00","strikePrc":"25.00","trdPrc":"27.61","netChg":"0.0","bid":"26.70","ask":"27.10","cumVol":"0","opnInt":"1","shortDatedInd":"0","putcallIndX":"P","symbolX":"MSFT1619N25","exchX":"US","exprX":"2016-02-20T00:00:00","occExprDateX":"2016-02-19T00:00:00","strikePrcX":"25.00","trdPrcX":"0.02","netChgX":"0.0","bidX":"0.0","askX":"0.04","cumVolX":"0","opnIntX":"2","shortDatedIndX":"0"}, {"putcallInd":"C","symbol":"MSFT1619B28","exch":"US","expr":"2016-02-20T00:00:00","occExprDate":"2016-02-19T00:00:00","strikePrc":"28.00","trdPrc":"20.30","netChg":"0.0","bid":"22.20"}]}
```

# {JSON} Formatter & Validator



The screenshot shows a web browser window with the URL <https://jsonformatter.curiousconcept.com/>. The page title is "JSON FORMATTER & VALIDATOR". The navigation menu includes "About", "Learn", "Bookmarklet", "Changelog", "Support", and "Contact". A green banner at the top of the main content area reads "VALID JSON (RFC 4627)". Below this, the text "Formatted JSON Data" is displayed above a code block containing the following JSON:

```
{
  "Header": [
    {
      "symbol": "MSFT",
      "issuerName": "MICROSOFT CORP",
      "trdPrc": "51.82",
      "netChg": "-0.37",
      "pctChg": "-0.71",
      "bid": "51.80",
      "ask": "51.81",
      "bidSize": "0",
      "askSize": "0",
      "cumVol": "33559073",
    }
  ]
}
```

---

# Demo



# {JSON} FOR JSON

## ■ AUTO Mode

```
SELECT name, database_id, create_date
FROM sys.databases
FOR JSON Auto
```

## ■ PATH Mode

```
SELECT name as 'name',
       , database_id as 'database_id',
       , create_date as 'create_date',
       , is_published as 'replication.is_published',
       , is_subscribed as 'replication.is_subscribed',
       , is_distributor as 'replication.is_distributor',
FROM sys.databases
FOR JSON PATH, ROOT('databases'), Include_NULL_Values;
```





# {JSON} FOR JSON

■ AU

```
file:///C:/ | /JSON_demo/autoMode.json
```

SELE [  
FROM  
FOR

```
file:///C:/ ... /JSON_demo/pathMode.json
```

■ PA

SELE

```
{  
  databases: [  
    {  
      name: "master",  
      database_id: 1,  
      create_date: "2003-04-08T09:13:36.390",  
      replication: {  
        is_published: false,  
        is_subscribed: false,  
        is_merge_published: false,  
        is_distributor: false  
      }  
    },  
    {  
      name: "tempdb",  
      database_id: 2,  
      create_date: "2016-02-24T11:40:54.827",  
      replication: {  
        is_published: false,  
        is_subscribed: false,  
        is_merge_published: false,  
        is_distributor: false  
      }  
    }  
  ]  
}
```

FROM ],  
FOR JS



# <xml /> FOR XML

---

## ■ AUTO Mode

```
SELECT name, database_id, create_date
FROM   sys.databases
FOR    XML Auto ;
```

## ■ PATH Mode

```
SELECT name                as 'name',
       , database_id       as 'database_id',
       , create_date       as 'create_date',
       , is_published      as 'replication/is_published',
       , is_subscribed     as 'replication/is_subscribed',
       , is_distributor    as 'replication/is_distributor',
FROM   sys.databases
FOR    XML PATH('db'), ROOT('databases');
```

# {JSON} OpenJSON

```
DECLARE @json NVARCHAR(MAX)
SET @json = '{
    "Name"      : "PASS Deutschland e.V.",
    "addInfo"   : null,
    "ID"        : 828,
    "Current"   : true,
    "Skills"    : ["SQL", "SSIS", "SSRS", 42, "MDX"],
    "Region"    : {"Country": "Germany", "Territory": "Hessen"}
}';
```

```
SELECT * -- [key], [value], type
FROM   OpenJSON(@json);
```

```
SELECT *
FROM   OpenJSON(@json, '$.Skills')
```

```
SELECT *
FROM   OpenJSON(@json, '$.Region')
```

# {JSON} OpenJSON

```
DECLARE @json NVARCHAR(MAX)
SET @json = '{
  "Name"
  "addInfo"
  "ID"
  "Current"
  "Skills"
  "Region"
}' ;
```

```
SELECT * -- [key]
FROM OpenJSON (@json, '$')
```

```
SELECT *
FROM OpenJSON (@json, '$.Skills')
```

```
SELECT *
FROM OpenJSON (@json, '$.Region')
```

	key	value	type
1	Name	PASS Deutschland e.V.	1
2	addInfo	NULL	0
3	ID	828	2
4	Current	true	3
5	Skills	["SQL","SSIS","SSRS",42,"MDX"]	4
6	Region	{"Country":"Germany","Territory":"Hessen"}	5

	key	value	type
1	0	SQL	1
2	1	SSIS	1
3	2	SSRS	1
4	3	42	2
5	4	MDX	1

	key	value	type
1	Country	Germany	1
2	Territory	Hessen	1

en" }

# {JSON}

## OpenJSON Data Type

```
1 DECLARE @json NVARCHAR(MAX)
2 SET @json = '{
3     "Name"      : "PASS Deutschland e.V.",
4     "addInfo"   : null,
5     "ID"        : 828,
6     "Current"   : true,
7     "Skills"    : ["SQL", "SSIS", "SSRS", 42, "MDX"],
8     "Region"    : {"Country": "Germany", "Territory": "Hessen"}
9 }';
10
```

100 % <

Results

Messages

	key	value	type	Data Type
1	Name	PASS Deutschland e.V.	1	string
2	addInfo	NULL	0	null
3	ID	828	2	int
4	Current	true	3	true/false
5	Skills	["SQL", "SSIS", "SSRS", 42, "MDX"]	4	array
6	Region	{"Country": "Germany", "Territory": "Hessen"}	5	object

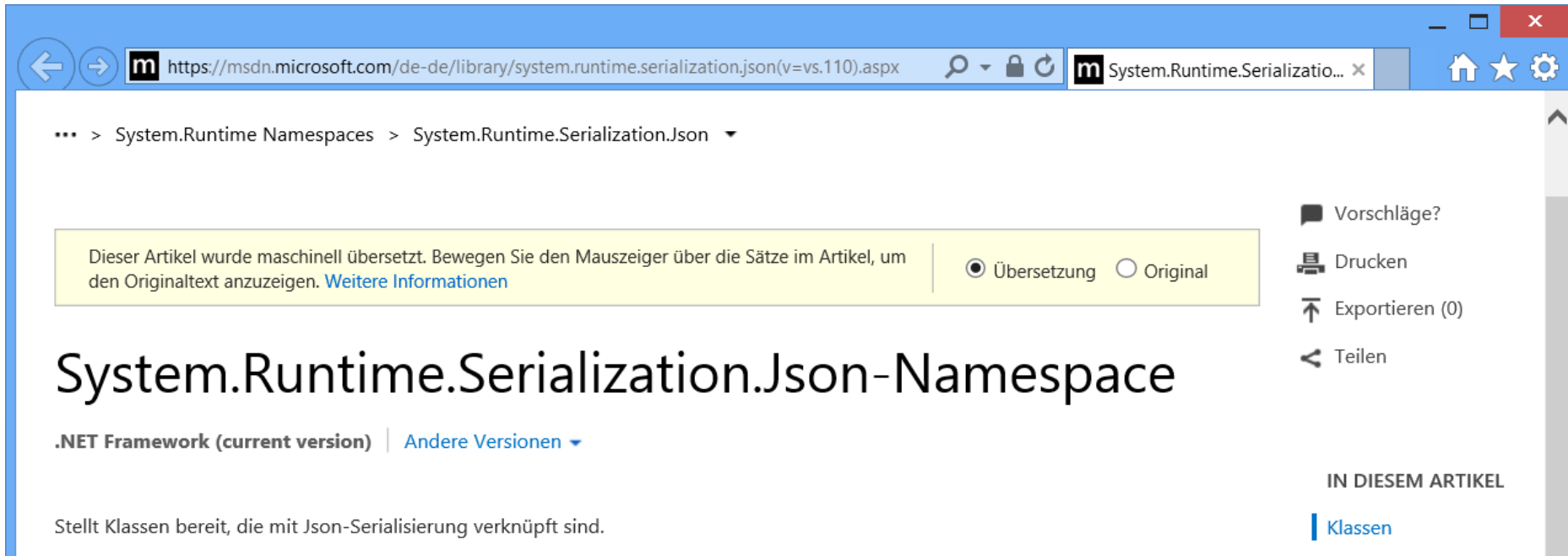
# {JSON} Storing JSON

- NVARCHAR (MAX)
- Index  
„abgeleitete Spalte“  
mit `JSON_VALUE ( @JsonCol, '$. ... . ... ' )`

```
>> CREATE INDEX ...
```



# {JSON} JSON Namespaces



The screenshot shows a web browser window displaying the Microsoft Docs page for the `System.Runtime.Serialization.Json` namespace. The browser's address bar shows the URL `https://msdn.microsoft.com/de-de/library/system.runtime.serialization.json(v=vs.110).aspx`. The page content includes a breadcrumb trail: `System.Runtime Namespaces > System.Runtime.Serialization.Json`. A yellow notification box states: "Dieser Artikel wurde maschinell übersetzt. Bewegen Sie den Mauszeiger über die Sätze im Artikel, um den Originaltext anzuzeigen. [Weitere Informationen](#)". Below this, there are radio buttons for "Übersetzung" (selected) and "Original". The main heading is "System.Runtime.Serialization.Json-Namespace". Underneath, it says ".NET Framework (current version) | [Andere Versionen](#)". The introductory text reads: "Stellt Klassen bereit, die mit Json-Serialisierung verknüpft sind." On the right side, there is a sidebar with navigation options: "Vorschläge?", "Drucken", "Exportieren (0)", and "Teilen". Below these, under the heading "IN DIESEM ARTIKEL", there is a link for "Klassen".



# JSON schema validation

JSON Schema Generator

jsonschema.net/#/

Home About Contact Resources Previous Version

### JSON

URL:

JSON: 

```
{
  "Header": [
    {
      "symbol": "MSFT",
      "issuerName": "MICROSOFT CORP",
      "trdPrc": "51.82",
      "netChg": "-0.37",
      "pcntChg": "-0.71",
      "bid": "51.80",
      "ask": "51.81",
      "bidSize": "0",
      "askSize": "0",
      "cumVol": "33559073",
    }
  ]
}
```

Well done! You provided valid JSON.

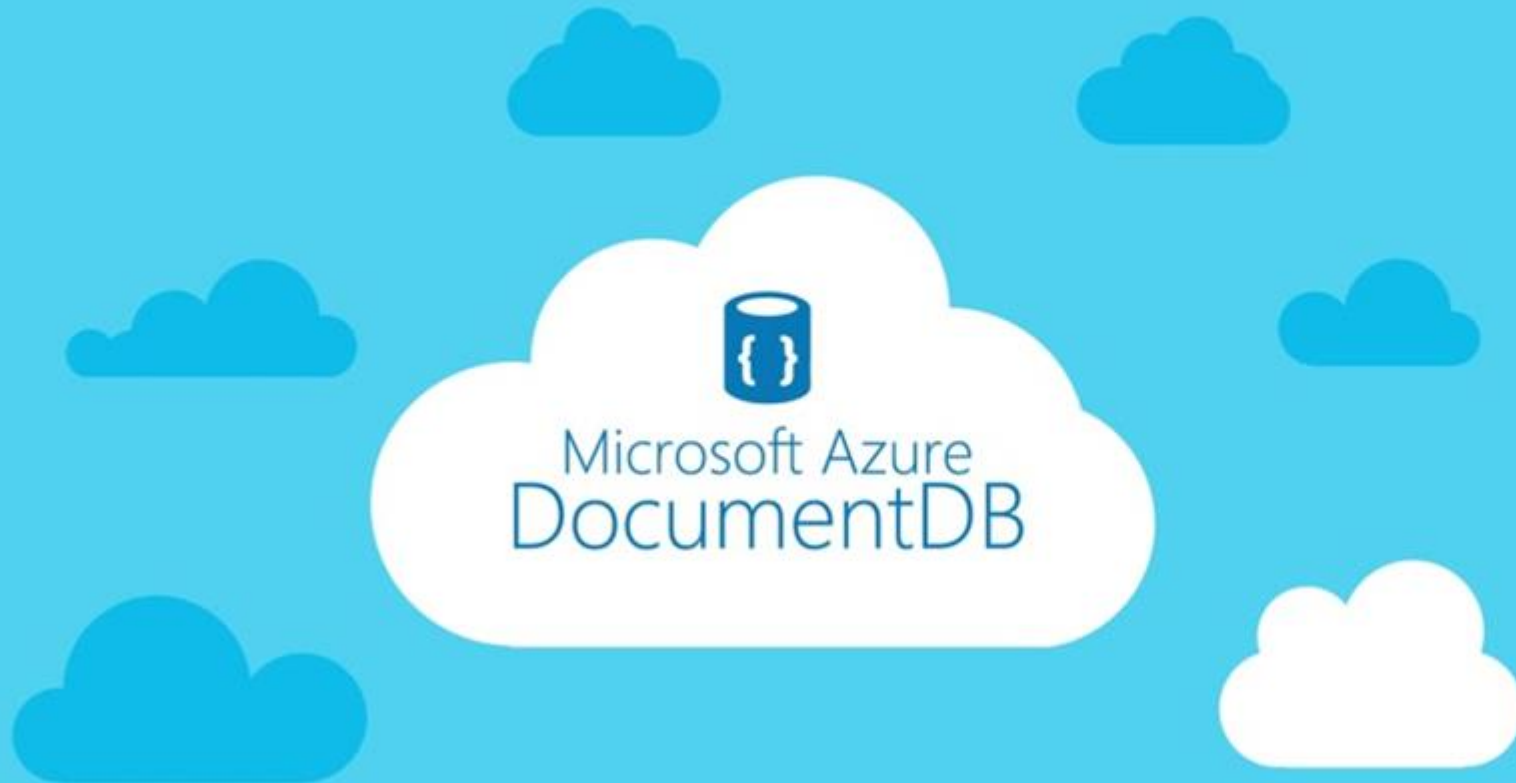
Metadata  Include metadata keywords

### Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "",
  "type": "object",
  "properties": {
    "Header": {
      "id": "/Header",
      "type": "array",
      "items": {
        "id": "/Header/0",
        "type": "object",
        "properties": {
          "symbol": {
            "id": "/Header/0/symbol",
            "type": "string"
          },
          "issuerName": {
            "id": "/Header/0/issuerName",
            "type": "string"
          },
          "trdPrc": {
            "id": "/Header/0/trdPrc",
            "type": "string"
          }
        }
      }
    }
  }
}
```



.. and Microsoft ❤️ to store JSON data here



# {JSON} JSON functions

---

- **ISJSON()**  
return 1 if JSON snippet is valid
- **JSON\_VALUE()**  
extract a scalar value from JSON snippet
- **JSON\_QUERY()**  
extract an object or an array from JSON snippet

# <xml /> Select & Ins/Upd/Del

---

- T-SQL conformer Ins/Upd/Del
- T-SQL XQuery
  - <column>.query ( return xml )
  - <column>.value ( return sql-Type )
  - <column>.exist ( return bit )
  - <column>.nodes



# <xml /> Select & Ins/Upd/Del

---

- T-SQL Xquery Navigation im <xml>
  - child
  - descendant
  - parent
  - Attribut
  - self
  - descendant-or-self
  
- .node()
- .text()





# <xml /> CRUD Samples

```
SQLQuery1.sql x
1  ---- 1st Query
2  SELECT [key], xmlValue.query('*') as Complete_Sequence
3  FROM   dbo.CarManufacturer_xml
4
5  ---- 2nd Query
6  SELECT [key], xmlValue.query('data(*)') as Complete_Data
7  FROM   dbo.CarManufacturer_xml
8
9  ---- 3rd Query /node() & /comment()
10 SELECT [key]
11        , xmlValue.query('/Customer/CompanyName/node()') as CompanyName
12        , xmlValue.query('/Customer/comment()')
13 FROM   dbo.CarManufacturer_xml
14
```

# <xml /> Indexing

---

- PRIMARY XML INDEX IX\_primaryXml
- CREATE XML INDEX IX\_name  
ON sch.table ( xmlCol )  
USING XML INDEX IX\_primaryXml
  - FOR PATH;
  - FOR VALUE;
  - FOR PROPERTY;

# <xml /> Indexing

```
xml_Indexes.sql x
1 CREATE PRIMARY XML INDEX IX_primaryXml
2 ON dbo.eConsumption_xml ( xmlValue );
3 GO
4
5
6 CREATE XML INDEX IX_Xml_PATH
7 ON dbo.eConsumption_xml ( xmlValue )
8 USING XML INDEX IX_primaryXml
9 FOR PATH;
10
11 CREATE XML INDEX IX_Xml_PROPERTY
12 ON dbo.eConsumption_xml ( xmlValue )
13 USING XML INDEX IX_primaryXml
14 FOR PROPERTY;
15
16 CREATE XML INDEX IX_Xml_VALUE
17 ON dbo.eConsumption_xml ( xmlValue )
18 USING XML INDEX IX_primaryXml
19 FOR VALUE;
20 GO
```

# <xml /> Schema Validation

---

- XML SCHEMA COLLECTION schemaName
- (table) xmlColumn  
xml ( DOCUMENT schemaName )



# <xml /> Schema Validation

```
xml_Schema.sql x
1 CREATE XML SCHEMA COLLECTION dbo.bookSchemaCollection
2 AS
3 N'<?xml version="1.0"?>
4 <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
5   targetNamespace="http://www.test-fabrik.de/BooksSchema"
6   xmlns="http://www.test-fabrik.de/BooksSchema"
7   elementFormDefault="qualified">
8   <xs:element name="Book">
9     <xs:complexType>
10      <xs:attribute name="Title" type="xs:string"/>
11      <xs:attribute name="Price" type="xs:decimal"/>
12    </xs:complexType>
13  </xs:element>
14 </xs:schema>';
15 GO
16
17 CREATE TABLE dbo.Books_xml(
18   ID int Identity(1,1)
19   , [key] nvarchar(50)
20   , xmlRaw xml
21   , xmlValue xml(DOCUMENT bookSchemaCollection)
22 );
23 GO
24
```

# <xml /> Schema Validation

```
xml_Schema.sql x
1  INSERT dbo.Books_xml
2      ( [key] , xmlValue )
3  VALUES ( 'key02', '<Book xmlns="http://www.test-fabrik.de/BooksSchema"
4              Title="the booksample2"
5              Price="49.90" />' )
6
7  -----
8  -- !! decimal ,
9  INSERT dbo.Books_xml
10     ( [key] , xmlValue )
11  VALUES ( 'key03', '<Book xmlns="http://www.test-fabrik.de/BooksSchema"
12              Title="! v ! Book"
13              Price="49,90" />' )
14
15  --Msg 6926, Level 16, State 1, Line 40
16  --XML Validation: Invalid simple type value: '49,90'. Location: /*:Book[1]/@*:Price
```



# <xml /> queries & transformation

---

- XML „*FLWOR*“  
for let where order return
- Aggregat functions
  - ✓ {count(\$j)}
  - ✓ {sum(\$j)}
  - ✓ {avg(\$j)}



# <xml /> queries & transformation

```
xml_FLWOR.sql x
1  -- sample FLWOR
2  SELECT ID, [key], xmlValue
3      , xmlValue.value('(CustomerData/Customer/CustID)[1]', 'nvarchar(40)' ) as 'nvarchar_CustID'
4      , xmlValue.query(' for $i in CustomerData/Customer/CustID
5                          let $j := CustomerData/Customer/Consumption/consumptionValue
6                          where avg($j) >= 5
7                          return
8                              <result>
9                                  {data($j)}
10                             </result>
11                             ') as 'data'
12      , xmlValue.query(' for $i in CustomerData/Customer/CustID
13                          let $j := CustomerData/Customer/Consumption/consumptionValue
14                          where avg($j) >= 5
15                          return
16                              <result>
17                                  <count> {count($j)} </count>
18                                  <sum> {sum($j)} </sum>
19                                  <avg> {avg($j)} </avg>
20                              </result>
21                              ') as 'summary'
22 FROM    dbo.eConsumtion_xml;
```

---

???



# Please review the event and sessions

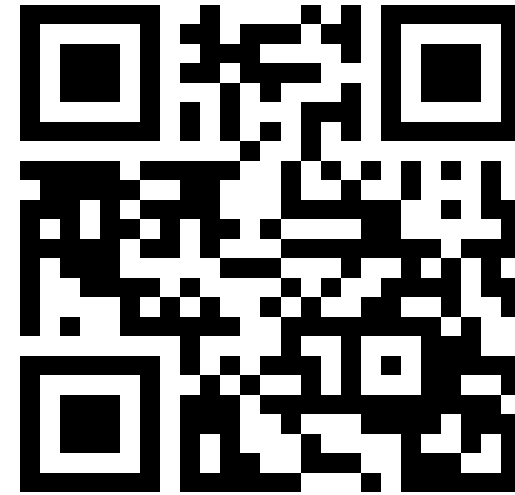
---

- EVENT



<http://speakerscore.com/ZGVX>

- SESSION



<http://speakerscore.com/FQ1W>