

XML a JSON, Chatgpt

1. XML

- Prvky, atribúty, **dobre** štruktúrovaný XML dokument
- SQL SERVER a XML

2. Úvod do XPATH a XQUERY

3. XSD (XML Schema Definition)

- Prvky, Atribúty, Typy a Indikátory **správny** dokument

4. XML vs JSON

5. Chatgpt.com

1. XML

a) Definícia

b) Značkovacie jazyky: HTML, ...

c) Prvky, atribúty, dobre štruktúrovaný a správny dokument

d) SQL SERVER a XML

a) Definícia

XML ako metajazyk slúži na popis štruktúry, výmenu a ukladanie dát.

XML (eXtensible Markup Language - rozšíriteľný značkovací jazyk) - W3C (World Wide Web Consortium)

- ako jazyk dovoľuje **popísať** (aj hierarchickú) **štruktúru** dokumentu, vytvárať štruktúrované dokumenty
- ako formát súboru, obsahujúci dáta, umožňuje **výmenu údajov** medzi aplikáciami a ich uloženie
- umožňuje **oddelenie obsahu** od **zobrazenia dát**

b) Značkovacie jazyky: HTML, ...

XML je pokračovanie jazyka SGML, je jeho jednoduchšia verzia.

Ale má prísnejšiu syntax ako HTML:

```
<!-- haha.html -->
<html>
<body>
  Blava
  <table style="width: 24%;">
    <tr>
      <td>1</td>
      <td>2</td>
    </tr>
    <tr>
      <td>3</td>
      <td style="background-color: blue">4</td>
    </tr>
  </table>
</body>
</html>
```

Blava	
1	2
3	4

Ďalšie jazyky pre podporu práce s XML:

XPath, XQuery + XML DML, XSLT; XSD (XML Schema Definition)

c) Prvky, atribúty a dobre štruktúrovaný dokument

XML dokument sa skladá

- z malého počtu preddefinovaných, **štandardných** prvkov a
- z užívateľom pomenovaných **vlastných** prvkov.

Prvok/element je kombinácia užívateľom pomenovaných **značiek (tagov <...>)** a **dát**. Každý prvok má svoj typ/názov, uvedený v značke. Značka slúži na označenie, popis prvku. Prvok môže obsahovať aj **atribúty**.

XML dokument je **dobre štruktúrovaný/vytvorený** ak dodržiava isté pravidlá - spĺňa špecifikáciu XML: http://www.w3schools.com/xml/xml_validator.asp

- dokument XML musí obsahovať koreňový prvok, v ktorom sú všetky ostatné prvky vložené
- každý prvok musí mať **začiatkovú** aj **koncovú** značku

```
<prvok1 atr1="1"> haha </prvok1>
```

 - existuje skrátený zápis pre prvky, neobsahujúce data `<prvok1 />`
 - názvy prvkov sú citlivé na malé a veľké písmená
 - hodnoty **atribútov** píšeme do úvodzoviek/ľavých apostrofov (a do začiatkových tagov)
- značky prvkov musia byť správne **vnorené** (vnútorný prvok má byť úplne obsiahnutý vo vonkajšom)

Lala.xml:

```
<?xml version="1.0" ?>
<!-- 1.priklad s komentarom -->
<senat>
  <stud rocnik="1"> Elizabet </stud>
  <stud />
  <![CDATA[<prv1> 1 &lt; 2 </prv1>]]>
</senat>
```

<	<
>	>
&	&
'	'
"	"

Špeciálne znaky.

MS Visual Studio (.NET)

Dáta môžeme vložiť/zapísať do XML dokumentu ako **prvok** alebo **atribút**.

1	2	3	
4	5	6	70

Stĺpce ako atribúty:

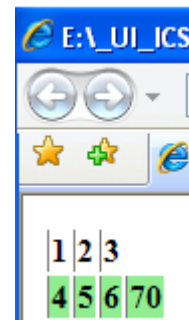
```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<Matrix>
  <row Col1="1" Col2="2" Col3="3" />
  <row Col1="4" Col2="5" Col3="6" Col4="70" />
</Matrix>
```

Stĺpce ako prvky + format MatrixAttributes.xml + MatrixElements.xsl:

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<?xml-stylesheet type='text/xsl' href='MatrixElements.xsl' ?>
<Matrix>
  <row>
    <col>1</col>
    <col>2</col>
    <col>3</col>
  </row>
  <row>
    <col>4</col>
    <col>5</col>
    <col>6</col>
    <col>70</col>
  </row>
</Matrix>
```

Predchádzajúci xml dokument sa odvoláva na nasledujúci xsl dokument MatrixElements.xsl. Prvok xsl:template popisuje množinu prvkov, ako ich spracovať a zobrazovať (beží na <https://www.w3schools.com/xml/tryxslt.asp?xmlfile=catalog&xsltfile=catalog> a **IE**)

```
<?xml version="1.0" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html" />
  <xsl:template match="/">
    <HTML>
      <BODY>
        <table>
          <xsl:for-each select="//Matrix/row">
            <tr>
              <xsl:if test="position() mod 2 = 0">
                <xsl:attribute name="bgcolor">lightgreen</xsl:attribute>
              </xsl:if>
              <xsl:for-each select="col">
                <td style="text-align:right;font-weight:bold;border-left:solid 0.5pt gray">
                  <xsl:value-of select="." />
                </td>
              </xsl:for-each>
            </tr>
          </xsl:for-each>
        </table>
      </BODY>
    </HTML>
  </xsl:template>
</xsl:stylesheet>
```



Menné priestory MPy

MP/Name Space v XML je unikátna kolekcia disjunktných mien definovaných pomocou URI referencie a atribútu xmlns. Menné priestory môžu byť definované implicitne alebo explicitne.

```
<matice xmlns = "http://torok.csaba.com/csaba">
```

d) SQL SERVER a XML

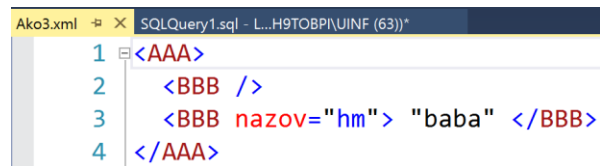
Stĺpce tabuľky, premenné T-sql a parametre uložených procedúr a funkcií môžu byť typu XML. V MS SQL Server XML môžeme

1. písať/vytvoriť
2. načítať
3. premapovať
4. generovať

1. manuálne písať/vytvoriť premennú či stĺpec

a) Premennú (píšeme do SQL query):

```
DECLARE @var XML
SET @var = '
    <AAA>
        <BBB />
        <BBB nazov="hm"> "baba" </BBB>
    </AAA> '
SELECT @var Ako
```



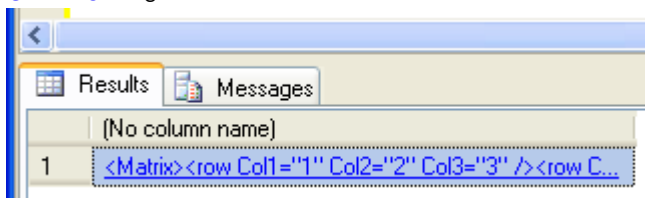
b) Stĺpec:

```
IF OBJECT_ID('T') IS NOT NULL DROP TABLE T
GO
CREATE TABLE T (k int, xx xml default
N'<kor><a /><prv2>222</prv2></kor>')
GO
INSERT T(k) VALUES (1), (2);
INSERT T(k,xx) VALUES (3, N'<kor><a /><prv2>333</prv2></kor>');
SELECT * FROM T
```

k	xx
1	<kor><a /><prv2>222</prv2></kor>
2	<kor><a /><prv2>222</prv2></kor>
2	<kor><a /><prv2>333</prv2></kor>

2. načítat' - FROM openrowset(bulk 'cesta', single_clob)

```
DECLARE @xmlPr xml -- s premennou neide
SET @xmlPr = (
    select * from openrowset
        (bulk N'C:\MatrixAttributes.xml', single_clob)
    AS a )
SELECT @xmlPr
```

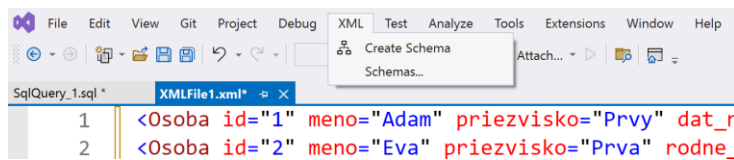


3. premapovať - FROM Osoba FOR XML AUTO

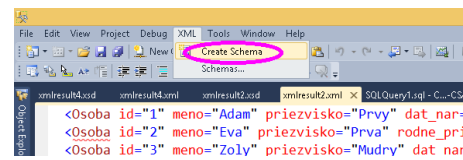
```
USE OsobaVztah
DECLARE @xmlDoc xml
SET @xmlDoc = (SELECT * FROM Osoba FOR XML AUTO) --a)
--SET @xmlDoc = (SELECT * FROM Osoba FOR XML AUTO, XMLSCHEMA) --b)
--SET @xmlDoc = (SELECT * FROM Osoba FOR XML AUTO, ELEMENTS XSINIL) -- c) d)
--SET @xmlDoc = (SELECT * FROM Osoba FOR XML AUTO, ELEMENTS XSINIL, XMLSCHEMA)
SELECT @xmlDoc
```

Poznamenáme, že v prípade:

- a) `SET @xmlDoc = (SELECT * FROM Osoba FOR XML AUTO) --a)`
`--SET @xmlDoc = (SELECT * FROM Osoba FOR XML AUTO, XMLSCHEMA) --b)`
- riadky neobsahujú rovnaký počet atribútov - z výsledku chýbajú NULL hodnoty
 - vygenerovaná (v MS VS, SQLS starý) XSD schéma sa opiera o atribúty.



MS Visual Studio



SQL Server – starý

```

- <xs:element name="Osoba">
-   <xs:complexType>
-     <xs:attribute name="id" type="xs:unsignedByte" use="required" />
-     <xs:attribute name="meno" type="xs:string" use="required" />
-     <xs:attribute name="priezvisko" type="xs:string" use="required" />
-     <xs:attribute name="dat_nar" type="xs:dateTime" use="required" />
-     <xs:attribute name="dat_smrti" type="xs:dateTime" use="optional" />
-     <xs:attribute name="pohlavie" type="xs:string" use="required" />
-     <xs:attribute name="vyska" type="xs:decimal" use="required" />
-     <xs:attribute name="vaha" type="xs:decimal" use="optional" />
-     <xs:attribute name="rodne_priezvisko" type="xs:string" use="optional" />
-     <xs:attribute name="otec" type="xs:unsignedByte" use="optional" />
-     <xs:attribute name="matka" type="xs:unsignedByte" use="optional" />
-   </xs:complexType>
- </xs:element>
  
```

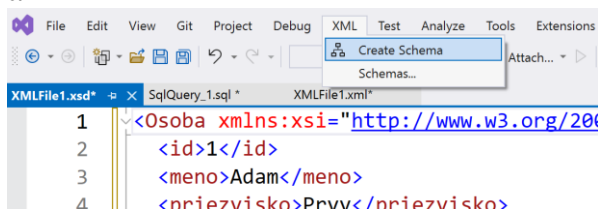
b) XMLSCHEMA

```

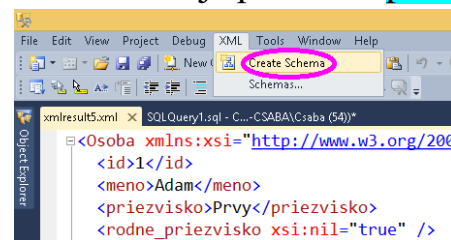
<Osoba xmlns="urn:schemas-microsoft-com:sql:SqlRowSet1" id="1" meno="Adam" priezvisko="Prvy"
dat_nar="1918-05-11" dat_smrti="1968-10-01" pohlavie="m" vyska="180.0" vaha="80.0" />
...
<xsd:attribute name="meno" use="required">
  <xsd:simpleType>
    <xsd:restriction base="sqltypes:varchar" sqltypes:localeId="1033"
      sqltypes:sqlCompareOptions="IgnoreCase IgnoreKanaType IgnoreWidth" sqltypes:sqlSortId="52">
      <xsd:maxLength value="10" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
...
  
```

- c) V prípade: `SET @xmlDoc = (SELECT * FROM Osoba FOR XML AUTO , ELEMENTS XSINIL) -- c) d)`
`--SET @xmlDoc = (SELECT * FROM Osoba FOR XML AUTO , ELEMENTS XSINIL, XMLSCHEMA)`

- výsledok obsahuje NULL hodnoty a XSD schéma sa konštruje pomocou prvkov a nie atribútov.



MS Visual Studio



SQL Server – starý

```

<xs:element name="Osoba">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="id" type="xs:unsignedByte" />
      <xs:element name="meno" type="xs:string" />
      <xs:element name="priezvisko" type="xs:string" />
      <xs:element name="rodne_priezvisko" nillable="true" type="xs:string" />
      <xs:element name="dat_nar" type="xs:dateTime" />
      <xs:element name="dat_smrti" nillable="true" type="xs:string" />
      <xs:element name="pohlavie" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
  
```

```

        <xs:element name="vyska" type="xs:decimal" />
        <xs:element name="vaha" nillable="true" type="xs:string" />
        <xs:element name="otec" nillable="true" type="xs:string" />
        <xs:element name="matka" nillable="true" type="xs:string" />
    </xs:sequence>
</xs:complexType>
</xs:element>

```

d)

```

6      <xsd:element name="id" type="sqltypes:int" nillable="1" />
7      <xsd:element name="meno" nillable="1">
8          <xsd:simpleType>
9              <xsd:restriction base="sqltypes:varchar" sqltypes:localeId="1051" sqlt
10                 <xsd:maxLength value="10" />
11                 </xsd:restriction>
12             </xsd:simpleType>
13         </xsd:element>
14         <xsd:element name="nriezvisko" nillable="1">

```

4. generovat' (pmocou query a for)

```

declare @x xml
set @x=''
SELECT @x.query('for $a in (1, 2, 3)
                return $a*10') -- = 10 20 30

```

```

-- declare @x xml
set @x=''
SELECT @x.query('for $a in (1, 2, 3)
                return <a10> {$a*10} </a10>
                ')

```

```

--<a10>10</a10>
--<a10>20</a10>
--<a10>30</a10>

```

```

-- Dalsie ukazky:
declare @x xml
set @x='<ROOT><a>789</a></ROOT>'
SELECT @x -- <ROOT> <a>789</a> </ROOT>

SELECT @x.query('
for $a in (xs:string( "haha"), xs:double( "-1.23" ), data(/ROOT/a ))
return $a') as cc -- haha -1.23 789

```

2. Úvod do XPATH a XQUERY

[http://msdn.microsoft.com/en-us/library/ms25471\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/ms25471(v=vs.110).aspx)

XML sa líši od relačných dát, preto potrebuje vlastný dopytovací jazyk. XQUERY spolu s XPATH //...[]... umožňujú dopytovanie XML dokumentov.

Príklady:

```

declare @x xml
set @x='<root>
        <Osoba Meno="B" />
        <Osoba />
        <Osoba Meno="A" />
    </root>'

--select @x
--select @x.query('<Osoba MENO="A"/>') --<Osoba Meno="A" />
select @x.query(
    'for $osoba in //Osoba
      order by $osoba/@Meno
      return $osoba')
1 <Osoba />
2 <Osoba Meno="A" />
3 <Osoba Meno="B" />

```

Alebo:

```
... //Osoba[3] ...  
... //Osoba[last()] ...  
... //Osoba[@Meno="B"] ...  
... //Osoba[attribute::Meno="B"] ...
```

```
select T.jaj.query(  
  'for $person in //Osoba[attribute::Meno]  
    order by $person/@Meno  
    return $person')  
FROM @x.nodes('.') AS T(jaj)
```

1 <Osoba Meno="A" />
2 <Osoba Meno="B" />

3. XSD (XML Schema Definition)

[http://msdn.microsoft.com/en-us/library/ms260351\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/ms260351(v=vs.110).aspx)

<http://msdn.microsoft.com/en-us/library/aa46549.aspx>

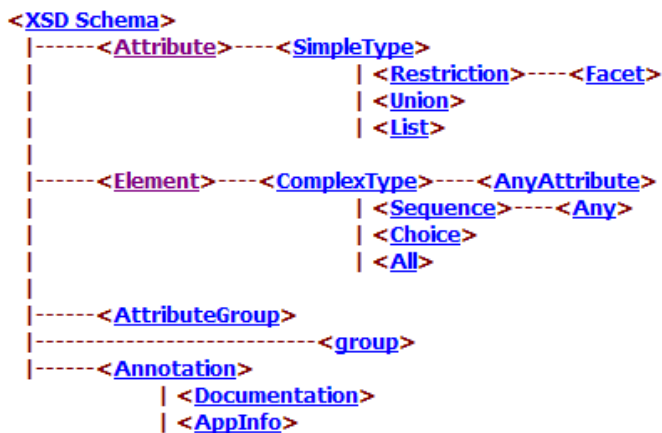
XML Schema je popis prípustných značiek, atribútov, ich štruktúry a typu.

Ak XML dokument obsahuje iba značky danej schémy, hovoríme, že je **voči nej správny/validný**.

XML Schema, ktorá má viac možností ako DTD (Document Type Definition), nevyžaduje špecifickú

syntax a umožňuje špecifikovať

- viachodnotové dátové typy
- počet výskytu prvkov
- maximálnu / minimálnu hodnotu prvku
- postupnosť a množinu prvkov
- OO prvky



XSD Schema Elements:

- <XSD_Schema>
- <Attribute>
- <Element>
- <ComplexType>
- <SimpleType>
- <Sequence>
- <Choice>
- <All>
- <Restriction>
- <Union>
- <List>
- <Facet>
- <Any>
- <AnyAttribute>
- <group>
- <AttributeGroup>
- <Annotation>
- <Documentation>
- <AppInfo>

Ilustračné príklady na **choice**, **sequence**, **enumeration** a **list**

```
<xsd:element name="obcan">  
  <xsd:complexType>  
    <xsd:choice>  
      <xsd:element name="domorodec" type="domorodec"/>  
      <xsd:element name="cudzinec" type="cudzinec"/>  
    </xsd:choice>  
  </xsd:complexType>  
</xsd:element>  
  
<xsd:element name="adresa">  
  <xsd:complexType>  
    <xsd:sequence>
```

```

        <xsd:element name="ulica" type="xsd:string"/>
        <xsd:element name="cislo" type="xsd:string"/>
        <xsd:element name="mesto" type="xsd:string"/>
        <xsd:element name="sm_cis" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
</xsd:element>
-- https://www.w3.org/TR/xmlschema-0/#ListDt
<xsd:simpleType name="USState">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="AK"/>
        <xsd:enumeration value="AL"/>
        <xsd:enumeration value="AR"/>
        <!-- and so on ... -->
    </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="USStateList">
    <xsd:list itemType="USState"/>
</xsd:simpleType>

<xsd:simpleType name="SixUSStates">
    <xsd:restriction base="USStateList">
        <xsd:length value="6"/>
    </xsd:restriction>
</xsd:simpleType>

<sixStates>PA NY CA NY LA AK</sixStates>

```

Príklad

Vytvorte schému pre špeciálnu maticu s tromi stĺpcami, kde do prvého stĺpca môžeme uložiť iba hodnoty 1, 4, 7, 10.

V MS Visual Studio otvoríme nový Xml súbor do ktorého napíšeme Xml prvok `<Matrix>`

<pre> <Matrix> <row Col1="1" Col2="2" Col3="3" /> <row Col1="4" Col2="5" Col3="6" /> <row Col1="7" Col2="8" Col3="9" /> <row Col1="10" Col2="11" Col3="12"/> </Matrix> </pre>	<pre> IF OBJECT_ID('T') IS NOT NULL DROP TABLE T GO CREATE TABLE T (id int, xx xml default N' <Matrix> <row Col1="1" Col2="2" Col3="3" /> <row Col1="4" Col2="5" Col3="6" /> <row Col1="7" Col2="8" Col3="9" /> <row Col1="10" Col2="11" Col3="null" /> </Matrix>) GO INSERT T(id) VALUES (1) SELECT * FROM T FOR XML AUTO, XMLSCHEMA SELECT * FROM T FOR XML AUTO, ELEMENTS XSINIL, XMLSCHEMA </pre>
---	--

na základe ktorého vygenerujeme schému (MS VS ponuka XML—Create Schema), do ktorej dopíšeme jednoduchý typ `xs:simpleType` s názvom `int1_4_7_10` a predefinujeme typ atribútu `Col1` Xml prvku `Matrix` na `int1_4_7_10`:

```

<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

```

```

  <xs:simpleType name="int1_4_7_10">
    <xs:restriction base="xs:nonNegativeInteger">
      <!-- <xs:maxInclusive value="100"></xs:maxInclusive -->
      <xs:enumeration value="1"></xs:enumeration>
      <xs:enumeration value="4"/>
      <xs:enumeration value="7"/>
      <xs:enumeration value="10"/>
    </xs:restriction>
  </xs:simpleType>

```



```

<xs:element name="Matrix">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" name="row">
        <xs:complexType>
          <xs:attribute name="Col1" type="int1_4_7_10" use="required" />
          <xs:attribute name="Col2" type="xs:int" use="required" />
          <xs:attribute name="Col3" type="xs:int" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Xml prvok <Matrix> a vygenerovanú a modifikovanú schému prekopírujeme do vhodných miest nasledujúcej kódovej šablóny:

```

----- Zoznam schema collections:
--SELECT * FROM sys.xml_schema_collections
----- Zoznam namespaces:
--SELECT name FROM sys.xml_schema_namespaces

USE tempdb
GO

DROP TABLE if exists hahahaTab
IF EXISTS (SELECT * FROM sys.xml_schema_collections WHERE name = 'hahahaSch')
  DROP XML SCHEMA COLLECTION hahahaSch
GO

CREATE XML SCHEMA COLLECTION hahahaSch AS
'Sem pride vygenerovana a upravena Schema
';
GO

CREATE TABLE hahahaTab (
  i int,
  x xml (hahahaSch))
GO

-- OK
INSERT INTO hahahaTab VALUES(1,
'Sem pride Xml prvok <Matrix>
')
SELECT * FROM hahahaTab;
-----
-----

```

Hotový T-sql kód so schémou.

Pri zápise do prvého stĺpca inej hodnoty ako 1, 4, 7 alebo 10, systém nehlási/hlási chybu.

KOPIROVAT z gmail !

```

USE tempdb
GO
DROP TABLE if exists hahahaTab
IF EXISTS (SELECT * FROM sys.xml_schema_collections WHERE name = 'hahahaSch')
  DROP XML SCHEMA COLLECTION hahahaSch
GO

CREATE XML SCHEMA COLLECTION hahahaSch AS

```

```
'<?xml version="1.0" encoding="utf-8"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:simpleType name="int1_4_7_10">
    <xs:restriction base="xs:nonNegativeInteger">
      <!-- <xs:maxInclusive value="100"></xs:maxInclusive -->
      <xs:enumeration value="1"></xs:enumeration>
      <xs:enumeration value="4"/>
      <xs:enumeration value="7"/>
      <xs:enumeration value="10"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="Matrix">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="row">
          <xs:complexType>
            <xs:attribute name="Col1" type=" int1_4_7_10" use="required" />
            <xs:attribute name="Col2" type="xs:int" use="required" />
            <xs:attribute name="Col3" type="xs:int" use="required" />
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

</xs:schema>
';
GO
```

```
----- Zoznam schema collections:
--SELECT * FROM sys.xml_schema_collections
----- Zoznam namespaces:
--SELECT name FROM sys.xml_schema_namespaces
```

```
CREATE TABLE hahahaTab (
  i int,
  x xml (hahahaSch))
```

```
GO
```

```
-- OK
INSERT INTO hahahaTab VALUES(5,
'<Matrix>
  <row Col1="1" Col2="2" Col3="3" />
  <row Col1="4" Col2="5" Col3="6" />
  <row Col1="7" Col2="8" Col3="9" />
  <row Col1="10" Col2="11" Col3="12" />
</Matrix>
')
```

```
SELECT * FROM hahahaTab;

-- NO
--INSERT INTO hahahaTab VALUES(5,
-- '<Matrix>
--   <row Col1="3" Col2="2" Col3="3" />
--   <row Col1="4" Col2="5" Col3="6" />
--   <row Col1="7" Col2="8" Col3="9" />
--   <row Col1="10" Col2="11" Col3="12" />
-- </Matrix>
--')
```

```
SELECT * FROM hahahaTab;
```

Příklad: [http://msdn.microsoft.com/en-us/library/ms256095\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/ms256095(v=vs.110).aspx)

4. XML versus JSON

JSON spolu s XML tvorí dnes základný dátový formát pre výmenu údajov na webe. Kým JSON je jednoduchší formát ako XML, jeho rozšírená verzia tvorí základ MongoDB. Preto JSON ilustrujeme s dvomi príkladmi.

1) Transformácia relácie na XML a JSON

```
SELECT ... FOR XML AUTO;  
SELECT ... FOR JSON AUTO;
```

```
SELECT * from Poliklinika..Lekari FOR XML AUTO;  
SELECT * from Poliklinika..Lekari FOR JSON AUTO;
```

```
<Poliklinika..Lekari idL="1" krstne="Oto" spec="Ocny" datNar="1960-05-05T00:00:00" />
```

...

```
[{"idL":1,"krstne":"Oto","spec":"Ocny","datNar":"1960-05-05T00:00:00"}, ...
```

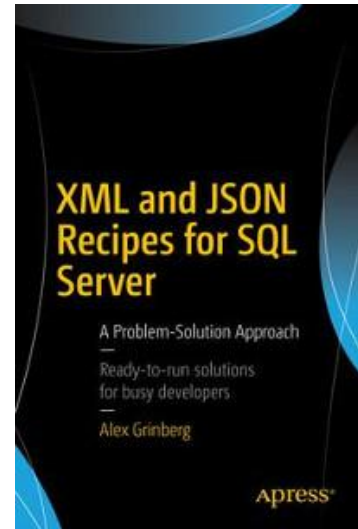
2) Vytvorenie XML (z prvkov a nie z atribútov) a JSON

```
Declare @x XML =
```

```
'<studenti>  
<student>  
  <krstne>Fero</krstne>  
  <priezvisko>Bak</priezvisko>  
</student>  
<student>  
  <krstne>Jano</krstne>  
  <priezvisko>Byk</priezvisko>  
</student>  
<student>  
  <krstne>Stevo</krstne>  
  <priezvisko>Buk</priezvisko>  
</student>  
</studenti>';
```

```
SELECT  
c.value('krstne[1]',  
'varchar(30)') AS krstne  
,c.value('priezvisko[1]',  
'varchar(30)') AS priezvisko  
FROM @x.nodes('//student') t(c) --FOR XML AUTO --PATH
```

```
declare @j nvarchar(max) =  
'{"studenti":  
  [  
    {"krstne":"Fero",  
     "priezvisko":"Bak"},  
    {"krstne":"Jano",  
     "priezvisko":"Byk"},  
    {"krstne":"Stevo",  
     "priezvisko":"Buk",  
     "vek":20}  
  ]  
}';  
SELECT krstne, priezvisko, vek  
FROM OPENJSON (@j, '$.studenti')  
WITH  
(  
  krstne varchar(30),  
  priezvisko varchar(30),  
  vek int  
);
```



```
<studenti>  
  <student krstne="Fero" priezvisko="Bak" />  
- vs  <student krstne="Jano" priezvisko="Byk" />  
      <student krstne="Stevo" priezvisko="Buk"/>  
</studenti>
```

	krstne	priezvisko
1	Fero	Bak
2	Jano	Byk
3	Stevo	Buk

	krstne	priezvisko	vek
1	Fero	Bak	NULL
2	Jano	Byk	NULL
3	Stevo	Buk	20

Dvojica key, value

```
XML: k1 ="v1" k2 ="20"
```

vs

```
JSON: "k1": "v1", "k2": 20
```

5. Chatgpt.com

```
-- chatgpt.com
-- Can you generate a result-table HockeyWCh2002 with ranking of the 2002 hockey world
championship using T-SQL?
use tempdb;
```

```
IF OBJECT_ID('HockeyWCh2002', 'U') IS NOT NULL
    DROP TABLE HockeyWCh2002;
GO
```

```
-- Create the result table for the 2002 Ice Hockey World Championship with ranking
```

```
CREATE TABLE HockeyWCh2002 (
    Rank INT PRIMARY KEY,           -- Final ranking in the championship
    Country NVARCHAR(50) NOT NULL, -- Team name
    MatchesPlayed INT NOT NULL,    -- Total matches played
    Wins INT NOT NULL,             -- Matches won
    Losses INT NOT NULL,          -- Matches lost
    Draws INT NOT NULL,           -- Matches drawn
    GoalsScored INT NOT NULL,     -- Total goals scored
    GoalsConceded INT NOT NULL,   -- Total goals conceded
    GoalDifference AS (GoalsScored - GoalsConceded) PERSISTED, -- Auto-calculated
    Points INT NOT NULL           -- Total points (Win = 3, Draw = 1)
```

```
-- Insert sample data (Replace with actual statistics)
```

```
INSERT INTO HockeyWCh2002 (Rank, Country, MatchesPlayed, Wins, Losses, Draws, GoalsScored,
GoalsConceded, Points) VALUES
(1, 'Slovakia', 9, 7, 1, 1, 35, 20, 22),
(2, 'Russia', 9, 6, 2, 1, 32, 18, 19),
(3, 'Sweden', 9, 5, 2, 2, 33, 22, 17),
(4, 'Finland', 9, 5, 3, 1, 30, 21, 16),
(5, 'Canada', 8, 4, 3, 1, 28, 19, 13),
(6, 'Czech Republic', 8, 4, 3, 1, 27, 18, 13),
(7, 'USA', 8, 3, 4, 1, 25, 22, 10),
(8, 'Germany', 7, 2, 4, 1, 18, 23, 7),
(9, 'Switzerland', 7, 2, 4, 1, 16, 21, 7),
(10, 'Latvia', 7, 1, 5, 1, 14, 25, 4);
```

```
-- Retrieve and display the result table with ranking
```

```
SELECT * FROM HockeyWCh2002 ORDER BY Rank ASC;
```

Rank	Country	MatchesPlayed	Wins	Losses	Draws	GoalsScored	GoalsConceded	GoalDifference	Points
1	Slovakia	9	7	1	1	35	20	15	22
2	Russia	9	6	2	1	32	18	14	19
3	Sweden	9	5	2	2	33	22	11	17
4	Finland	9	5	3	1	30	21	9	16
5	Canada	8	4	3	1	28	19	9	13
6	Czech Republic	8	4	3	1	27	18	9	13
7	USA	8	3	4	1	25	22	3	10
8	Germany	7	2	4	1	18	23	-5	7
9	Switzerland	7	2	4	1	16	21	-5	7
10	Latvia	7	1	5	1	14	25	-11	4

```
-- Are the table data real?
```

The data I provided in the table is not real. It was an example created for demonstration purposes
...

```
-- A poradia? (zmenime jazyk)
```

Áno, . . .

```
-- chatgpt.com (dopytujeme prirodzeným jazykom bez SQL)
-- Ktoré krajiny dostali v tejto tabuľke najmenej gólov?
- Rusko - 18 gólov
- Česká republika - 18 gólov
- Kanada - 19 gólov
```

```
-- Did you use sql query?
```

```
No . . .
```

```
-- Will AI end with SQL, Is this the end of SQL?
```

```
"The way we used to work will change. ... We will use a lot of AI in our jobs."
```

alebo (dané odpovede sme získali v rôznych dňoch)

```
"AI isn't the "end" of SQL, but rather, it's transforming how we interact with data.
```

```
. . .
```

1. AI and SQL are Complementary: . . .
2. AI Can Automate SQL Query Generation: . . .
3. SQL in Modern Data Architectures: . . .
4. AI's Impact on Data Storage & Processing: . . .
5. SQL's Continued Dominance in Business: . . .

```
Conclusion: SQL's Evolution, Not End
```

```
AI is not the end of SQL; it's more of an evolution . . ."
```