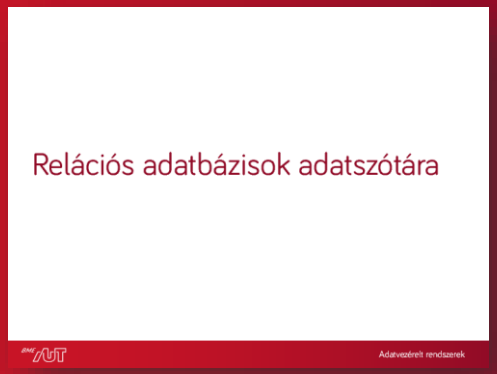
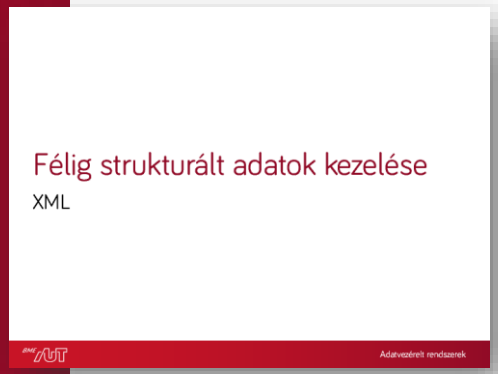
Adatvezérelt rendszerek

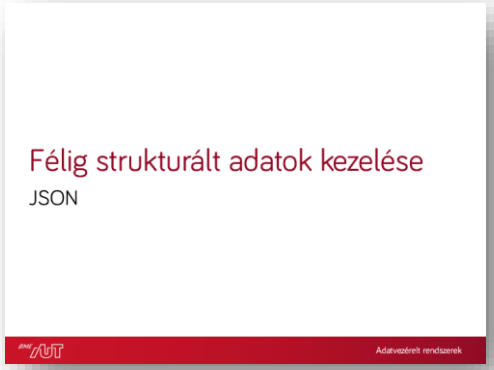
Adatszótárak
Félig strukturált adatok (XML, JSON)

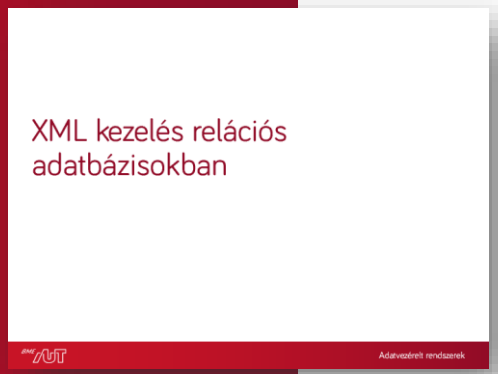
BME AUT
Automatizálási és
Alkalmazott
Informatikai Tanszék

Tartalom


Relációs adatbázisok adatszótára
Adatvezérelt rendszerek


Félig strukturált adatok kezelése
XML
Adatvezérelt rendszerek


Félig strukturált adatok kezelése
JSON
Adatvezérelt rendszerek


XML kezelés relációs adatbázisokban
Adatvezérelt rendszerek


JSON kezelés relációs adatbázisokban
Adatvezérelt rendszerek

Relációs adatbázisok adatszótára

Mire való az adatszótár?

```
IF  EXISTS (SELECT * FROM sys.objects
       WHERE object_id = OBJECT_ID(N'[dbo].Invoice')
              AND type in (N'U'))
  drop table Invoice

create table [Invoice](...)
```

Mire való az adatszótár?

```
IF  EXISTS (SELECT * FROM sys.objects
       WHERE object_id = OBJECT_ID(N'[dbo].Invoice')
              AND type in (N'U'))
  drop table Invoice

create table [Invoice](...)
```

- A minta adatbázisunk létrehozó scriptje
- Miért?
  - Idempotens script: adatbázis állapotától függetlenül újra lefuttatható és ugyanazt az eredményt adja

Adatszótár

- Data dictionary
- Központi helyen tárolt információ az adatról, a formátumról, kapcsolatokról.
  - Pl. táblák nevei, oszlopok nevei, típusai
- (Lehet egy dokumentum is.)
- Adatbáziskezelő integrált része.
  - Csak olvasható nézetek.
  - Felhasználható DML és DDL utasításokban.
    - Pl. séma migráció: hozzuk létre a táblát, ha még nem létezik

Adatszótár tartalma

- Minden séma objektum leírása
  - Táblák, nézetek, indexek, szekvenciák, tárolt eljárások, …
- Integritási kritériumok
- Felhasználók, jogosultságok
- Monitoring információk
  - Pl. aktív kapcsolatok száma, használt zárak
- Auditing információ
  - Pl. ki módosított egyes séma objektumokat

MS SQL adatszótár

- Information Schema Views
  - ISO standard szerint
  - `INFORMATION_SCHEMA.`
    - `TABLES, VIEWS, COLUMNS, PARAMETERS, TABLE_PRIVILEGES, …`
- Catalog Views
  - Teljes körű információ a szerverről.
  - `sys.`
    - `databases, database_files, filegroups, messages, schemas, objects, tables, columns, foreign_keys,`
- Dynamic Management Views
  - Szerver diagnosztikai információk.
  - `sys.dm_tran_locks, sys.dm_exec_cached_plans, sys.dm_exec_sessions`

MS SQL adatszótár példa

```
select * from sys.tables
select * from INFORMATION_SCHEMA.TABLES
select * from sys.objects
select * from INFORMATION_SCHEMA.COLUMNS
```

séma	tábla	oszlop	index	default	nullable	típus
dbo	VAT	ID	1	NULL	NO	int
dbo	VAT	Perc	2	NULL	YES	int
dbo	PaymentMethod	ID	1	NULL	NO	int
dbo	PaymentMethod	Mode	2	NULL	YES	nvarchar
dbo	PaymentMethod	Deadline	3	NULL	YES	int
dbo	Status	ID	1	NULL	NO	int

MS SQL adatszótár példa

```
IF EXISTS (
      SELECT * FROM sys.objects
      WHERE type = 'U' AND name = 'Product')
DROP TABLE Product
```

```
if NOT EXISTS(
      SELECT * FROM INFORMATION_SCHEMA.COLUMNS
      WHERE TABLE_NAME = 'Product',
      AND COLUMN_NAME = 'Description')
alter table Product add Description xml;
```

Félig strukturált adatok kezelése

XML

XML: Extensible Markup Language

- Adatok szöveges, platformfüggetlen reprezentációja.
- Emberi szemmel és programmal is olvasható.
- Célja: egyszerű, általános használat.
- Eredetileg dokumentum leírásnak készült.
  - Sok más helyen is használják: pl. RSS, Atom, SOAP, OpenXML, XHTML, OpenDocument, …
- Önleíró.

```
<course>
  <title>Adatvezérelt rendszerek</title>
  <code>VIAUAC01</code>
</course>
```

XML dokumentum elemei

```
<?xml version="1.0"
      encoding="UTF-8"?>
<!-- komment -->
<elem attributum="érték">
  <tag>tartalom</tag>
  <![CDATA[ bármilyen tartalom ]]>
</elem>
```

XML deklaráció

XML - példa

```
public class Customer
{
    public string Name;
    public DateTime Registered;
    public Address Address;
}
```

```
public class Address
{
    public string City;
    public int ZipCode;
}
```

```
<?xml version="1.0"?>
<Customer>
  <Name>Nagy Ádám</Name>
  <Registered>2016-10-26T08:58:26.6412829+02:00</Registered>
  <Address>
    <City>Budapest</City>
    <ZipCode>1118</ZipCode>
  </Address>
</Customer>
```

Encoding

- A karaktereket hogyan fordítjuk le byte-okra
- ASCII: 0-127 angol karakterek -> 1 byte / karakter
  - A kódkészlet (code page) jelzi, hogy a 127 fölött lévő kódok milyen karaktert jelentenek
  - Például a ISO/IEC 8859-1 nem tartalmaz 'ő'-t, csak 'õ'-t
- Unicode: minden absztrakt karakter kap egy számot
- UTF-8: változó hosszúságú kódolás 1 vagy 4
- UTF-16 : változó hosszúságú kódolás 2 vagy 4
- XML: a '<?xml' stringből tudja a parser, hogy 1 vagy 2 byte-os, high endian stb., de a kódkészlet még mindig szükséges

Névterek

- Mint a C++, C# namespace, Java package
- Tag nevek szabadon választhatóak, ütközés.
- Névtér egy prefix: `<ns:tag>`
- Deklarárlni kell: *`xmlns:ns="URI"`*

```
<ns:tag xmlns:ns="http://www.aut.bme.hu">

<root xmlns:ns="http://www.aut.bme.hu">
   <ns:tag>abc<ns:tag>
</root>
```

- Alapértelmezett namespace

```
<root xmlns="http://www.aut.bme.hu">
```

Névterek példa: HTML és XSLT

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org
/1999/XSL/Transform"><xsl:template match="/">
<html><body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr>
      <th style="text-align:left">Title</th>
      <th style="text-align:left">Artist</th>
    </tr>
    <xsl:for-each select="catalog/cd">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
    </tr>
    </xsl:for-each>
  </table>
</body></html>
</xsl:template></xsl:stylesheet>
```

XML – előnyök / hátrányok

- Szöveges adat reprezentáció
  - Platformfüggetlen
  - Szabványos megoldások (pl. SOAP)
  - Dokumentált séma, pl. XSD
- Típusos
  - Séma leírás, származás stb.
  - Gráfok reprezentációja nehézkes
- Nem egyértelmű adatreprezentáció
  - Attribútum? Gyerek elem? Null?
- Szöveges -> nagyobb méret

XML - .NET

- System.Xml.Serialization.XmlSerializer

```
var ser = new XmlSerializer(typeof(C));
ser.Serialize(<stream>, <obj>);
myobj = (C)ser.Deserialize(<stream>);
```

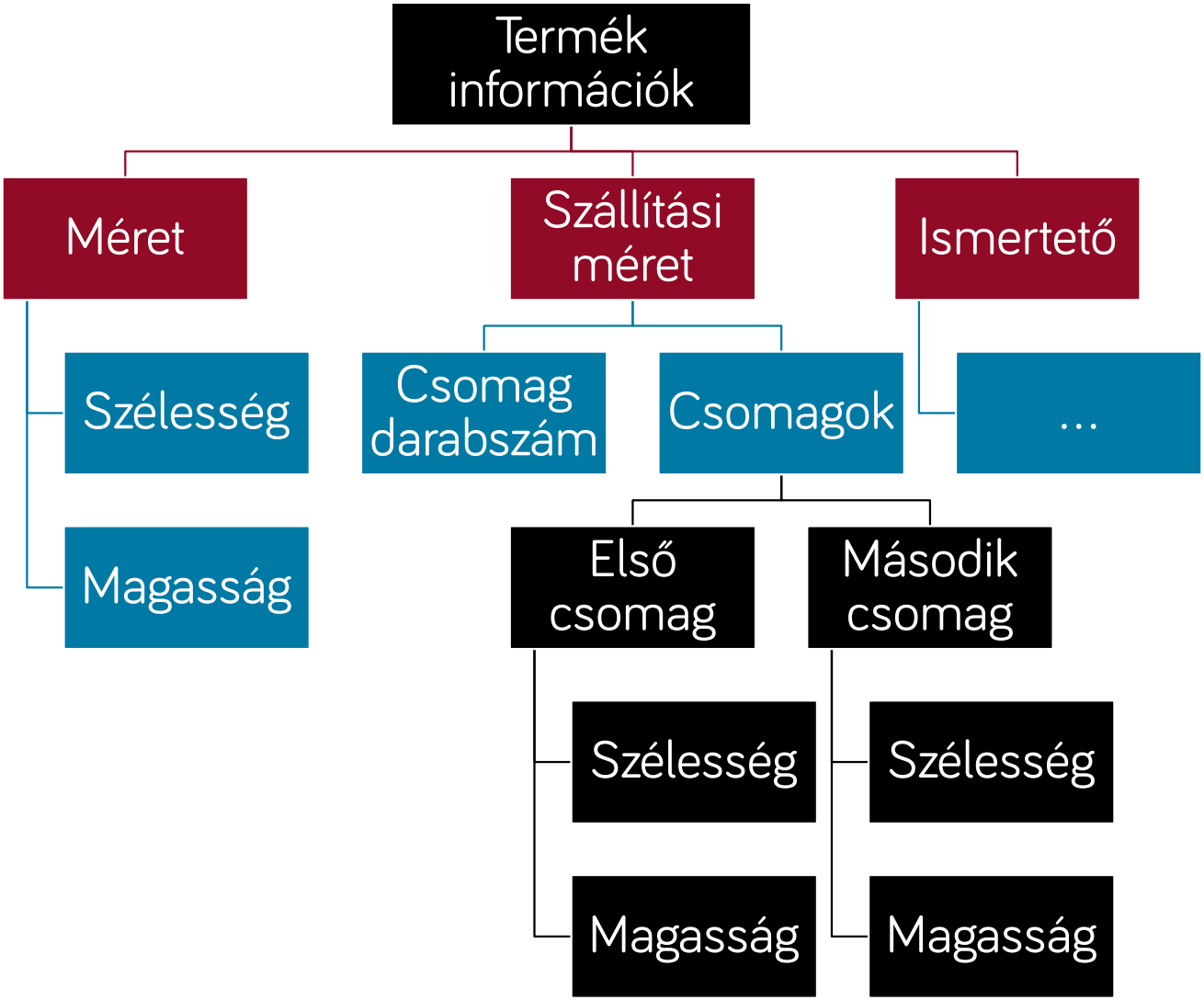
- Testreszabás attribútumokkal

```
[XmlElement("Cim")]
public class Address
{
    [XmlAttribute("Varos")]
    public string City;
    [XmlIgnore]
    public int SzamoltTavolsag;
}
```

Séma

- Xml dokumentum **jól formázott**, ha
  - Szintaktikailag megfelelő a tartalma.
    - minden nyitó tag lezárt, zárójelezés szabályai szerint
    - egyetlen gyökér eleme van
    - Stb.
- Tartalom **érvényes**sége más kérdés
  - Jó névvel vannak benne a tagek?
  - Olyan tartalom van a tagekben, amire számítunk?
  - Sémával (saját nyelvtannal) leírható a várt tartalom, pl. DTD, XSD
- Validálás: egy adott XML dokumentum megfelel-e egy adott sémának
  - Programozottan eldönthető

DOM: Document Object Model



XPath

- Standard XML lekérdező nyelv
- Lekérdezi az XML dokumentum egy részét
  - Egy csomópontot vagy boolean/szám/szöveg adatot ad vissza
- Szorosan kapcsolódik az XSLT-hez

XPath

```
<konytar>
 <cim>1118 Budapest ...</cim>
 <konyv>
    <cim nyelv="en">Harry Potter</cim>
    <ar>1234</ar>
 </konyv>
 <konyv>
    <cim nyelv="hu">Adatvezerelt rendszerek</cim>
    <ar>5678</ar>
 </konyv>
</konyvtar>
```

XPath

```
<konytar>
 <cim>1118 Budapest ...</cim>
 <konyv>
    <cim nyelv="en">Harry Potter</cim>
    <ar>1234</ar>
 </konyv>
 <konyv>
    <cim nyelv="hu">Adatvezerelt rendszerek</cim>
    <ar>5678</ar>
 </konyv>
</konyvtar>
```

konyvtar/konyv

/konyvtar/konyv

XPath

tagnev	Csomópont névvel
/	Gyökértől kezdve
//	Aktuális csomóponttól kezdve bármely leszármazottban
.	Aktuális csomópont
..	Szülő csomópont
@nev	Adott nevű attribútum
/konyvtar/konyv[1]	Valahányadik gyerek (1-től indexelt)
/konyvtar/konyv[last()]	Utolsó gyerek
/konyvtar/konyv[position()<3]	Első kettő gyerek
//cim[@nyelv='en']	Olyan title elem, aminek a nyelv attribútuma en értékű

XPath

```
<konytar>
 <cim>1118 Budapest ...</cim>
 <konyv>
    <cim nyelv="en">Harry Potter</cim>
    <ar>1234</ar>
 </konyv>
 <konyv>
    <cim nyelv="hu">Adatvezerelt rendszerek</cim>
    <ar>5678</ar>
 </konyv>
</konyvtar>
```

//konyv

XPath

```
<konytar>
 <cim>1118 Budapest ...</cim>
  <konyv>
    <cim nyelv="en">Harry Potter</cim>
    <ar>1234</ar>
  </konyv>
  <konyv>
    <cim nyelv="hu">Adatvezerelt rendszerek</cim>
    <ar>5678</ar>
  </konyv>
</konyvtar>
```

//cim

XPath

```
<konytar>
 <cim>1118 Budapest ...</cim>
 <konyv>
    <cim nyelv="en">Harry Potter</cim>
    <ar>1234</ar>
 </konyv>
 <konyv>
    <cim nyelv="hu">Adatvezerelt rendszerek</cim>
    <ar>5678</ar>
 </konyv>
</konyvtar>
```

//@nyelv

XPath

```
<konytar>
 <cim>1118 Budapest ...</cim>
 <konyv>
    <cim nyelv="en">Harry Potter</cim>
    <ar>1234</ar>
 </konyv>
 <konyv>
    <cim nyelv="hu">Adatvezerelt rendszerek</cim>
    <ar>5678</ar>
 </konyv>
</konyvtar>
```

/konyvtar/konyv[1]

XPath

```
<konytar>
 <cim>1118 Budapest ...</cim>
 <konyv>
    <cim nyelv="en">Harry Potter</cim>
    <ar>1234</ar>
 </konyv>
 <konyv>
    <cim nyelv="hu">Adatvezerelt rendszerek</cim>
    <ar>5678</ar>
 </konyv>
</konyvtar>
```

/konyvtar/konyv[ar>5000]

XPath

```
<konytar>
 <cim>1118 Budapest ...</cim>
 <konyv>
    <cim nyelv="en">Harry Potter</cim>
    <ar>1234</ar>
 </konyv>
 <konyv>
    <cim nyelv="hu">Adatvezerelt rendszerek</cim>
    <ar>5678</ar>
 </konyv>
</konyvtar>
```

/konyvtar/konyv/ar[text()]

Félig strukturált adatok kezelése

JSON

JSON

- JavaScript Object Notation
  - Nem csak JavaScript!
- Kompakt, olvasható, szöveges reprezentáció
- Egy memóriabeli objektum egy JSON objektum
- Alapelemei:
  - *Objektum*
    - Kulcs-érték párok halmaza
  - Tömb
    - Értékek halmaza
  - Érték
    - Szöveg, szám, igaz/hamis, null, *objektum*, tömb

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

JSON - problémák

- Nincs komment
- Byte order mark nem lehet a fájl elején
  - Nincs rá szükség, az első karakter kódja mindig kisebb, mint 128
- Gyakori adat típusokra nincs egyértelmű reprezentáció
  - Pl. dátum
  - A szabvány nem határozza meg
  - Külön leírás szükséges a parsoláshoz
- Biztonsági kockázat
  - Tipikus, de nem szerencsés gyakorlat: JSON eredményt JavaScript motorral végrehajtunk (`eval()`)

JSON Schema

- Séma leírás
  - Mint az XSD XML-hez
- Maga is JSON fájl

JSON Schema

```
{ "$schema": "http://json-schema.org/schema#",
  "title": "Product",
  "type": "object",
  "required": ["id", "name"],
  "properties": {
    "id": {
      "type": "number",
      "description": "Product identifier"
    },
    "name": {
      "type": "string",
    },
    "stock": {
      "type": "object",
      "properties": {
        "warehouse": { "type": "number"  },
        "retail": { "type": "number" }
      }
    } } }
```

JSON – mikor használjuk

- Backend – vékonykliens kommunikáció
  - Tömör, rövid
    - Kevés hálózati forgalom, mobil klienseknek előnyös
  - JavaScript tudja parsolni
    - Webes rendszerekben
- REST
  - Lásd később
- JSON adatbázisban
  - MS SQL Server 2016, Oracle Server 12c
  - MongoDB → látni fogjuk

.NET

- System.Text.Json

```
var weatherForecast = new WeatherForecast {
     Date = DateTime.Parse("2019-08-01"),
     TemperatureCelsius = 25,
     Summary = "Hot" };

string jsonString =
     JsonSerializer.Serialize(weatherForecast);


WeatherForecast? weatherForecast =
     JsonSerializer.Deserialize<WeatherForecast>(jsonString);
```

XML <-> JSON

	XML	JSON
Adat típusok	Több beépített adattípus.	Pár skalár, objektum, tömb.
Tömbök	Nem ismeri, de reprezentálható.	Definiált fogalom.
Objektumok	Nem ismeri, több féle képen reprezentálható.	Definiált fogalom.
Null érték	xsi:nil (+névtér import)	Van
Komment	Van	Nincs
Névtér	Van	Nincs
Reprezentáció, formázás	Több lehetséges megoldás, nem egyértelmű.	Egyértelmű, kivéve dátum.
Méret	Hosszabb	Kompakt, az adat foglalja el.
Parsolás JavaScript-ben	Bonyolultabb.	Támogatott.
Szükséges ismeretek	Több technológia együttesen.	JavaScript

XML kezelés relációs adatbázisokban

Félig strukturált adatok – példa eset

FRIHETEN

Sarokkanapé tárolóval, Skiftebo sszürke

114.900 Ft

Az ár a kiválasztott feltételekre vonatkozik
Az ár az áfát tartalmazza.

Cikkszám:392.167.54

4.1 (11) Értékelés

Egy kiadós éjszakai alvás után erőfeszítés nélkül újra átalakíthatod a hálószobádat vagy vendégszobádat nappalivá. A beépített tárolóhoz könnyű hozzáférni és elég tágas ahhoz, hogy ágyneműt, könyveket és pizsamát tárolj benne.

Huzat: Skiftebo sszürke

mennyiség: 1

Vásárolj online / add a terméket a bevásárlólistához

Vedd meg a legközelebbi IKEA áruházban!

Áruház: Válassz:

Az árak áruházanként eltérhetnek! Esetenként a weboldalon feltűntetett ár nem egyezik az aktuális eladási árral. Az áruházi raktárkészlet megegyezik a weboldalon és a telefonos információ által adott adatokkal.

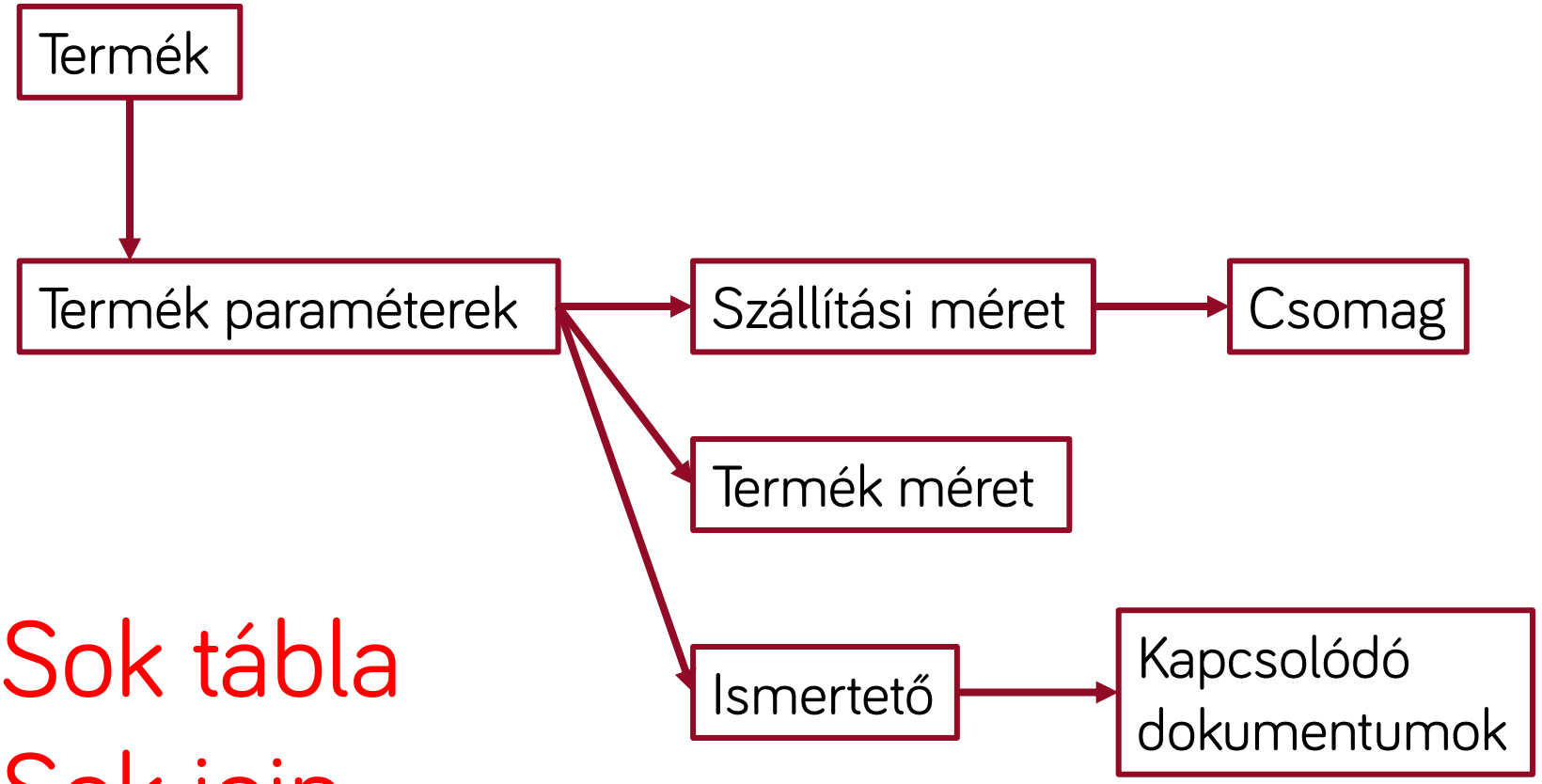
Az áruk összeszedése

Termék információ | Anyagok és környezet | Összeszerelés és más dokumentumok | Csomagméretek | Vélemények

Csomagok: 3

Cikkszám	Csomagok	Szélesség	Magasság	Hosszúság	Átmérő	Súly
00311059	1	83 cm	41 cm	141 cm	-	35.00 kg
80311060	1	78 cm	32 cm	207 cm	-	40.55 kg
80311734	1	73 cm	40 cm	146 cm	-	42.35 kg

Félig strukturált adatok – relációs példa



Sok tábla

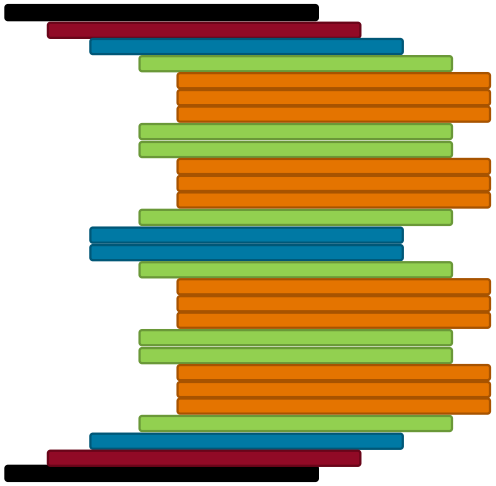
Sok join

Félig strukturált adatok – XML példa

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<product>
   <shipping_size>
      <number_of_packages>2</number_of_packages>
      <package id="1">
         <size>
            <width>80</width>
            <height>20</height>
            <depth>40</depth>
         </size>
      </package>
      <package id="2">...</package>
   </shipping_size>
   < product_size>...</product_size>
   <description>...</description>
</product>
```

Félig strukturált adatok adatbázisban

- Létező xml formájú adatok.
- Ismeretlen, nem definiált formájú adat.
- Külső rendszerből ilyen formán érkeznek, vagy külső rendszernek ilyen formában kell átadni.
- Csak tárolt, nem manipulált adattartalom.
- Mélyen egymásba ágyazott adatformátum.



Félig strukturált adatok adatbázisban

ID	Név	…	
123	Gardrób szekrény	…	<?xml>…
456	Sarokkanapé	…	<?xml>…

XML tárolása relációs adatbázisban

- Xml-képes relációs adatbázisok.
  - Microsoft SQL, Oracle, PostgreSql, …
- Relációs adatok **mellett** xml adat is.
- A relációs a fő tartalom, abból van több.
- Xml adat köthető a relációshoz.
  - Pl. a termék adatai az ár és név mellett egy xml leírás

Adattípus

- nvarchar(max)
  - Validáció nélküli szöveg.
  - Tartalmat „betűről betűre” megőrzi.
  - Futási időben konvertálható (költséges).
- xml
  - Jólformázottnak kell lennie.
  - Csatolható hozzá séma, automatikusan ellenőrzi a megfelelést.
  - Kereshető, lekérdezhető (pl. egy tag tartalma).
  - Manipulálható (pl. törölhető egy adott tag).
  - Index definiálható rá.

```
create table Product(
    Name nvarchar(100),
    Description XML )
```

Index xml típusú oszlopra

- Csak ha keresünk az xml adatban
  - Az egész xml adat lekérdezéséhez nem használja az indexet
- Két fajta index:
  - Elsődleges: teljes tartalmat indexeli
    - Egy darab ilyen indexet definiálható
    - Ha indexelt az oszlop, egy ilyen indexnek léteznie kell

```
CREATE PRIMARY XML INDEX idxname on Table(Col)
```

  - Másodlagos: konkrét xml elemre definiált
    - Tetszőleges darabszámú definiálható
    - Tovább segíti az optimalizációt

```
CREATE XML INDEX idxname2 ON Table(Col)
USING XML INDEX idxname FOR VALUE;
```

Séma hozzárendelés xml oszlophoz

- Az adat validációját automatikusan elvégzi a rendszer a séma szerint
  - Mint egy tartományi integritási kritérium
- Lekérdezés optimalizáláshoz is használja
- Opcionális, nem kötelező sémát hozzárendelni

Lekérdezés

- query(XQuery)

```
> select Description.query('/product/num_of_packages')
  from Product
    <num_of_packages>1</num_of_packages>
```

- value(XQuery, SQLType)

```
> select Description.value(
   '(/product/num_of_packages)[1]','int') from Product
    1
```

- exist (XQuery)

```
> select Name from Product
  where Description.exist(
        '/product/num_of_packages eq 2')=1
```

Manipulálás

- modify()

```
update Product
set Description.modify(
'replace value of
(/product/num_of_packages/text())[1] with "2"')
where ID=8

update Product
set Description.modify(
'insert <a>1</a> after (/product)[1]')
where ID=8

update Product
set Description.modify('delete /product/a')
where ID=8
```

FOR XML

- Lekérdezés eredményének konvertálása XML formába

```
select ID, Name from Customer
for xml auto
```

```
<Customer ID="1" Name="Puskás Norbert" />
<Customer ID="2" Name="Hajdú Katalin" />
<Customer ID="3" Name="Grosz János" />
```

JSON kezelés relációs adatbázisokban

MS SQL JSON támogatás

- Nincs speciális adattípus, mint az XML-nél, NVARCHAR-ban tárolja a JSON-t
  - Nincs speciális index sem
- Funkciók
  - **ISJSON**: adott sztring json-e
  - **JSON_VALUE**: a jsonből egy skalár értéket emel ki
  - **JSON_QUERY**: egy json darabot ad vissza (objektumot vagy tömböt jsonben)
  - **JSON_MODIFY**: hasonló az xml.modify-hoz
    - update / delete / insert
  - **OPENJSON**: SQL sorokat ad vissza jsonből
  - **FOR JSON**: json formátumba adja vissza a lekérdezés eredményét

Lekérdezés példa

- A lekérdezésekben a relációs és jsonből származó adatokat együtt lehet használni

```
SELECT Name,
    Surname,
    JSON_VALUE(jsonCol, '$.info.address.PostCode') AS PostCode,
    JSON_VALUE(jsonCol, '$.info.address."Address Line 1"')
        + ' ' + JSON_VALUE(jsonCol, '$.info.address."Address Line 2"') AS Address,
    JSON_QUERY(jsonCol, '$.info.skills') AS Skills
FROM People
WHERE ISJSON(jsonCol) > 0
    AND JSON_VALUE(jsonCol, '$.info.address.Town') = 'Belgrade'
    AND STATUS = 'Active'
ORDER BY JSON_VALUE(jsonCol, '$.info.address.PostCode');
```

Lekérdezés json-ből példa

- Változóból (például sp paraméter)

```
DECLARE @json NVARCHAR(MAX);

SET @json = N'[
  {"id": 2, "info": {"name": "John", "surname": "Smith"}, "age": 25},
  {"id": 5, "info": {"name": "Jane", "surname": "Smith"}, "dob": "2005-11-04T12:00:00"}
]';

SELECT *
FROM OPENJSON(@json) WITH (
    id INT 'strict $.id',
    firstName NVARCHAR(50) '$.info.name',
    lastName NVARCHAR(50) '$.info.surname',
    age INT,
    dateOfBirth DATETIME2 '$.dob'
    );
```

ID	firstName	lastName	age	dateOfBirth
2	John	Smith	25	
5	Jane	Smith		2005-11-04T12:00:00

Exportálás jsonbe

- Egy query eredményét

```
SELECT id,
    firstName AS "info.name",
    lastName AS "info.surname",
    age,
    dateOfBirth AS dob
FROM People
FOR JSON PATH;
```

```
[
  {
    "id": 2,
    "info": {
      "name": "John",
      "surname": "Smith"
    },
    "age": 25
  },
  {
    "id": 5,
    "info": {
      "name": "Jane",
      "surname": "Smith"
    },
    "dob": "2005-11-04T12:00:00"
  }
]
```