

Adatbázisok 1.

XML – 2. rész

Document Type Definitions (DTD)

XML séma

XML séma (*XML schema*)

- Az XML sémák segítségével szintén XML dokumentumok szerkezetét adhatjuk meg. Itt több megszorítást lehet előírni, mint a DTD-k esetén.
- Az XML séma maga is egy XML dokumentum.

Az XML séma dokumentum szerkezete

```
<? xml version = ... ?>
```

```
<xs:schema xmlns:xs =  
    "http://www.w3.org/2001/XMLSchema">  
    . . .
```

```
</xs:schema>
```

A schema tehát az xs
névtérhez tartozó elem.

Az "xs" *névtér* (*namespace*) adja meg,
amit a megadott URL azonosít.
Itt a névtérhez tartozó elemek
leírása is megtalálható.

Az `xs:element` elem

- Az attribútumai:
 1. `name` = a definiált elem neve (tagben miként szerepel).
 2. `type` = az elem típusa.
 - Lehet XML séma típus, pl. `xs:string`.
 - Vagy egy olyan típus, amit az adott XML sémában deklarálunk.

Példa: xs:element

```
<xs:element name = "név"  
    type = "xs:string" />
```

- A következő elemet írja le:

```
<név>Joe teázója</név>
```

EMPTY

Séma definíció szinten
az xs:element tag-nek nincs
aleleme (<xs:element ... />)

Összetett típusok (*complex types*)

- Az alelemeket tartalmazó elemek leírásához az **xs:complexType** használjuk.
 - A **name** attribútummal nevet adhatunk ennek a típusnak.
- Az **xs:complexType** egy tipikus aleleme az **xs:sequence**, amihez **xs:element** elemek egy sorozata tartozik.
 - A **minOccurs** és **maxOccurs** attribútumok használatával az adott **xs:element** előfordulásainak számát korlátozhatjuk.

Példa: típus a teához

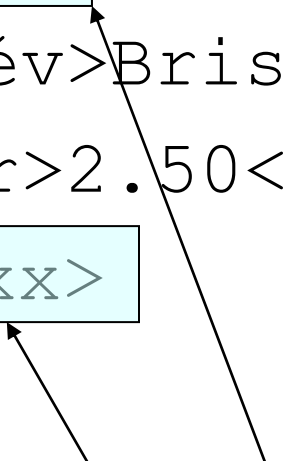
```
<xs:complexType name = "teaTípus">
  <xs:sequence>
    <xs:element name = "név"
      type = "xs:string"
      minOccurs = "1" maxOccurs = "1" />
    <xs:element name = "ár"
      type = "xs:float"
      minOccurs = "0" maxOccurs = "1" />
  </xs:sequence>
</xs:complexType>
```

Pontosan egy előfordulás.

Mint a ? a DTD-ben.

Egy tea típusú elem

```
<xxx>  
<név>Brisk</név>  
<ár>2.50</ár>  
</xxx>
```



Nem ismerjük az ilyen típusú elem nevét.
Az előbbi dián csak magát a típust definiáltuk, a konkrét
elemet – `xs:element` taggel – még nem, amelyik
majd a típust használná!

Egy tea típusú elem – kiegészítés

```
<xs:element name = "tea" type = "teaTípus" />
```

- A következő elemet írja le:

```
<tea>
```

```
  <név>Brisk</név>
```

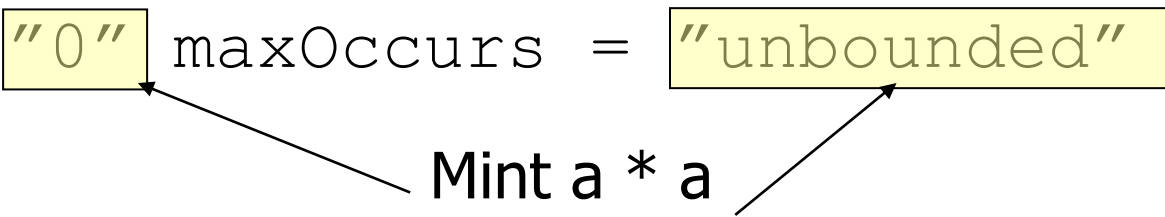
```
  <ár>2.50</ár>
```

```
</tea>
```

Példa: típus a teázókhoz

```
<xs:complexType name = "teázóTípus">
  <xs:sequence>
    <xs:element name = "név"
      type = "xs:string"
      minOccurs = "1" maxOccurs = "1" />
    <xs:element name = "tea"
      type = "teaTípus"
      minOccurs = "0" maxOccurs = "unbounded" />
  </xs:sequence>
</xs:complexType>
```

Mint a * a DTD-ben.



xs:attribute

- **xs:attribute** elemek használatával az összetett típuson belül a típushoz tartozó elemek attribútumait adhatjuk meg.
- Az **xs:attribute** elem attribútumai:
 - **name** és **type** mint az **xs:element** esetén.
 - **use** = "required" vagy "optional".

Példa: xs:attribute

```
<xs:complexType name = "teaTípus">  
  <xs:attribute name = "név"  
    type = "xs:string"  
    use = "required" />  
  <xs:attribute name = "ár"  
    type = "xs:float"  
    use = "optional" />  
</xs:complexType>
```

Az új teaTípusnak megfelelő elem

`<xxx` név = "Brisk"
ár = "2.50" `/>`

Itt sem tudjuk az
elem nevét.

Üres elemmel van
dolgunk, mert
nincsenek alelemei.

Egyszerű típus *(simple type)* megszorítások

- Az `xs:simpleType` segítségével felsorolásokat adhatunk meg és az alaptípusra vonatkozó megszorításokat.
- `name` attribútuma van és
- `xs:restriction` aleleme.

Megszorítások: `xs:restriction`

- A `base` attribútum adja meg, hogy melyik egyszerű típusra (simple type) vonatkozik a megszorítás: pl. `xs:integer`.
- `{min, max}{Inclusive, Exclusive}` a négy attribútum használatával alsó és felső korlátokat adhatunk meg.
- `xs:enumeration` alelem, a `value` attribútuma után megadhatjuk a felsorolás elemeit.

Példa: engedély attribútum a teázókhoz

```
<xs:simpleType name = "engedély">  
  <xs:restriction base = "xs:string">  
    <xs:enumeration value = "minden" />  
    <xs:enumeration value = "csak tea" />  
    <xs:enumeration value = "csak üdítő" />  
  </xs:restriction>  
</xs:simpleType>
```


Példa: az árak [1,5) intervallumba eshetnek

```
<xs:simpleType name = "ár">  
  <xs:restriction  
    base = "xs:float"  
    minInclusive = "1.00"  
    maxExclusive = "5.00" />  
</xs:simpleType>
```

Példa: DTD és XML séma

- **Legyen az alábbi egyszerű DTD példa:**

```
<!DOCTYPE teázók [  
  <!ELEMENT teázók (teázó*)>  
  <!ELEMENT teázó (tea+)>  
    <!ATTLIST teázó név CDATA #REQUIRED>  
  <!ELEMENT tea EMPTY>  
    <!ATTLIST tea név CDATA #REQUIRED>  

```

Példa: DTD és XML séma

- **Ugyanez XML-sémaként:**

```
<? xml version = "1.0" encoding = "utf-8" ?>
```

```
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
```

```
  <xs:complexType name = "teaTípus">
```

```
    <xs:attribute name = "név"
```

```
      type = "xs:string"
```

```
      use = "required" />
```

```
  </xs:complexType>
```

...

Példa: DTD és XML séma

- **Ugyanez XML-sémaként:**

...

```
<xs:complexType name = "teázóTípus">
  <xs:sequence>
    <xs:element name = "tea"
      type = "teaTípus"
      minOccurs = "1" maxOccurs = "unbounded" />
  </xs:sequence>
  <xs:attribute name = "név"
    type = "xs:string"
    use = "required" />
</xs:complexType>
```

...

Példa: DTD és XML séma

- **Ugyanez XML-sémaként:**

...

```
<xs:complexType name = "teázókTípus" >
    <xs:sequence>
        <xs:element name = "teázó"
            type = "teázóTípus"
            minOccurs = "0" maxOccurs = "unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:element name = "teázók" type = "teázókTípus" />
</xs:schema>
```

Példa: DTD és XML séma

- **Ugyanez XML-sémaként – alternatív megoldás:**

...

```
<xs:element name = "teázók">
  <xs:complexType>
    <xs:sequence>
      <xs:element name = "teázó"
        type = "teázóTípus"
        minOccurs = "0" maxOccurs = "unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

XML séma használata

- Az előbbi XML-sémát tegyük fel, hogy kimentettük **teahouses.xsd** néven.
- Ha ezt fel szeretnénk használni egy XML dokumentumban, mint sémát, akkor az alábbi részlethez hasonlóan kell szerepelni:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<teázók
```

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
  xsi:noNamespaceSchemaLocation="teahouses.xsd">
```

```
...
```

```
</teázók>
```

Kulcsok az XML sémában

- Az **xs:element** elemhez tartozhat **xs:key** alelem.
- **Jelentése**: ezen az elemen belül, minden elem, ami egy adott **szelektor** (**selector**) ösvényen keresztül elérhető, egyedi értékekkel kell, hogy rendelkezzen a megadott mezőinek (**field**) kombinációin (alelem, attribútum).
- **Példa**: egy teázók elemen belül a teázó elemek **név** attribútumának egyedinek kell lennie.

Példa: kulcs

```
<xs:element name = "teázók" ... >
```

```
. . .
```

```
<xs:key name = "teázóKulcs">
```

```
<xs:selector xpath = "teázó" />
```

```
<xs:field xpath = "@név" />
```

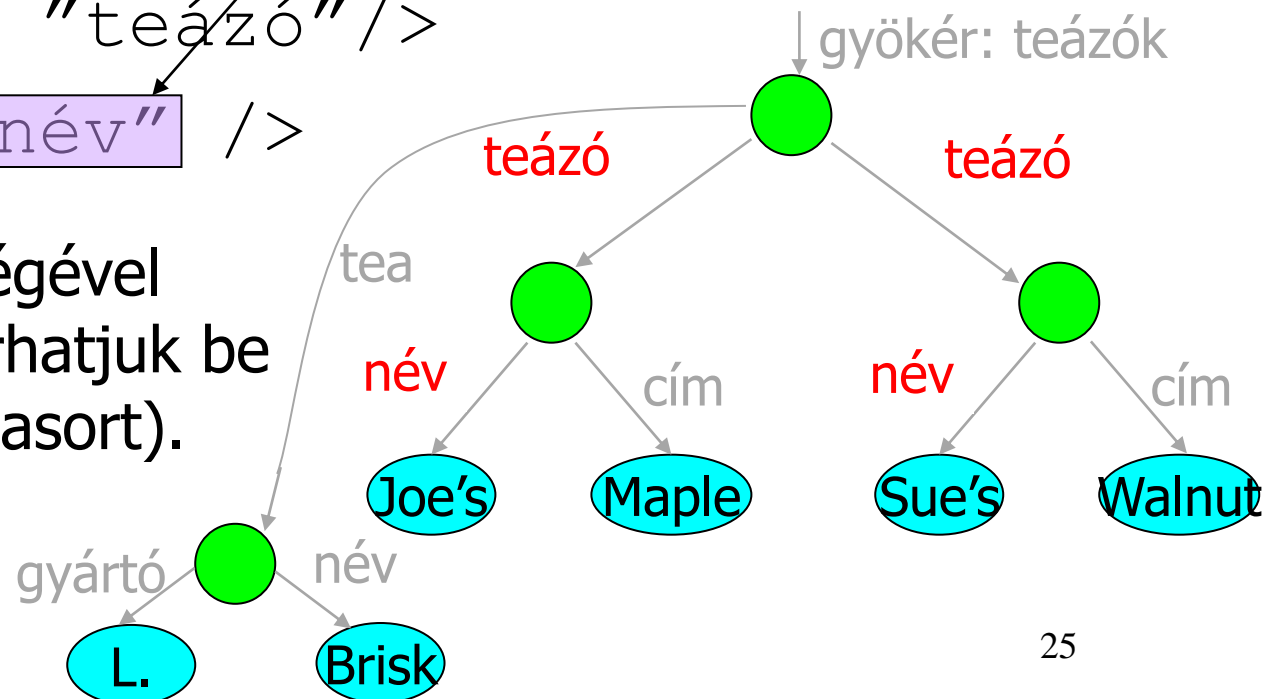
```
</xs:key>
```

```
. . .
```

```
</xs:element>
```

A @
attribútumot
jelöl.

Az XPath segítségével
az XML fákat járhatjuk be
(ld. a későbbi diaszt).



Idegen kulcsok

- A **xs:keyref** alelem az **xs:element** elemen belül előírja, hogy ezen az elemen belül bizonyos értékek, melyeket ugyanúgy a szelektor és mezők használatával adhatunk meg, egy kulcs értékei között kell, hogy szerepeljenek.

Példa: idegen kulcs

- Tegyük fel, hogy a *név* alelem kulcs a *teázó* elemekre vonatkozóan.
 - A kulcs neve legyen *teázóKulcs*.
- A vendég elemhez *látogat* alelemeket szeretnénk hozzáadni. Ezen elemek *teázó* attribútuma *idegen kulcs* lesz, a *teázó* elem *név* attribútumára hivatkozik.

Példa: idegen kulcs az XML sémában

```
<xs:element name = "vendégek"  
    . . .  
    <xs:keyref name = "teázóRef"  
        refer = "teázóKulcs">  
        <xs:selector xpath =  
            "vendég/látogat" />  
        <xs:field xpath = "@teázó" />  
    </xs:keyref>  
</xs:element>
```

Keretrendszer

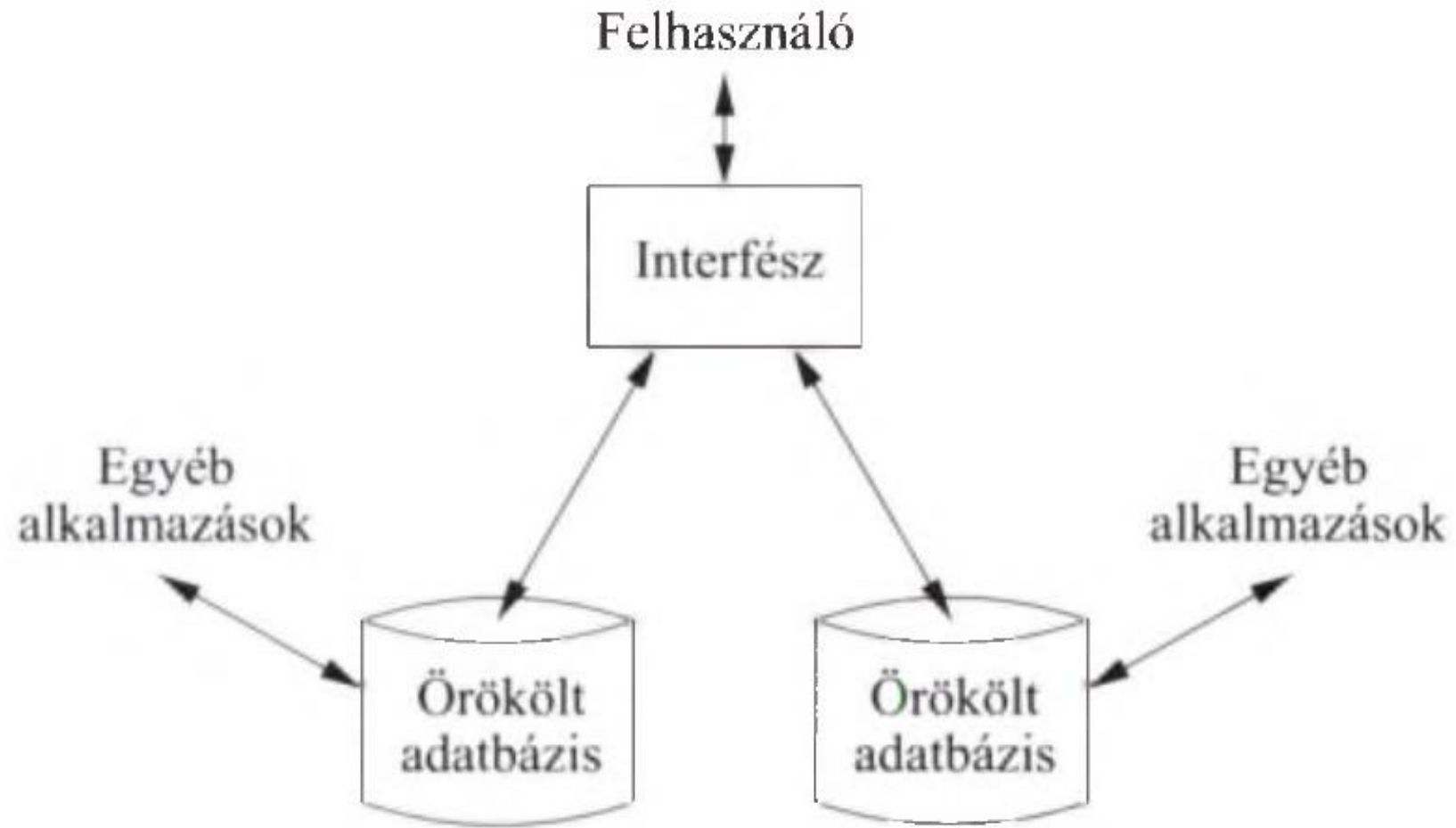
1. *Információintegráció*: A különböző helyekről származó adatbázisokat úgy üzemeltetni, mintha egységes egésznek alkotnának.
2. *Félig-strukturált adat*: Viszonylag új adatmodell, amely segít megbirkózni az adatintegráció problémájával.
3. *XML*: Szabványos nyelv a félig-strukturált adatok leírására.

Információintegrációs probléma

- Egymáshoz kapcsolható adatelemek sok helyen léteznek és elvileg együttműködésre alkalmasak volnának:
- De a különböző adatbázisok több tekintetben különböznek:
 1. Adatbázis modellek (relációs, objektum-orientált, NoSQL, dokumentum stb.)
 2. Séma (normalizált, nem normalizált)
 3. Szakkifejezések: tanácsadó alkalmazott-e? Visszavonult nyugdíjas-e? Alvállalkozó?
 4. Konvenciók (méter kontra láb [metrikus (SI, CGI), birodalmi]).

Példa

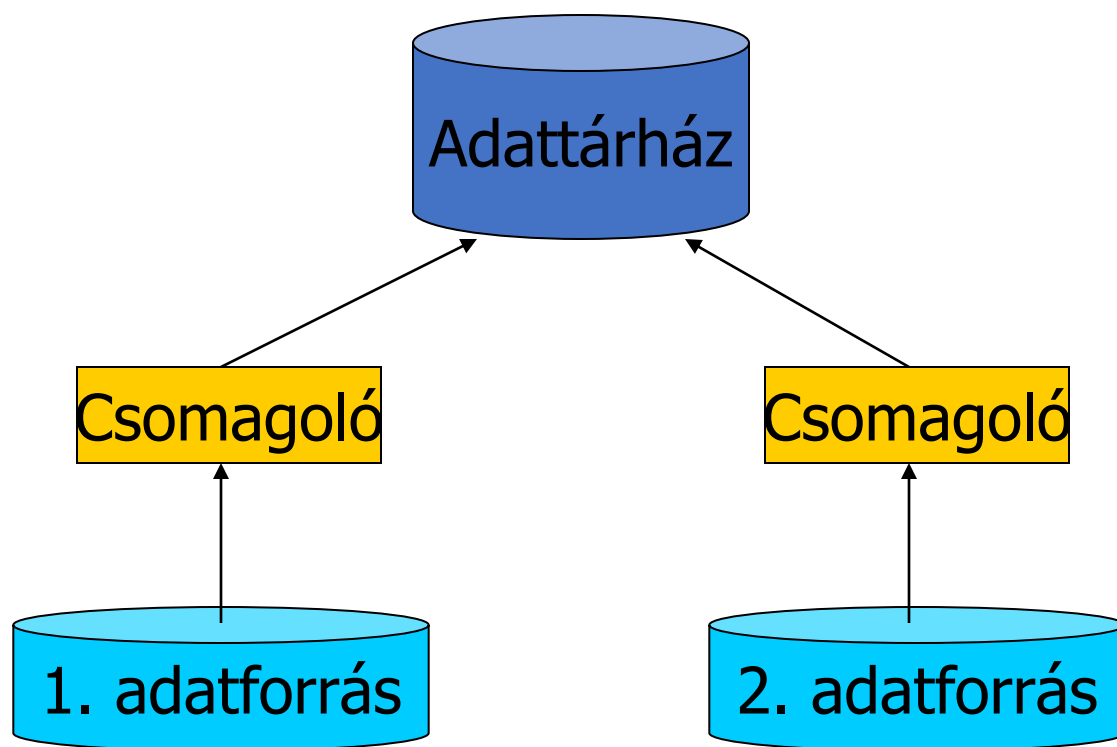
- Mindegyik teázónak van adatbázisa:
 - Az egyik relációs adatbáziskezelőt használ; másik MS-Word-ben tartja nyilván a menüt.
 - Az egyik nyilvántartja a mobil telefonok forgalmazóit, a másik nem.
 - Az egyik megkülönbözteti az angol zöld teát a többi teától, a másik nem.
 - Az egyik a leltárban a teákat dobozonként tartja nyilván, másik a göngyöleg egységei alapján.



Két megközelítés az integrációra

1. **Adattárház:** Az adatforrásokról egy központi másolatot készít, és egy közös adat sémává transzformálja.
 - Az adatokat naponta, hetente frissítik, de ennél nem szabad nagyobb pontosságot megcélozni.
2. **Mediáció, közvetítés:** Az összes adatforrásra egy nézetet kell létrehozni, mintha egy integrált rendszer részei volnának:
 - A nézetre vonatkozó lekérdezést úgy lehet megválaszolni, hogy a lekérdezést az egyes adatforrások szakkifejezéseire fordítják le és azután kérdezik le az eredeti adatforrásokat

Adattárház diagram



Közvetítő (*mediator*)

