Adatbázisok II.

5

Jánosi-Rancz Katalin Tünde
tsuto@ms.sapientia.ro
327A

Féligstrukturált adatok, XML, DTD, XSD

5

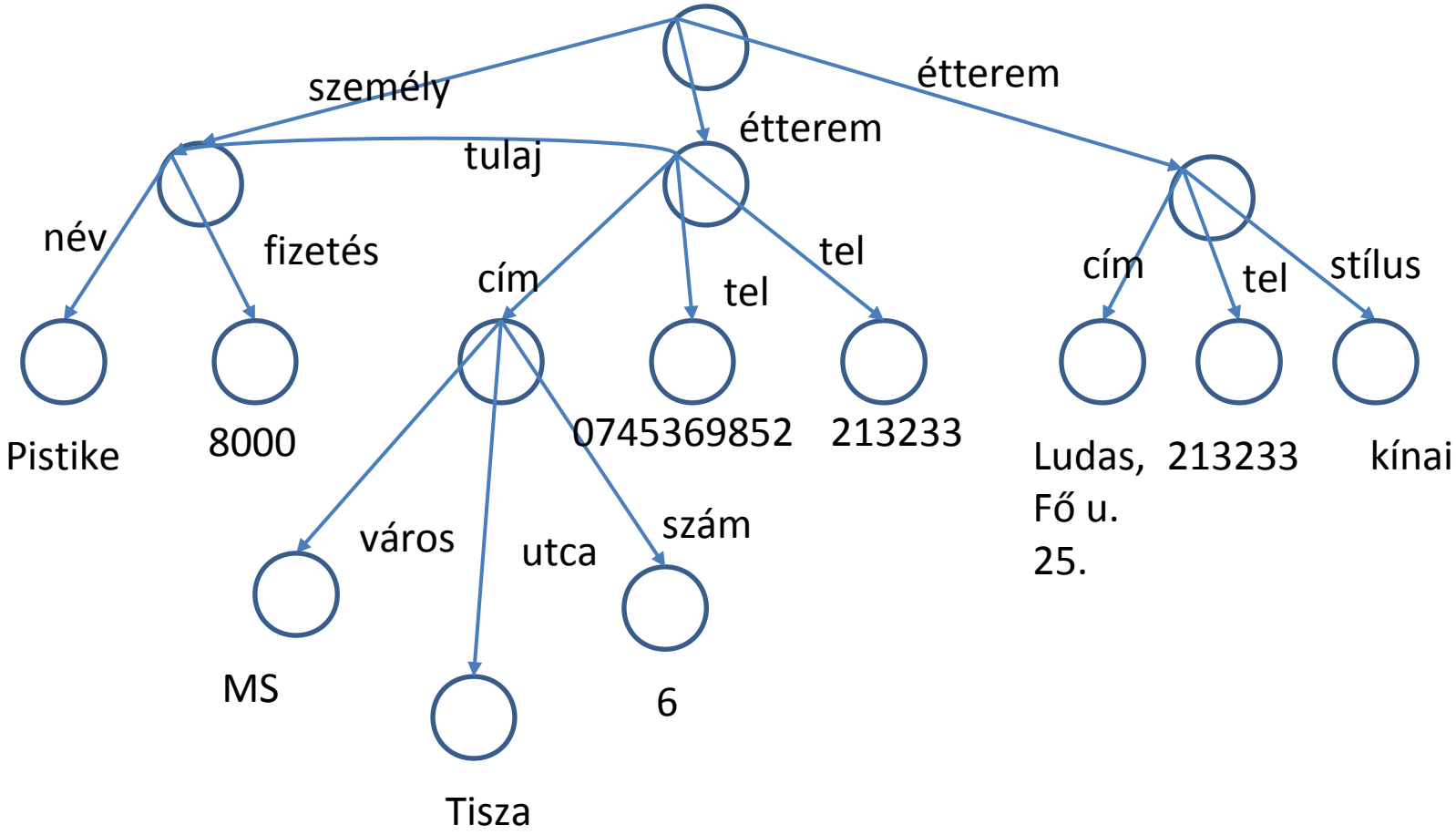
Adat séma nélkül

- Egy adatmodell általában három összetevőből áll:
  - séma: amivel az adatok típusa, szerkezete adható meg,
  - az engedélyezett műveletek,
  - megszorítások.
- A séma szerepe a relációs adatmodellben:
  - segít megőrizni az adatok konzisztenciáját,
  - könnyebben megszervezhető a tárolás,
  - gyorsítja a lekérdezések végrehajtását.
- Ugyanakkor a séma néha túlságosan is erős "megszorításokat" ír elő az adatok szerkezetére vonatkozóan.

Féligstrukturált adat

- A féligstrukturált adat önleíró, azaz nincsen séma.
- Ez nagyfokú rugalmasságot biztosít, természetesen ennek ára van.
- Adatbázisok integrációjánál használható például, amikor több adatbázis fölé egy közös adatbázist építenek.

Példa


személy
étterem
étterem
tulaj
név
fizetés
cím
tel
tel
cím
tel
stílus
Pistike
8000
0745369852
213233
Ludas,
Fő u.
25.
213233
kínai
város
utca
szám
MS
Tisza
6

Féligstrukturált adatok és W3C

- Az XML megjelenésével szinte egy időben kezdődött a féligstrukturált adatok kutatása, ami a korábbi relációs modell helyett gráfokat használ az adatok reprezentálásának modelljeként, s így nagyobb rugalmasságot biztosít.
- A két világ, a dokumentum- és az adatbázisvilág, hamar egymásra talált.
- A World Wide Web Consortium (W3C) (többek között a html kiötlője) szabványokat kezdett el kidolgozni az XML adatok egységes kezelésére vonatkozóan. A legfontosabb javaslataik:
  - XML, XMLDOM
  - XPath
  - XSLT
  - XQuery

XML- eXtensible Markup Language:

- Kiterjeszthető jelölőnyelv
  - Jelölő nyelv
  - Általában szöveges fájlban tároljuk
  - Címkékből (tag), attribútumokból és magából a tartalomból áll
  - Hierarchikus szerkezetű
- Megjelenését az világhálón történő adatcsere tette szükségessé.
- Az adatok sokszor relációs adatbázisból származnak.
- Kiegészíti a HTML-t.
  - tetszés szerint új elemek (tag) vezethetők be;
  - a struktúra bármilyen mélységig beágyazható;
  - tartalmazhat leírást a nyelvtanáról
- Kimondottan az adatok tartalmának leírására tervezték, és nem annak megjelenítésére
- Inkompatibilis rendszerek között is lehetővé teszi (vagy legalábbis megkönnyebbíti) a kommunikációt az interneten.

XML felhasználási területei

Struktúrált dokumentumszerkesztés, leírás: docx
Vektorgrafika tárolása: VML, SVG
Alkalmazások közötti webes kommunikáció: WSDL
Pénzügyi információk cseréje: OFX
Digitális őrlapok kezelése: XFDL, Infopath
Emberi erıforrás menedzsment jelölınyelv: HRMML
Jogi dokumentumok kezelése: OXCI
Matematikai képletek leírása, formázása: MathML
Konfigurációs adatok leírása: egyedi
News feeds leírás: RSS
3D grafikai leírás: X3D
Stílus leírásra megjelenítéshez: XSL
Adatbázis, lekérdezés: XQuery
HTML leírás: XHTML

Miért vagyunk mi AB-osok XML érdekeltek?

Adatbázis kérdések:
Hogy tudjuk mi modelálni az XML-t? (gráf)
Hogy tudjuk mi lekérdezni az XML-t? (xquery)
Hogy tudjuk mi az XML-t relaciós vagy OOAB-ban eltárolni?
Hogy tudjuk mi az XML-t hatékonyan feldolgozni?

XML (szintaxis)

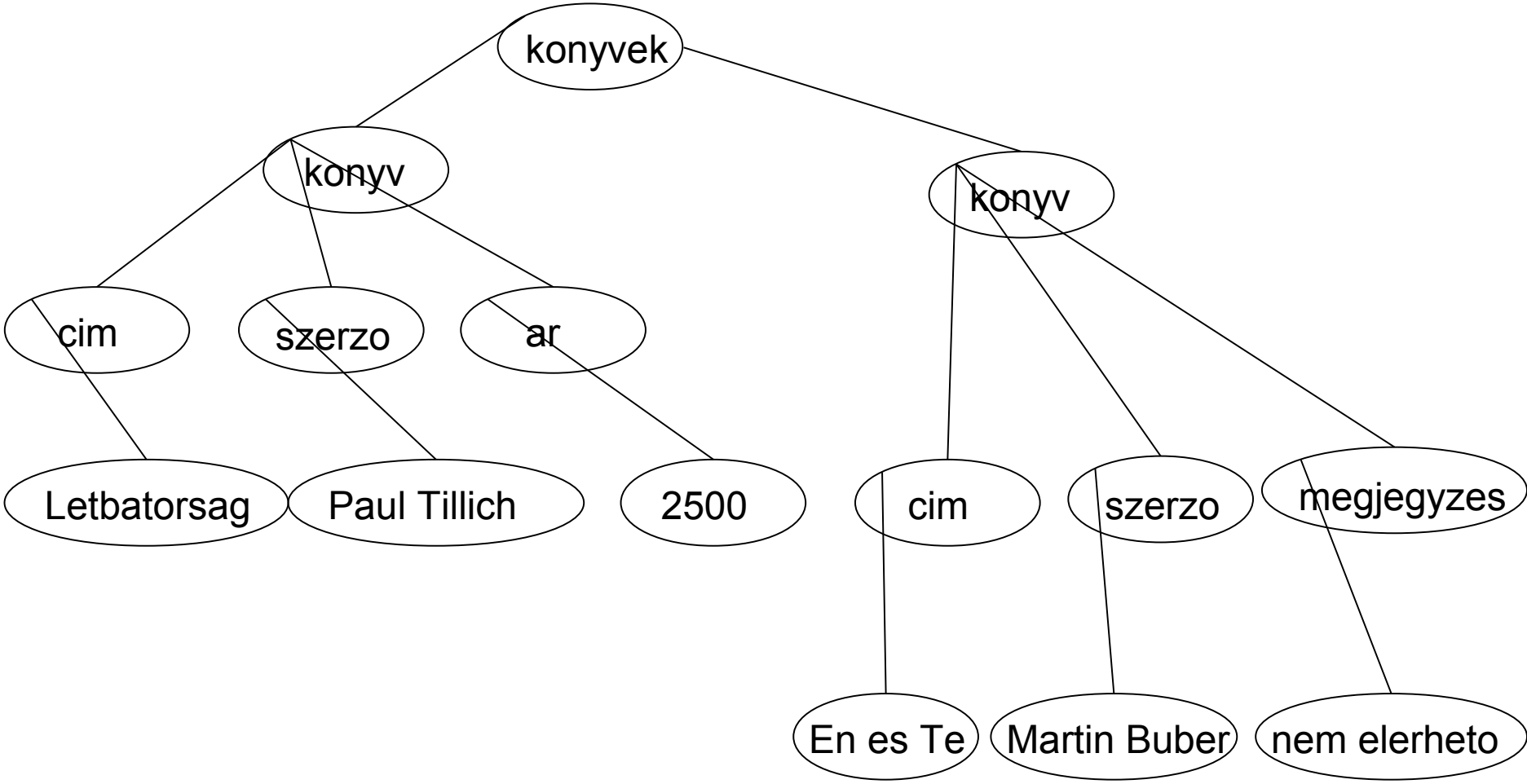
```
<?xml version="1.0" encoding="ISO-8859-2"?>
<!-- Kommentár -->
<szakdolgozat>
<oldal azon="0">
        <nev>MyTutor</nev>
        <lablec mi="oldalszam" />
</oldal>
</szakdolgozat>
```

- Sor1: XML deklaráció szerepel, mely megmutatja a használt XML verziót és a karakterkódolást.
- Sor2: egy megjegyzés szerepel.
- Sor3: a gyökérelem, a "<szakdolgozat>" található.
- Sor4: egy újabb elem következik, ahol már egy attributum is található, az "<azon>".
- Sor5: egy szöveget tartalmazó csomópont latható.
- Sor6:Ezt egy üres elem követi.
- Sor7: lezárjuk a "</oldal>"-lal az "oldal" csomópontot.
- Végül a gyökérelem lezárásával végződik a dokumentum.

Példa (XML dokumentum)

```
<? xml version=”1.0” ?>
<konyvek>
    <konyv nyelv=” magyar”>
        <cim>Létbátorság</cim>
        <szerzo>Paul Tillich</lang>
        <ar>2400</ar>
    </konyv>
    <konyv nyelv=”magyar” megjegyzes=”nem elerheto”>
        <cim>En es Te</cim>
        <szerzo>Martin Buber</szerzo>
     </konyv>
</konyvek>
```

Példa (gráf ábrázolás)



Tulajdonságai

- Kis és nagybetűk különbözőek.
- Egy vezérlési utasítással kell kezdődjön
- Kötelező megadni a záró tag-eket.
- Fontos az egymásba ágyazás, így a zárás sorrendje.
  - Vastag és dölt szöveg. HIBÁS!
  - Vastag és dölt szöveg. JÓ!
- Mindig van egy gyökérelem, annak lehet egy gyermeke, mely tartalmazhat további leszármazottakat
- Bővíthető
- Elem név nem tartalmazhat szóközöket (space). Nem kezdődhet: számmal és xml karakterekkel.
- Elem vagy attribútum? Nincs rá szabály!

```
<szemely nem="ffi">
    <nev>Ka Pál</nev>
</szemely>
<szemely>
    <nem>ffi</nem>
    <nev>Ka Pál</nev>
</szemely>
```

Névterek

- Névütközés
- Előfordulhat, hogy ugyanazt a tag-et több jelentésben szeretnénk használni. XML nem tesz megkötést az elemek neveire vonatkozóan
- Egy áru árának leírása:

```
<ar>
    <mennyi>1234</mennyi>
    <valuta>Ron</valuta>
</ar>
```

- Egy cipészszerszám (ár) adatai:

```
<ar>
    <hossz>1234</hossz>
    <mertek>mm</mertek>
</ar>
```

• **Megoldás 1**

Előnév használata:

```
<aru:ar>
    <aru:mennyi>1234</aru:mennyi>
    <aru:valuta>Ron</aru:valuta>
</aru:ar>
<c:ar>
    <c:hossz>1234</c:hossz>
    <c:mertek>mm</c:mertek>
</c:ar>
```

Megoldás névterek segítségével

- "xmlns" attributum értékeként
- 1. Az elemnév első megjelenésénél definiálni kell, hogy az egy névteret és hozzá kell rendelni egy egyedi névtér nevet

```
<aru:ar
    xmlns:aru="http://ms.sapientia.ro/tsuto/ns.htm">
    <aru:mennyi>1234</aru:mennyi>
    <aru:valuta>Ron</aru:valuta>
</aru:ar>
```

- 2. A gyökér-elemben: mely ekkor az összes nem minősített elem névtere lesz:

```
<gyoker xmlns="http://ms.sapientia.ro/tsuto/ns2.htm">
    … <ar>…</ar>
</gyoker>
```

- Az XML értelmező ezt az URL-t nem használja, csak egy egyedi névnek tekinti, de ez lehet egy hivatkozás az adott névtért illetően is

XML dokumentumok felépítése

- Elemek
- Attribútomok (jellemző)
- Egyedek (Entities)
  - Az XML dokumentumban az adatok nem különülnek el az elemnevektől így a feldolgozás során azokat is végigolvassa a rendszer.
  - XML feldolgozását megkönnyítő karakterek:
    - < < kisebb, mint (less than)
    - > > nagyobb, mint (greater than)
    - & & és (ampersand)
    - ' ' aposztróf (apostrophe)
    - " " macskaköröm (quotation mark)

- PCDATA - Karakterekbıl álló szöveg. A parser értelmezni fogja a tartalmát.
- CDATA - Olyan adathalmazok is, ahol nehéz lenne megoldani a karaktercseréket. Itt megadhatjuk, hogy az adott rész ne kerüljön feldolgozásra.

```
<![CDATA  [ez itt < nem feldolgozandó & karaktersorozat  ]]>
```

XML adatok sémával és séma nélkül I.

- A jólformált (well-formed) XML dokumentumok esetén nincs séma.
- Az érvényes (valid) dokumentumok nem csupán jólformáltak, de tartozik hozzájuk egy séma is.
- A séma megadja a megengedett tag-eket, nyelvtant ad a tag-ek egymásba ágyazásához, ellenőrzi a különböző forrásból származó XML dokumentumok azonos típusúak-e, felhasználhatóak-e ugyanabban az alkalmazásban. A sémát használjuk a dokumentum érvényességének a vizsgálatára
- A séma definiálására két nyelvet fejlesztett ki a W3C (The World Wide Web Consortium):
  - DTD (Document Type Definition),
  - XSD (XML Schema Definition).

DTD

- Egy eszköz, amivel az XML dokumentumok szerkezetét lehet megadni
- Egy XML dokumentumra vonatkozó szabálykészletet állít fel.
- Meghatározza a nyelvtant és az elem-készletet az adott XML-formázáshoz.
- A DTD szintaxisa kompaktabb, mint az XML szintaxis. A DTD definíció nem XML dokumentum.
- Az elemek esetén megadjuk, hogy mi lehet a tartalmuk.
- Emellett megadjuk, hogy milyen attribútumok tartozhatnak egy-egy elemhez.
- DTD az, ami az XML adatok hordozhatóságát biztosítja
- a DTD teszi lehetővé, hogy az XML állományt fogadó másik alkalmazás felismerje, hogyan kell a kapott állományt feldolgozni, és hogyan kell benne keresni.

Elemek deklarálása DTD-ben

```
<SZEMELYEK>
    <SZEMELY>
        <NEV>Szabó János</NEV>
        <TELEFON>413476</TELEFON >
        <EMAIL>jani76@hotmail.com</EMAIL>
    </SZEMELY>
</SZEMELYEK>
```

- Egy megfelelő DTD (!DOCTYPE, gyökér elem, gyermek elemek):

```
<!DOCTYPE SZEMELYEK [
    <!ELEMENT SZEMELYEK (SZEMELY*)>
        <!ELEMENT SZEMELY (NEV,TELEFON,EMAIL)>
        <!ELEMENT NEV (#PCDATA)>
        <!ELEMENT TELEFON (#PCDATA)>
        <!ELEMENT EMAIL (#PCDATA)>
        ] >
```

A DTD, mint nyelvtan

```
<?xml version="1.0"?>
<ab>
    <cim>
    <szemely>
        <cimzes>Dr.</cimzes>
        <csaladnev>Kelemen</csaladnev>
        <keresztnev>Imre</keresztnev>
    </szemely>
    <utca>Tavasz 12</utca>
    <varos>Kolozsvar</varos>
    <megye>Kolozs</megye>
    <iranyitoszam>400231</iranyitoszam>
    </cim>
    …
</ab>
```

Ennek a DTD-je a következő lenne:

- <!DOCTYPE ab [

  <!ELEMENT cim(szemely,utca,varos, megye,iranyitoszam)*>

  <!ELEMENT szemely(cimzes?, csaladnev,keresztnev+)>

  <!ELEMENT cimzes (#PCDATA)>

  <!ELEMENT csaladnev (#PCDATA)>

  <!ELEMENT keresztnev (#PCDATA)>

  <!ELEMENT utca (#PCDATA)>

  <!ELEMENT varos (#PCDATA)>

  <!ELEMENT megye (#PCDATA)>

  <!ELEMENT iranyitoszam (#PCDATA)>

]>

- DTD megköveteli, hogy a <szemely>, <utca>, <varos>, <megye>, <iranyitoszam> a megadott sorrendben jelenjenek meg a <cim> elemben.

Belső deklaráció

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cikk [
    <!ELEMENT cikk (datum, szerzo, tartalom)>
    <!ELEMENT datum (#PCDATA)>
    <!ELEMENT szerzo (#PCDATA)>
    <!ELEMENT tartalom (#PCDATA)>
 ]>
<cikk>
    <datum>2003/05/01</datum>
    <szerzo>LAci</szerzo>
    <tartalom>tananyag</tartalom>
</cikk>
```

Külső deklaráció

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE cikk SYSTEM "cikk2.dtd">
<cikk>
    <datum>2003/05/01</datum>
    <szerzo>LAci</szerzo>
    <tartalom>tananyag</tartalom>
</cikk>
```

- A külső DTD fájl:

```
<!ELEMENT cikk (datum, szerzo, tartalom)>
<!ELEMENT datum (#PCDATA)>
<!ELEMENT szerzo (#PCDATA)>
<!ELEMENT tartalom (#PCDATA)>
```

- Ez lehetővé teszi, hogy többfelhasználó is ugyanazt a közös sémát használja, megkönnyítve így az adatcserét.

DTD építőelemek

- **Elemek** (ELEMENT): XML milyen típusú elemeket tartalmazhat, ezek tartalma, sorrendje
- **Attribútumok**, jellemzők (ATTLIST): elemtípusokban használható jellemzők meghatározása. Adattípus, alapértelmezett érték
- **Egyedek** (ENTITY): olyan változók, amelyek vmilyen szöveggel, értékkel helyettesítik a változót, továbbá paraméter megadással.
- **Tag-ek**: Ezek jelölik az elemeket. A kezdő tag az elem nevéből áll, míg a záró tagben az elem nevét megelőzi egy per (/) jel.
- **PCDATA**: Az elemek értelmezése során azok tartalma is feldolgozásra kerül.
- **CDATA**: Külön kell megadni, hogy az elemek adatait ne értelmezze a rendszer.

Elemek megadása

- <!ELEMENT elemnév típus>
- vagy
- <!ELEMENT elemnév (az_elem_tartalma)>
- Példák:
- <!ELEMENT Nev (#PCDATA)>: csak karakteres adatokat (szöveget) tartalmazhat, gyermek elemet nem!
- <!ELEMENT Altalanos ANY>: bármit tartalmazhat
- <!ELEMENT br EMPTY>: üres elem.

Elemek megadása

- <!ELEMENT cikk (datum,szerzo,tartalom)>
- Elem leszármazottait is ebben a sorrendben kell megadni.
- <!ELEMENT konyv (cim)> Leszármazottjának (cim) kell szerepelnie, de mást nem tartalmazhat (kötelezı gyermek elem)
- <!ELEMENT konyv (cim+)> Leszármazottjának (cim) legalább egyszer kell szerepelnie, többször is lehet
- <!ELEMENT elemnév (leszármazott*)> Egy elem leszármazottjának nullaszor vagy többször való előfordulását a csillaggal jelöljük.
- <!ELEMENT konyv (peldany?)> Leszármazottja (peldany) maximum egyszer szerepelhet, ha szerepel.
- <!ELEMENT cikk ((szerzo|datum),tartalom)> szerzo vagy datum elemet tartalmazza, tartalom kötelező
- <!ELEMENT Hegy (Nev+, Magassag?, Allam)> Nev elemet egyszer vagy többször is tartalmazhatja, Magassag elhagyható, Allam kötelező

Szabályok és egy példa.

- – k* (tetszőleges számú előfordulás)
- – k+ (legalább egy előfordulás)
- – k? (egy vagy egy sem)
- – k | k’ (egyik a kettőből)
- – k,k’ (összetevés).

```
<!ELEMENT  FILM (Szereplo* | Cim | Rendezo)?>
    <!ELEMENT Szereplo (#PCDATA)>
    <!ELEMENT Cim (#PCDATA)>
    <!ELEMENT Rendezo (#PCDATA)>
```

Példák XML-ben:

```
    <FILM>
        <Szereplo>Wentworth Miller</Szereplo>
        <Szereplo>Dominic Purcell</Szereplo>
    </FILM>
    <FILM>
        <Cim>Prison break</Cim>
    </FILM>
    <FILM/>
```

Attribútumok

- Kiegészítő információkat tartalmaznak:
  - Társított jellemzők neveit,
  - Típusát,
  - Kötelezőségét
  - Nem kötelezőség esetén, hiányában a feldolgozó teendőit
- Megadása:
  - <!ATTLIST elemnév attributumnév attributumtípus alapértelmezett_érték>
- Példa:
- <!ATTLIST cikk datum CDATA "2003/05/01">
- <!ATTLIST szerzo CDATA #REQUIRED>
- XML-ben:
  - <cikk datum="2007/02/19" szerzo="xy"/>
  - <cikk szerzo="xy"/>

Attribútum típusok

Típus	Jelentés
CDATA	Nem ellenőrzött karakter sor
(en1|en2|...)	Az érték egy számozott lista valamely eleme lehet
ID	Az érték egy egyedi azonosító
IDREF	Az érték egy másik elem egyedi azonosítója
IDREFS	Más elemek egyedi azonosítóinak listája
NMTOKEN	Az érték egy érvényes XML név
NMTOKENS	Az érték érvényes XML nevek listája
ENTITY	Az érték egy egyed
ENTITIES	Az érték egyedek listája
NOTATION	Az érték egy megjegyzés neve
Xml:	Az érték előre definiált xml: érték

- Az alapértelmezett_érték lehet:
  - Érték
  - #REQUIRED - kötelező ez az attributum
  - #IMPLIED - nem feltétlenül tartalmazza ezt az attributumot
  - #FIXED érték - fix érték
    - Megadása: <!ATTLIST Film kategoria CDATA #FIXED "Akció">
      - Érvényes XML:
        - <Film>Prison break</Film>
        - <Film kategoria= "Akció">Prison break</Film>
      - Érvénytelen XML:
        - <Film kategoria="oktatófilm">Prison break</Film>

Egyedek - ENTITY

- Deklarációja után az egyedre, mint (konstans) paraméterre bárhol hivatkozni lehet.
- Célja az olvashatóság, egyszerőbb szerkeszthetőség
- <!ENTITY egyed_neve "ertek">
  - DTD:
    - <!ENTITY iro "LAci">
  - XML:
    - <szerzo>&iro;</szerzo>
- Beillesztése & és ; között
- Külső egyedre történő hivatkozás, URI/URL megadásával:
- <!ENTITY temak SYSTEM "http://xml.inf.elte.hu/temakorok.xml">
- Ahhoz, hogy bárhol hivatkozhassunk az egyedre:
- <!ENTITY % konyv_dekl SYSTEM "konyv.dtd">
- <!ENTITY % kontinens "(Europa|Ázsia|Afrika|Amerika">
- <!ELEMENT orszag (#PCDATA)>
- <!ATTLIST orszag kontinens %kontinens; "Ázsia" nyelv CDATA #REQIRED>

Példa

```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE cikkek [
        <!ELEMENT cikkek (cikk*)>
        <!ELEMENT cikk (szerzo, tartalom)>
        <!ELEMENT szerzo (#PCDATA)>
        <!ELEMENT tartalom (#PCDATA)>
        <!ATTLIST cikk datum CDATA "2003/05/01">
    ]>
    <cikkek>
        <cikk datum="2004/01/02">
            <szerzo>Deb Ella</szerzo>
            <tartalom>tananyag</tartalom>
        </cikk>
        <cikk>
            <szerzo>Ka Pál</szerzo>
            <tartalom>Növényvédelem</tartalom>
            </cikk>
    </cikkek>
```

XSD

Formázás és megjelenítés, CSS

CSS ismertető
P
{display:block;
margin-top:12pt;
font-size:10pt}

Környezetfüggő kiválasztás
XML:

```
<?xml-stylesheet type="text/css" href="1.css"?>
<TERKEPEK>
      <VAROS>
            <NEV>Szeged</NEV>
            <MEGYE>Csongrád</MEGYE>
      </VAROS>
            <MEGYE>Pest</MEGYE>
</TERKEPEK>
```

CSS:

```
VAROS MEGYE{font-style:normal}
MEGYE{font-style:italic}
```

Adatkötés

XML dokumentum hozzákapcsolása a HTML dokumentumhoz

```
<XML ID="LeltarDSO" src="Leltar.xml"></XML>
```

HTML elemekhez XML jellemzők rendelése

```
<SPAN DATASRC="#LeltarDSO" DATEFLD="SZERZO"></SPAN>
```

XML és HTML összekapcsolása

Adatsziget a HTML BODY elemében:

```
<BODY>
<XML ID="LeltarDSO">
<?xml version="1.0" encoding="ISO-8859-2"?>
...
</XML>
</BODY>
```

Külső dokumentumban:

```
<HTML>
<BODY>
<XML ID="LeltarDSO" SRC="Leltar.xml">
<!– HTML folytatása --!>
</BODY>
</HTML>
```

HTML és XML összerendelése

Táblázatos adatötéssel (HTML)
Egyrekordos adatkötéssel (SPAN)

Táblázatos adatkötés használata
TABLE egyéb paraméterei
DATAPAGESIZE: méret
ID: elnevezés
DATSRC adatforrás
DATAFLD: melyik rekordhoz rendeljük hozzá
TABLE elem metódusok (ONCLICK)
firstPage
previousPage
nextPage
lastPage

```
<BODY>
    <XML ID="LeltarDSO" SRC="Leltar.xml"></XML>
    <H2>Könyvleltár</H2>
    <TABLE DATASRC="#LeltarDSO" BORDER="1" CELLPADDING="5">
        <THEAD>
            <TH>Cím</TH>
            <TH>Szerzı</TH>
            <TH>Kötés</TH>
            <TH>Oldalszám</TH>
            <TH>Ár</TH>
        </THEAD>
        <TR ALIGN="center">
            <TD><SPAN DATAFLD="CIM" STYLE="font-style:italic"></SPAN></TD>
            <TD><SPAN DATAFLD="SZERZO"></SPAN></TD>
            <TD><SPAN DATAFLD="KOTES"></SPAN></TD>
            <TD><SPAN DATAFLD="OLDALSZAM"></SPAN></TD>
            <TD><SPAN DATAFLD="AR"></SPAN></TD>
        </TR>
    </TABLE>
</BODY>
```

Egyrekordos adatkötéssel (SPAN)

HTML elem: SPAN, BUTTON, LABEL, IMG
Paraméterek:
DATASRC
DATAFLD
Rekordok közötti mozgás
moveFirst
movePrevious
moveNext
moveLast
move (pl.: LeltarDSO.recordset.move(5)) adott pozícióra ugrik

Oracle XML DB

- Az XML adatok kezelésére az Oracle egy külön komponenst készített, az Oracle XML DB-t.
- Az XML adatok tárolására az XMLType típust fejlesztették ki.
- Az XMLType típus tulajdonképpen egy objektumtípus. Jellemzői:
  - tábla és tábla oszlopa is lehet XMLType típusú,
  - ugyanúgy használható, mint bármelyik másik típus, pl. szerepelhet PL/SQL eljárás paramétereként, függvény visszatérési értékeként stb.,
  - csak jól formázott (well-formed) XML dokumentumok lehetnek ilyen típusúak,
  - legfontosabb metódusai: extract(), extractValue(), existsNode(), xmlSequence(), updateXML(), ezek a függvények azonban önállóan is léteznek.
- Alapesetben az XML dokumentumok CLOB-ként (Character Large Object) tárolódnak.

extract(), extractValue(), existsNode()

- Az extract() azt a pontot vagy pontokat adja vissza, amelyek illeszkednek a függvényben megadott XPath kifejezésre.
- A függvény az XMLType egy metódusaként is használható.
- Az existsNode() függően szolgáltat igaz vagy hamis értéket (1 vagy 0), hogy a megadott XPath kifejezésre illeszkedik pont vagy sem.
- Segítségével olyan feltételeket fogalmazhatunk meg többek között a WHERE feltételben, melyeknek kifejezése különben bonyodalmas lenne.
- Az extractValue() annak az attribútumnak, elemnek a szöveges értékét szolgáltatja vissza, ami illeszkedik a függvényben megadott XPath kifejezésre.
- Példákhoz lásd az XML_XPath_pelda.html fájt.

XMLSequence()

- Az extract() függvény sok esetben dokumentum helyett dokumentum-töredékeket (document fragments) ad vissza, vagyis olyan XML elemeket, amelyeket „nem fog össze” egy közös gyökér, hanem függetlenek egymástól.
- Az XMLSequence() függvény minden egyes ilyen fragmentet XMLType típusú objektummá alakít, majd veszi ezek kollekcióját.
- A table() függvénnyel aztán a kollekció virtuális táblává alakítható.
- Példákhoz lásd az XML_XPath_pelda.html fájlt.

Feladatok I.

- A kolcsonzesek tábla felett adjuk meg a következő lekérdezéseket.
  - Adjuk meg a gyökérelemet és annak tartalmát.
  - Adjuk meg a táblában szereplő neveket.
  - Adjuk meg az összes attribútumot és azok értékét.
  - Adjuk meg a Gipsz Jakab által kölcsönzött könyvek címeit.
  - Adjuk meg a második kölcsönző által kölcsönzött cd-k közül az elsőt.
  - Adjuk meg azokat, akik kölcsönöztek dvd-t, de nem kölcsönöztek könyvet.
  - Adjuk meg azon könyveket melyeknek a nevében szerepel az ‘s’ betű (kicsi vagy nagy).
  - Adjuk meg azokat, a könyveket, amelyeket legalább ketten kölcsönöztek.
  - Adjuk meg hány gyermeke van a kolcsonzesek elemnek.

Feladatok II.

- A levelezes tábla felett adjuk meg a következő lekérdezéseket.
  - Adjuk meg Melák kiknek küldött email-t.
  - Adjuk meg, hogy Luca összesen hány email-t küldött.
  - Adjuk meg ki írt olyan email-t, amelyben szerepel a Melak szó.
  - Adjuk meg Luca válaszként küldött email-jeinek (subject RE:-vel kezdődik) szövegét.
  - Adjuk meg azon email-ek szövegét, amelyekre választ is küldtek. A válaszok szövegét is adjuk meg.