

XMLSchema szabvány

Az előadás anyaga

Prof. Dr. Kovács László: Adatkezelés XML környezetbe
jegyzete alapján készült el

XMLSchema szabvány

Az előadás anyaga

Kovács László: Adatkezelés XML környezetbe
jegyzete alapján készült el

Témakör kérdései

1. XMLSchema jellemzése
2. XMLSchema szerkezete és típusai
3. *Elemi jelölő elemek megadása*
4. *Összetett jelölő elemek megadása*
5. *Attribútum megadása*
6. *Kulcs integritási elem*
7. *Hivatkozási kulcs elem*
8. ER modell konverziója XMLSchema-ra

Igényelt kompetenciák

- XMLSchema szerkezete és típusai
- noname típusokkal dolgozó XMLSchema készítése
- ER/relációs séma konverzója XMLSchema-ra
- Környezet: XML szerkesztő (Oxygen, EditIX, Eclypse,)

DTD korlátai

Fő probléma: nem ismeri a különböző *adattípusok kezelését*,

- igen szűk a támogatott *integritási feltételek* köre,
- túl laza az *ID* és az *IDREF mechanizmus*,
- nem alkalmas a *moduláris sémafejlesztésre* (*nincs öröklés*),
- nem tudja kezelní a *névtereket*,
- igen egyszerű a *struktúra kezelés*,
- nincs *felhasználói adattípus*.

Séma típusok

Az említett DTD korlatok miatt le kellett cserélni egy jobban „xmlesített” sémaleírásra.

Többféle javaslat is napvilágot látott, név szerint a legfontosabbak: RELAX, TREX, Schematron, SOX, DSD, XDR és **XML Schema**.

Sémák lehetnek:

- *szabályalapú*: szabályellenőrzésében fejlettebbek (validálás)
- *nyelvtani elemekből* építkezők: ezek jobbak struktúra definiálásra.

Séma típusok

A *nyelvtanalapú* sémákból *objektumokat* lehet generálni: *osztályokat, interfészeket.*

Ez is oka annak, hogy a *World Wide Web* konzorciumnál az *XML Schema Definition* (XSD) győzött, mint XML séma ajánlás.

Ez nem jelenti azt, hogy a többi kihal, valószínűleg hosszabb távon a *szabályalapúak közül* is ki fog fejlődni egy jól használható leírás.

Séma szerepe

„A séma szerepe, hogy korlátozza a tárolt előfordulások, értékek és elvégezhető műveletek körét.

Ellenőrzi az előfordulás helyességét, egyszerűsíti az előállítását.

Előnye: az egyértelműség és egyszerűség.

Az XML sémakezelése opcionális, laza a kapcsolat a séma és az XML dokumentum között.

Sémaleírás tartalma

A sémaleírásnak tartalmaznia kell a következőket:

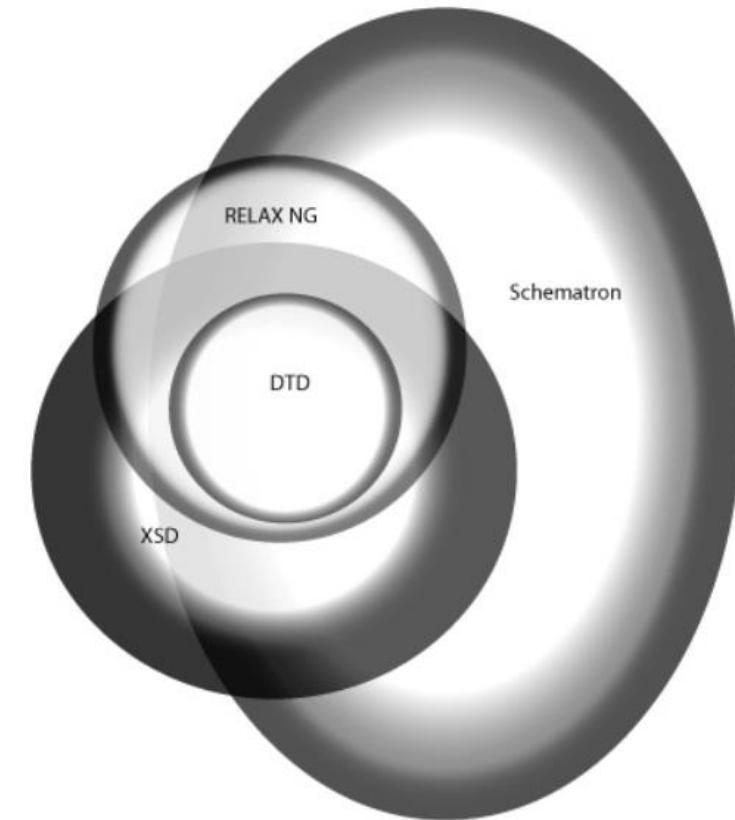
- *Elemek és belső szerkezetük megadása, jellemzőik meghatározása.*
- *Elemhierarchia.*
- *Elemek és attribútumok számososságának, értékének korlátozása.*
- *Érték egyediség, hivatkozási integritás ellenőrzése.*
- *Szimbólumok definiálása.*

XML sémanyelv

Az XML 1.0 részeként definiált *DTD* egy primitív XML séma nyelv.

Korszerű XML séma nyelvek:

- **W3C XML Schema**
<http://www.w3.org/XML/Schema>
- RELAX NG (ISO/IEC 19757-2:2003)
<http://www.relaxng.org/>
- Schematron (ISO/IEC 19757-3:2006)
<http://www.schematron.com/>



Forrás: schematron.com

XSD (XML Schema Definition) - jellemzése

Az **XSD (XML Schema Definition)**, vagy **XMLSchema**, egy a W3C ajánlásával publikált XML sémaleírónyelv.

- A sémaleírás a *XML szintaktikát követi*, és a séma egy *helyesen formált XML dokumentum*.
- A DTD-vel ellentétben az *XMLSchema sémaleírás mindig külön fájlban helyezkedik el*.

XSD (XML Schema Definition) - jellemzése

- Használhatók benne *gyári* és *egyedileg definiált típusok* is.
- Nevesített típusokat és a köztük lévő *specializációs kapcsolatokat*, ill. *absztrakt típusokat* is képes kezelní.
- Támogatja a *névterek használatát*, az alapelemek névtere szerkezetleíró és *integritási elemeket* tartalmaz.

XMLSchema - jellemzése

XMLSchema kialakulása:

- XML-Data javaslat: 1998 (Microsoft)
- SOX (Schema for Object-Oriented XML): 1998 Veo
- Schematron: 1999
- W3C XMLSchema 1.0 ajánlás: 2001.
- W3C XMLSchema 1.1 ajánlás: 2012.

W3C XML Schema

- Mivel vannak más *XML séma nyelvek* is, az egyértelműség végett használják az **XSD** elnevezést is.
 - A W3C XML Schema 1.1 verziója már hivatalosan is az XSD nevet viseli – 2012.

W3C XML Schema Definition Language

- A legfrissebb verzió az 1.1 számú:
 - *W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures* (W3C ajánlás, 2012. április 5.)
<http://www.w3.org/TR/xmlschema11-1/>
 - *W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes* (W3C ajánlás, 2012. április 5.)
<http://www.w3.org/TR/xmlschema11-2/>
- Visszafelé kompatibilis az 1.0 számú verzióval.
- Sok újdonságot tartalmaz.

W3C XML Schema Definition Language

- Jelenleg még nem elterjedt, néhány teljes implementáció létezik:

- *Saxon-EE* <http://saxonica.com/>

- Nem szabad szoftver.

- *Apache Xerces2 Java*
<http://xerces.apache.org/xerces2-j/>

Az *Apache License v2* hatálya alatt elérhető szabad és nyílt forrás szoftver.

Példák sémákra

- Apache Maven: <https://maven.apache.org/xsd/maven-4.0.0.xsd>
- Java EE: <http://xmlns.jcp.org/xml/ns/javaee/>
- Java Persistence API (JPA):
<http://xmlns.jcp.org/xml/ns/persistence/>
- KML: <http://schemas.opengis.net/kml/>

XMLSchema - jellemzése

- XML szintaktika,
- *moduláris szerkezet*,
- *típusok használata*,
- *gazdag gyári adattípusok*,
- *nevesített típusok*,
- *absztrakt típusok kezelése*,
- *kulcs, idegenkulcs megkötések*,
- *gazdag struktúra elemek*,
- *névterek támogatása*.

```
<xs:element name=
  type="kl:dolg_t">
  <xs:key name=...>
  </xs:key>
</xs:element>
```

XMLSchema - jellemzése

- XML szintaxis használata.
- *Önleíró* (létezik a sémához séma).
- *Primitív adattípusok* biztosítása (pl.: SQL és Java típusok).
- *Felhasználói típusok* létrehozását támogató típus rendszer.
- *XML névérték* támogatása.
- Alkalmazások széles köre által használható.
- *Bonyolult és kényelmetlen szintaxis*.

XML dokumentum – DTD - XMLSchema

Hol tartunk az XML tanulmányokban?

Készítettünk egy egyszerű XML dokumentumot note.xml

```
<!DOCTYPE note SYSTEM „note.dtd”>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!DOCTYPE jegyzet SYSTEM "note.dtd">
4
5 <jegyzet>
6   <to>Péter</to>
7   <from>Jani</from>
8   <heading>Emlékeztető</heading>
9   <body>Ne felejtsd el, amit megbeszéltünk!</body>
10 </jegyzet>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!ELEMENT jegyzet (to,from,heading,body)>
4 <!ELEMENT to (#PCDATA)>
5 <!ELEMENT from (#PCDATA)>
6 <!ELEMENT heading (#PCDATA)>
7 <!ELEMENT body (#PCDATA)>
~
```

XML dokumentum – DTD - XMLSchema

Majd készítettünk egy note.dtd fájlt, amely meghatározza az XML dokumentum el

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3 <!ELEMENT jegyzet (to,from,heading,body)>
4 <!ELEMENT to (#PCDATA)>
5 <!ELEMENT from (#PCDATA)>
6 <!ELEMENT heading (#PCDATA)>
7 <!ELEMENT body (#PCDATA)>
```

A 3 sor meghatározza , hogy jegyzet elemnek négy gyermeké van.

A 4-7 sor pedig, hogy a to, from, title, body elements típusa (#PCDATA) – ezzel validálja a dokumentumot.

XML dokumentum – DTD - XMLSchema

Következik az XML Schema note.xsd nevű XML-séma fájl, amely meghatározza a fenti XML-dokumentum elemeit - note.xml.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
30 <jegyzet
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:noNamespaceSchemaLocation="note.xsd">
6
7     <to>Péter</to>
8     <from>Jani</from>
9     <heading>Emlékeztető</heading>
10    <body>Ne felejtsd el, amit megbeszéltünk!</body>
11
12 </jegyzet>
```

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
30 <xss:schema
4   xmlns:xs="http://www.w3.org/2001/XMLSchema"
5   elementFormDefault="qualified">
6
70 <xss:element name="jegyzet">
80   <xss:complexType>
90     <xss:sequence>
10    <xss:element name="to" type="xs:string"/>
11    <xss:element name="from" type="xs:string"/>
12    <xss:element name="heading" type="xs:string"/>
13    <xss:element name="body" type="xs:string"/>
14  </xss:sequence>
15 </xss:complexType>
16 </xss:element>
17
18 </xss:schema>
```

XML dokumentum – DTD - XMLSchema

A jegyzet element **Complex type**, mert más elemeket (gyerek) tartalmaz, ill. amikor mi adjuk meg, hogy mit tartalmazhat.

A to, from, title, body **Simple type**, mivel nem tartalmaznak más elemeket, ill. amikor az XSD a típusa.

XML dokumentum – XMLSchema

Ez az *XML-dokumentum hivatkozik* egy XMLSchema-ra:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2
3⊖ <jegyzet
4   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5   xsi:noNamespaceSchemaLocation="note.xsd"
6
7     <to>Péter</to>
8     <from>Jani</from>
9     <heading>Emlékeztető</heading>
10    <body>Ne felejtsd el, amit megbeszéltünk!</body>
11
12 </jegyzet>
```

XMLSchema alapfogalmak

- „Sémakomponens, séma, sémadokumentum.
- Deklaráció, definíció.
- Típusdefiníció.
- Egyszerű és komplex típus.
- Típushierarchia.

Sémák és sémadokumentumok

- A sémák **sémakomponensekből** állnak, melyek nem más mint *absztrakt objektumok*.
- *Sémadokumentum*: séma *XML reprezentációja*.
 - Az XML szyntaxt a szabvány határozza meg.

Sémakomponensek

- Elsődleges sémakomponensek:
 - elemdeklarációk
 - attribútum-deklarációk
 - típusdefiníciók
- Segédkomponensek:
 - kommentárok
 - modellcsoportok
 - részecskék
 - helyettesítők
- Másodlagos sémakomponensek:
 - modellcsoport-definíciók
 - attribútumcsoport-definíciók
 - azonossági megszorítások definíciói
 - jelölésdeklarációk

Sémák ábrázolása - Mit tartalmazhat a schema?

A schema elem tartalmazhatja *gyermekként tetszőleges számban* és *sorrendben*:

- minden *elsődleges sémakomponenst*,
- a *másodlagos sémakomponensek* közül a *modell-csoportokat, attribútum-csoportokat* és *jelölésdeklarációkat*,
- a *segédkomponensek* közül a *kommentárokat*.

A targetNamespace attribútum *opcionális*, a séma cél-névteret *szolgáltatja*.

XMLSchema létrehozása

A gyökérelem és a sémakomponenseket ábrázoló elemek neve: a <http://www.w3.org/2001/XMLSchema> URI-val azonosított névtérbe tartoznak.

- A **névtér-nevet**: az xs vagy az xsd előtaghoz szokás hozzákötni egy névtér deklarációban.

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

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

```
...
```

```
</xss: schema>
```

XML Schema létrehozása

A **schema** elemnek a *legfontosabb attribútuma*:

- az elementFormDefault, ami megadja, hogy az elemeknek minősítettnek (qualified) kell-e lenni vagy nem.
- az attributeFormDefault hasonlóan igaz ez az attribútumokra is .
 - unqualified = csak a gyökérnél él a névtér,
 - qualified = a gyökér alatti elemknél is él a névtér.

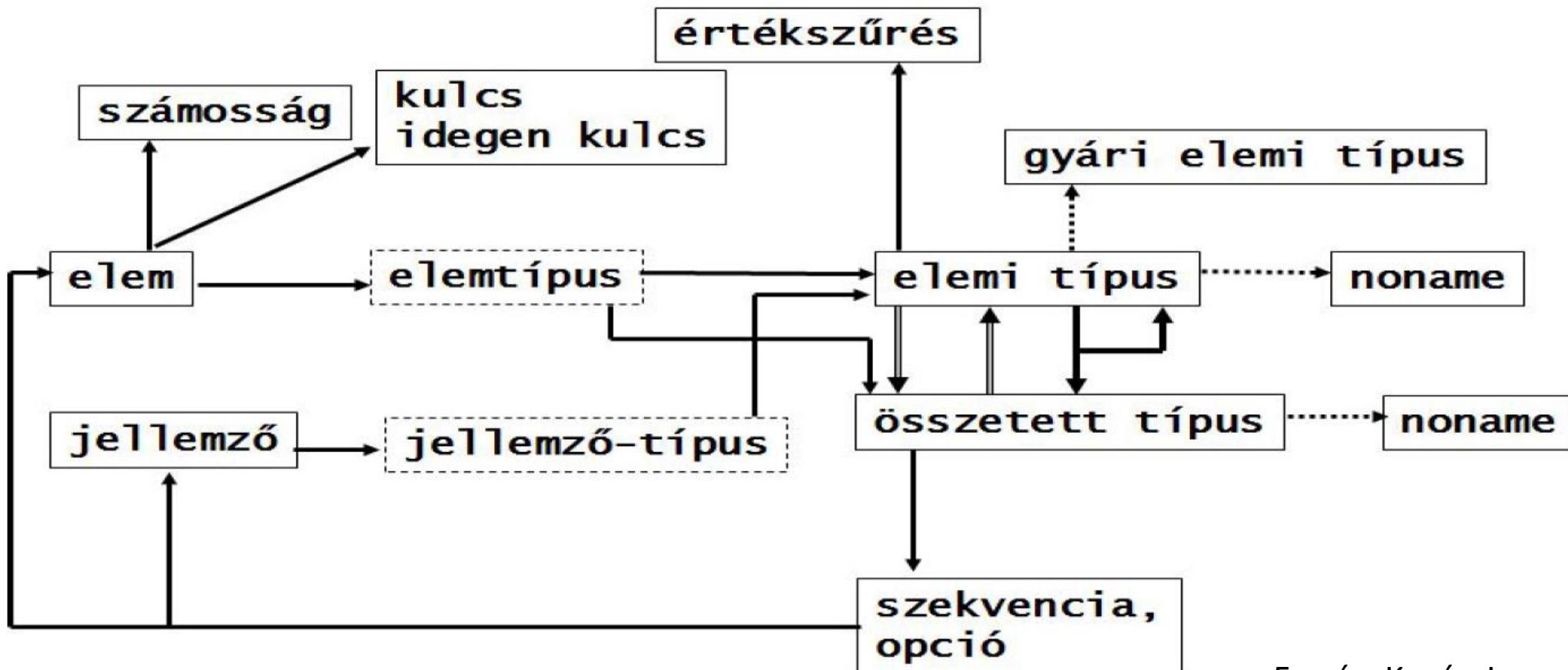
Jelölő elemek

Egy XML elemet az XSD-ben az *element* segítségével adhatunk meg, ahol meg kell adni a *nevét* és azt, hogy *mi lehet a tartalma*.

Az *elemek lehetnek*:

- Egyszerű (elemi) jelölő elem,
- Komplex (összetett) jelölő elem.

XMLSchema elemei - XMLSchema logikája



Forrás: KovácsL

Egyszerű (elemi) jelölő elem

Egyszerű elemet *simpleType* segítségével definiálhatjuk.

- Nem tartalmaz gyermekelemeket.
- Az elemekhez számos *beépített egyszerű típus* áll rendelkezésre (boolean, date, double, int etc).
- Attribútumok és elemek típusaként is használhatók.

Az XML-séma által támogatott **egyszerű típusok** teljes lista:

<https://www.w3.org/TR/xmlschema-0/#CreatDt>

Egyszerű jelölő elem - példa

Példa:

```
<xs:element name="kredit" type="xs:integer">  
<xs:element name="oktato" type="xs:string"/>
```

A *kredit* és *oktató* egyszerű jelölő elem, mert nem tartalmaz gyerek elemet, ill. XSD a típusa.

Komplex jelölő elem

Komplex elemet *complexType* segítségével definiálhatunk:

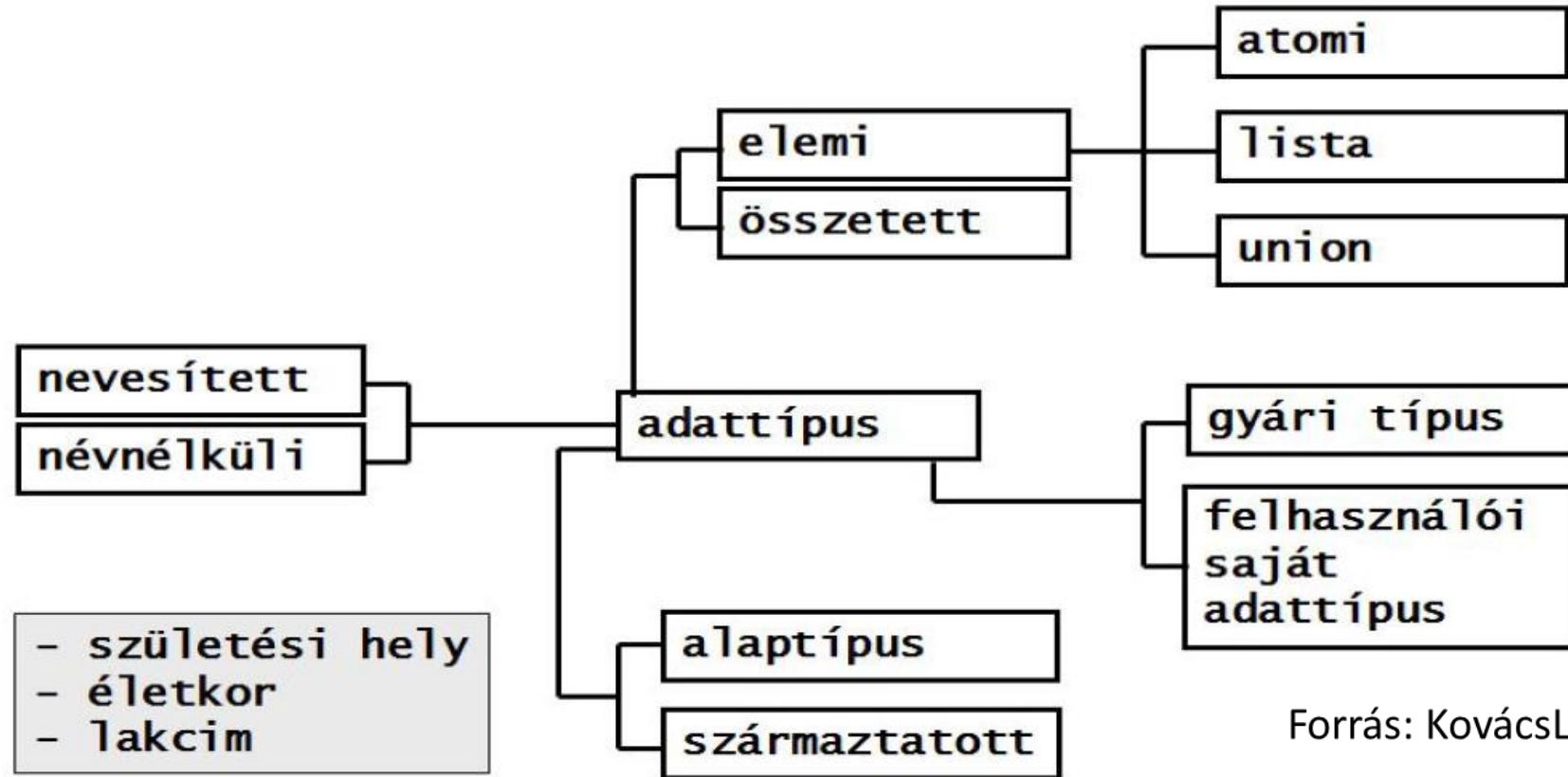
- milyen *elemeket* tartalmazzon,
- melyikből *mennyit tartalmazzon*,
- és hogy *számít-e a sorrendjük van sem.*

Lehetővé teszik *tartalomként* elemek, valamint *attribútumok* használatát.

Nincsenek beépített komplex típusok.

XMLSchema típusrendszere – előre-tekintés

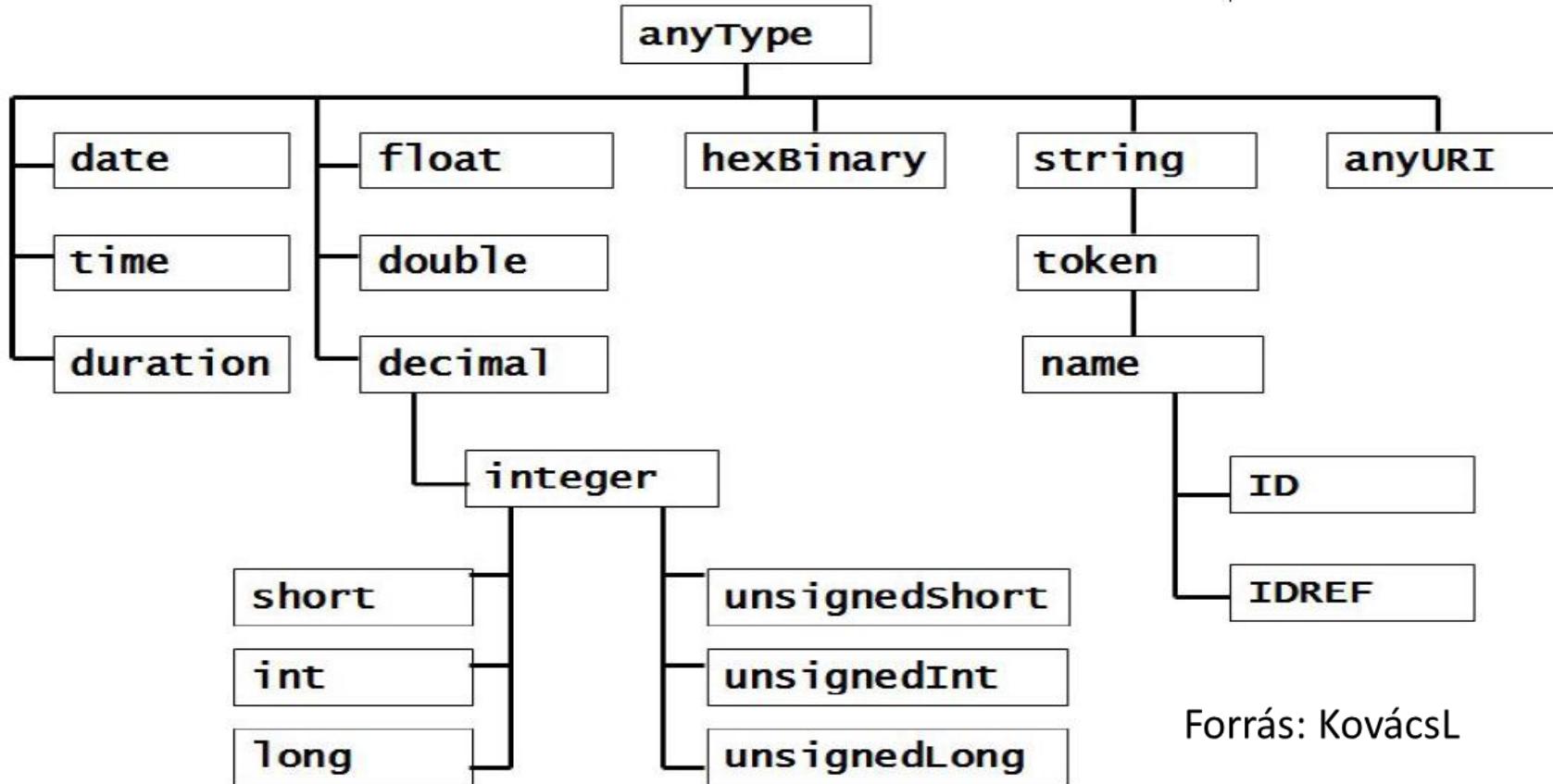
XMLSchema típusrendszere (egyszerűsítve)



Forrás: KovácsL

XMLSchema szerkezetek és típusok – előre tekintés

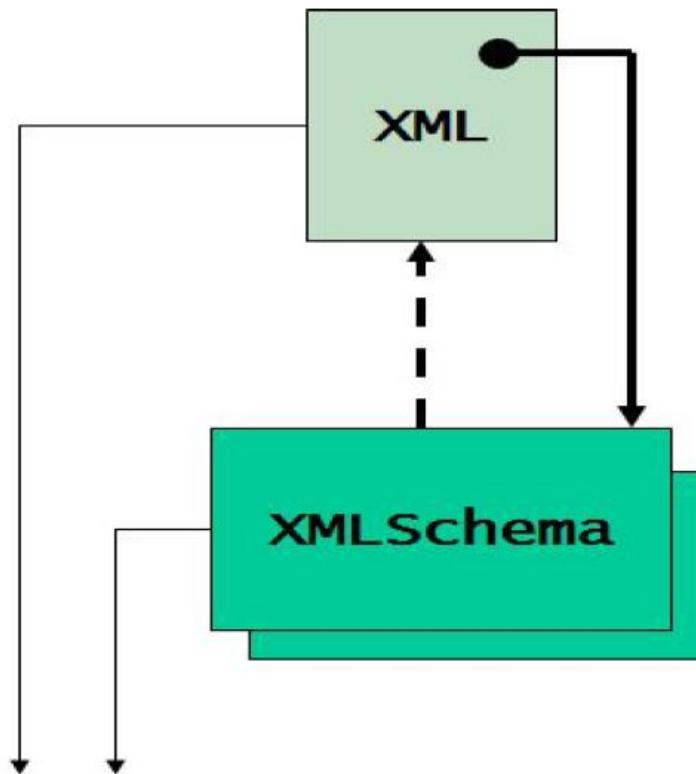
XMLSchema *gyári típusok* (beépített, egyszerűsítve)



Forrás: KovácsL

Az XMLSchema elemei

XMLSchema működési logika



```
<gyoker xmlns:xsi=
  "http://www.w3.org/2001/XMLSchema-instance"
  xsi:SchemaLocation="file:/d:..">
  ...
</gyoker>
```

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

XMLSchema típusai – tervezési lehetőség

XMLSchema: Helyesen formált XML dokumentum

- Elemeinek névtere: *http://www.w3.org/2001/XMLSchema*

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    -- alapértelmezési névterek --- >

    <!-- külső források bevonása -->

    <!-- globális elemek megadása (gyökérelemek) -->

    <!-- összetett típusok megadása -->

    <!-- elemi típusok megadása -->

</xs:schema>
```

Az XMLSchema elemei – globális elemek

- A *globális elemek* az XML dokumentumban *gyökérelemként szerepelhetnek*.
- Az XMLSchema szabványban a séma több *globális elemet* is tartalmazhat.
- A tervezőnek célszerű ezt leszűkíteni a ténylegesen használt *gyökérelemekre*.

Elemi jelölő elemek megadása

Egy *elem definiálása* az **ELEMENT** elemmel történik.

Jelölő elem megadása:

```
<xs:element jellemző > tartalom </xs:element>
```

Attribútumok megadása:

- *name*: elem azonosító neve (sztring)
- *type*: elem típusa (sztring) - opcionális
- *ref*: létező globális elemmel való egyezőséget fejezi ki (elemnév)

Elemi jelölő elemek megadása

- *abstract*: absztrakt elem jelölése (true vagy false)
- *form*: szükséges-e a dokumentumban névtér jelölése (qualified vagy unqualified)
- *substitutionGroup*: elemcsoport megadása

Az elemnél a *type* típuskijelölés is opcionális .

Ha szerepel, akkor megadhatjuk a *elemtípus azonosítóját*.

Ha nem, akkor egy un. *névnélküli típust* is ki lehet jelölni.

Elemi jelölő elemek megadása

Példában - a hangsúly a névvel ellátott típuskezelésen van:

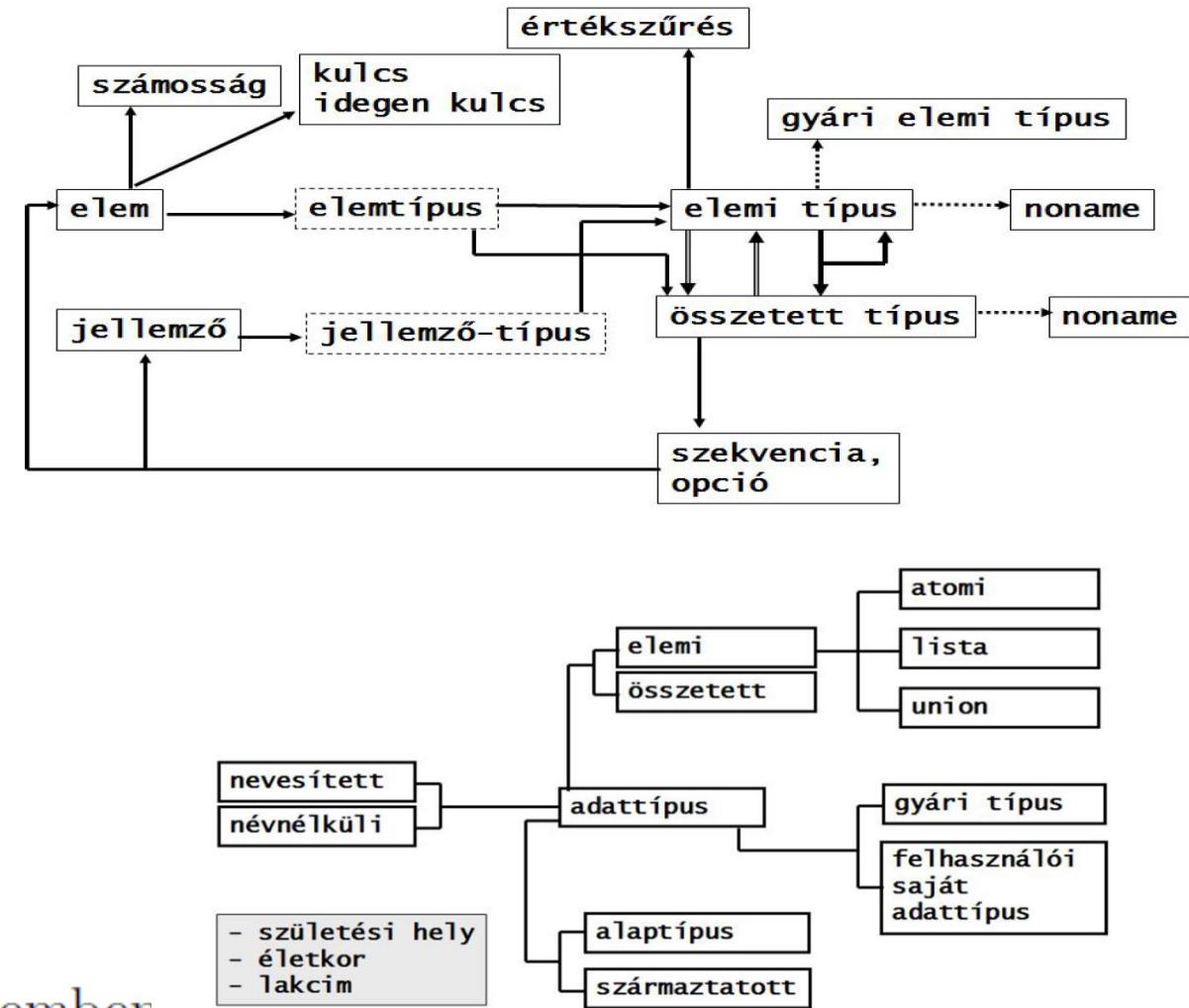
- a) <xs:element name="nev" type="xs:string">..</xs:element>

- b) <xs:element name="ember" type="emberTípus">.</xs:element>

A típus lehet gyári alaptípus és saját objektumtípus is.

Elemi jelölő elemek – gyári típusok

- string: szöveges
- integer: egész
- decimal: tizedes tört
- double: duppla pontosságú tört
- boolean: logikai
- date: dátum
- time: idő
- ID: azonosító DTD értelemben
- IDREF: azonosító hivatkozás DTD értelemben



Forrás: KovácsL

Adattípusok fajtái

- „*Primitív* és *származtatott* adattípusok.
- *Beépített* és *felhasználói származtatott* adattípusok.
- *Atomi, lista* és *unió* adattípusok.

Adattípusok fajtái

- *Primitív adattípusok*

A definiálásához nem kerülnek felhasználásra más adattípusok (pl.: [boolean](#), [double](#) és [string](#)).

- *Származtatott adattípusok*

Meglévő adattípusok felhasználásával (megszorítással, listavagy unióképzéssel) történik (pl.: decimal - integer, ill. string - [normalizedString](#) típust származtatnak.).

Adattípusok fajtái

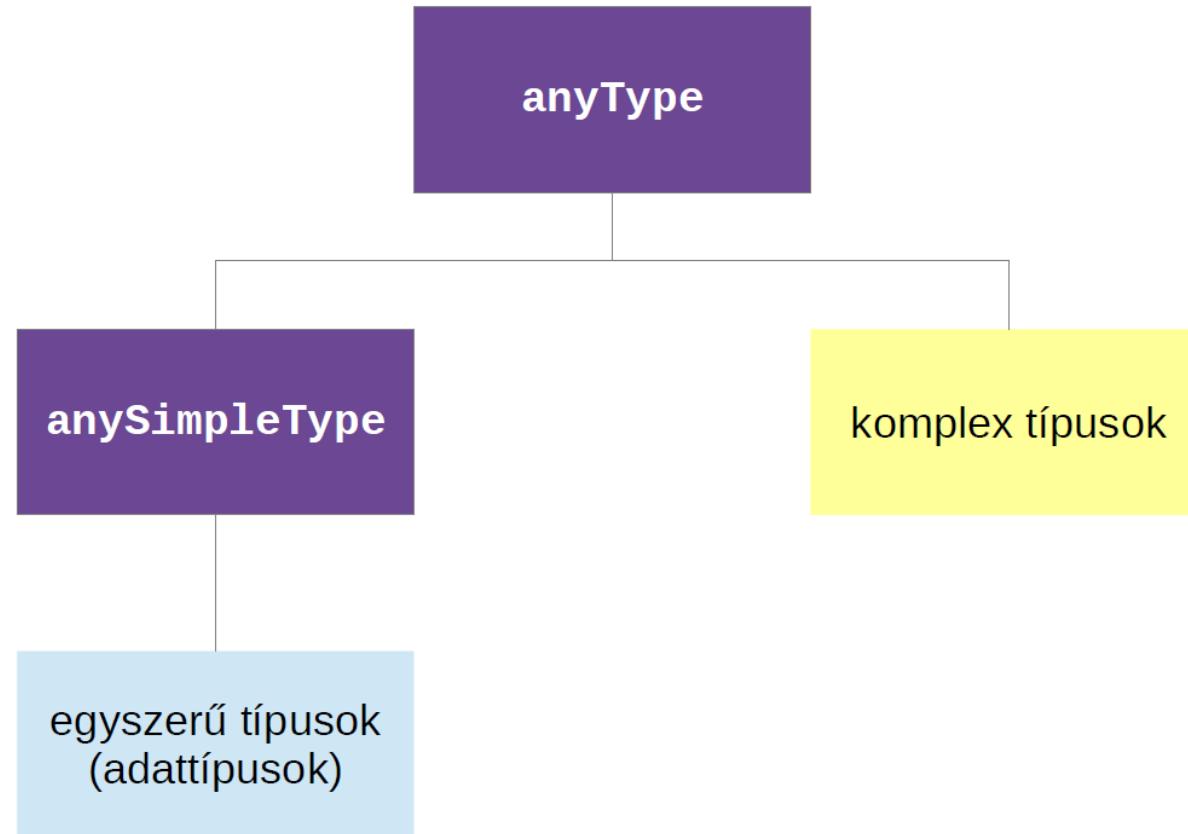
- *Beépített adattípusok*
Amelyeket az [XML Schema: Datatypes] specifikáció definiál
(a boolean, integer és time)
- *Felhasználói adattípusoknak* nevezzük a felhasználók által
definiált adattípusokat (gyakran **Saját típusok** is nevezik).

Megadása: elemTípus formában történik.”

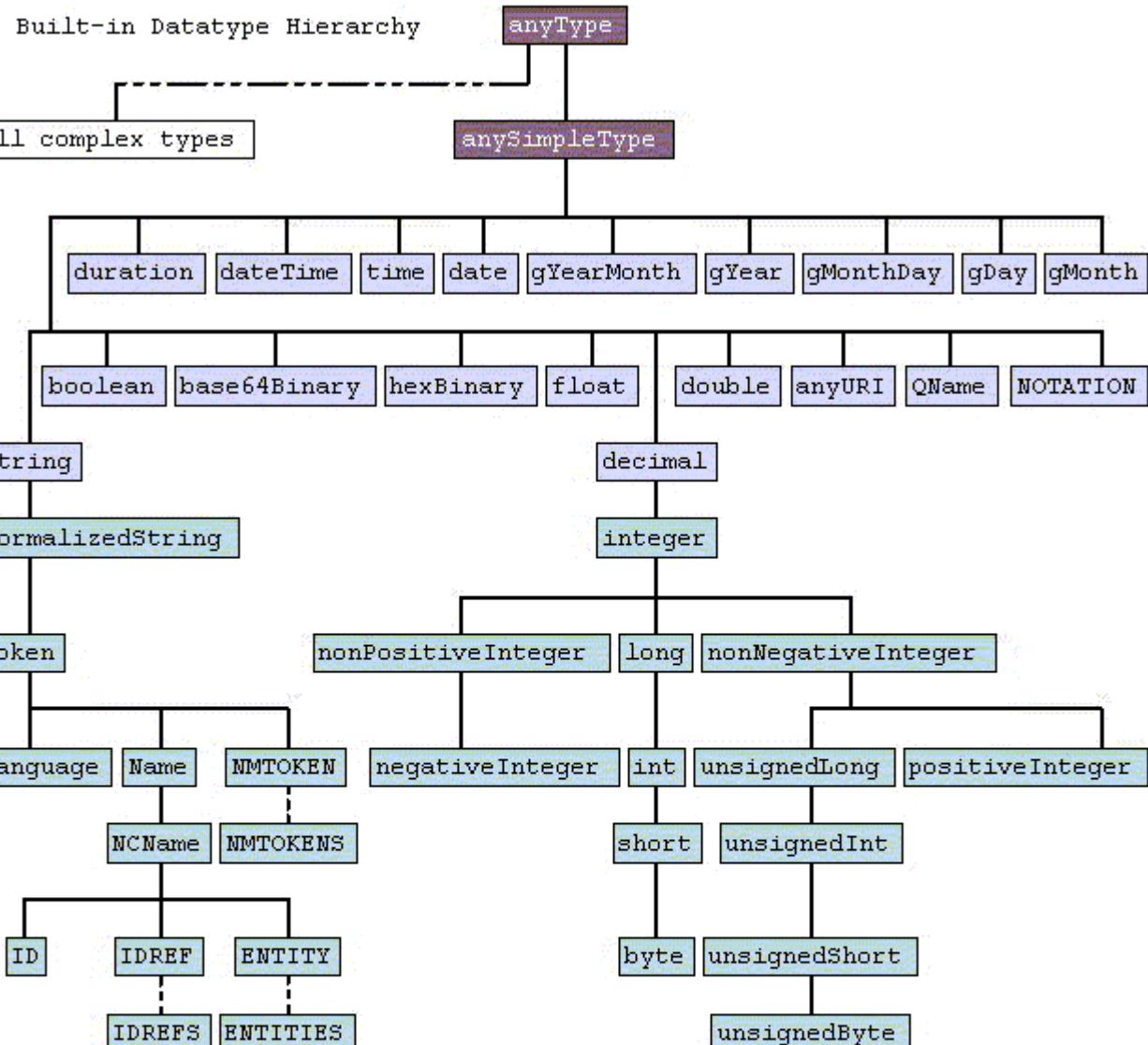
URL: <https://www.w3.org/TR/xmlschema-2/>

https://www.w3schools.com/xml/schema_dt_types_string.asp

Típushierarchia



Adattípusok



Forrás:

<https://www.w3.org/TR/2004/REC-xml-infoset-20040216/#dtypes>



ur types



built-in primitive types



built-in derived types



complex types



derived by restriction



derived by list



derived by extension or
restriction

Elemi adattípusok fajtái

- *Atomi adattípusok*: az értékeit a specifikáció oszthatatlannak tekinti. (pl.: a [double](#) és [string](#) beépített adattípusok).
- *Lista adattípusok*: értékei egy adott *atomi adattípus* értékeiből álló véges sorozatok. Pl.: IDREFS, ENTITIES, NMTOKENS.
- *Unió adattípusok*: értékhalmaza más, tagtípusoknak nevezett *adattípusok* értékhalmazainak halmazelméleti uniója.

Elemi adattípusok szerkezete - megadása

- *atom*

```
<xs:element name="nev" type="xs:típus"/>
```

- *lista*

```
<xs:element name="nev"><xs:simpleType> <xs:list  
itemType="tip"/></xs:simpleType></xs:element>
```

- *union*

```
<xs:element name="nev"><xs:simpleType> <xs:union  
memberTypes="tip tip2
```

Elemi adattípus - unió adattípus - példa

A union elem gyermeként megadható tetszőleges számú típus simpleType elem.

A definiált unió adattípus értéktere a beépített [date](#) és [gYear](#) adattípusok értékterének halmazelméleti uniója. Példa:

```
<xs:simpleType name="dateOrYear">  
    <xs:union memberTypes="xs:date xs:gYear"/>  
</xs:simpleType>
```

Elemtípusok

Elemtípusok megadása:

- A complexType és a simpleType parancsok szolgálnak *új elemtípusok megadására*.
- A simpleType esetén csak *szövegtartalma lehet* az elemnek és *nem tartozhat hozzá jellemző*.
- Az ettől eltérő esetekben a complexType kulcsszót kell használni.

Elemtípusok megadása

Az ide tartozó parancsok alapformája:

```
<xs:simpleType name="típusnev" >  
<!-- leíras --&gt;<br/></xs:simpleType>
```

```
<xs:complexType name="típusnev" >  
<!-- leíras --&gt;<br/></xs:complexType>
```

Elemtípusok megadása – simpleType példa

A *karakterlánc* (string) típus:

Szövegrészt tartalmazó elem vagy attribútum használatakor.

Alakja: type="xs:string"

Pl.: <nev>Adatkezelés XML környezetben</nev>
<xs:element name="nev" type="xs:string"/>

Elemtípusok megadása – simpleType példa

Logikai típus

Igaz vagy hamis értékű elemek vagy attribútumok használatakor.

Alakja: xs:boolean

Pi.: <xs:attribute name= "házas"
type="xs:boolean"/>

Elemtípusok megadása – simpleType példa

Számtípusok - Numerikus értékek esetén:

- xs:byte, xs:short, xs:integer, stb. : egész számok, pl.: 3,
- xs:decimal: decimális számok, például 3,14 (törtrészt is tartalmazhatnak),
- xs:float: 32 biten ábrázolt lebegőpontos számok, pl.: 6,0022E23,
- xs:double: 64 biten ábrázolt lebegőpontos számok,
- xs:positiveInteger, xs:negativeInteger,
xs:nonPositiveInteger, xs:nonNegativeInteger.

Elemtípusok megadása – simpleType példa

Dátum és időtípusok

xs:date - "YYYY-MM-DD" alakú dátum

- <xs:element name="start" type="xs:date"/>
- <start>2022-09-24</start>
- <start>2022-09-24-06:00</start>

xs:time - "hh:mm:ss" alakú időpont

- <xs:element name="start" type="xs:time"/>
- <start>15:00:00</start>

Elemtípusok megadása – simpleType példa

xs:dateTime – "YYYY-MM-DDThh:mm:ss" alakú
dátum és időpont, a T az elválasztókarakter:

```
<xs:element name="startdate"  
type="xs:dateTime"/>
```

```
<startdate>2022-09-30T12:00:00</startdate>
```

Elemtípusok megadása – simpleType példa

Dátum és időtípusok

- gDay: a dátum *nap részét* jelenti (DD)
- gMonth: a dátum *hónap részét* jelenti(MM)
- gMonthDay: a dátum *hónap és nap részét* jelenti(MM-DD)
- gYear: a dátum *év részét* jelenti(YYYY)
- gYearMonth: a dátum *év és hónap részét* jelenti(YYYY-MM)

Elémtípusok megadása – alaptípusok

- **string** - bármilyen érvényes karakter (érvényes karakterek definícióját lásd a cikksorozat 2. részében)

XML alap típusok

- **integer** - <-126789, 126789> intervallumból bármilyen egész szám, vagy nulla. Pl. (-100, 0, 15)
- **positiveInteger** - <1, 126789> intervallumból bármilyen egész szám. Pl. (2, 50, 999)
- **negativeInteger** - <-126789, -1> intervallumból bármilyen egész szám. Pl. (-25, -559, -78902)
- **nonNegativeInteger** - <0, 126789> intervallumból bármilyen egész szám. Pl. (0, 50, 98745)
- **nonPositiveInteger** - <-126789, 0> intervallumból bármilyen egész szám. Pl. (0, -50, -98745)
- **decimal** - pozitív illetve negatív tizedes szám
- **time** - idő a következő formátumban: óó:pp:mm[.mm+-UTC], Pl. (14:30:00, 16:30:00.0, 16:30:00-01:00, 16:30:00.00-01:00)
- **dateTime** - dátum és idő a következő formátumban: éééé-hh-nnTóó:pp:mm.mm+-UTC. Pl. (2004-03-15T15:30:00.0-01:00)
- **gYear** - év a következő formátumban: éééé. Pl. (2004, 1983)
- **gMonth** - hónap a következő formátumban: --hh--. Pl. (--03--, --12--) -> (március, december)
- **gDay** - nap a következő formátumban: ---nn. Pl. (---01, ---23)
- **gYearMonth** - év és hónap a következő formátumban: éééé-hh. Pl. (1983-02)

Elemi jelölő elemek - megszorítás

Az **elemi típusoknál** a leírásban pontosítható a *típus szerkezete*.

A szerkezetet a már meglévő típusokból lehet származtatni.

Hogyan?

- meglévő elemi típusok *megszorításokkal* való kiegészítése,
- elemi típusra épülő *listaképzés* (többértékű elem),
- elemi típusok *uniója*, ekkor a megadott típusok valamelyikéhez tartozhat az elem.

Elemi jelölő elemek - listatípus

A *listatípus* megadása alapvetően kétféle szintaktikával történhet.

- Az egyik esetben egy *nevesített elemi típusra* épül a lista:

```
<simpleType name="ujnev">  
    <list itemType="elemi_típus" />  
</simpleType>
```

Elemi jelölő elemek megadása - listatípus

- A másik lehetőség, hogy itt kerül pontosításra az *alap, elemi típus* is:

```
<simpleType name="ujnev">  
  <list>  
    <simpleType> ... leírás ... </simpleType>  
  </list>  
</simpleType>
```

Elemei jelölő elemek - union

Az *union* szerkezet esetében fel kell sorolni a lehetséges *elemi típusokat*.

Ennek egyszerűbb alakja:

```
<simpleType name="ujnev">  
  <union memberTypes ="elemi_típusok listája" />  
</simpleType>
```

Elemi típus szigorítása - restriction

A meglévő **elemi típusok szigorítása** esetén a felvehető értékhalmazt korlátozzuk:

```
<simpleType name="ujnev">  
    <restriction base="elemi_tipus" >  
        <!-- megszorítások -->  
    </restriction>  
</simpleType>
```

Elemi típus megadása - megszorítás

A *megszorítások* között az alábbi lehetőségek élnek:

- *minExclusive*: minimális érték megadása (nyílt vég)
- *maxExclusive*: maximális érték megadása (nyílt vég)
- *minInclusive*: minimális érték megadása (zárt vég)
- *maxInclusive*: maximális érték megadása (zárt vég)
- *totalDigits*: számoknál a számjegypozíciók darabszáma

Elemi típus megadása - megszorítás

- *fractionDigits*: számoknál a törtrészt leíró számjegyek darabszáma
- *length*: szöveg, lista pontos hossza
- *minLength*: szöveg minimális hossza
- *maxLength*: szöveg maximális hossza
- *enumeration*: felvehető értékek listája
- *pattern*: szöveg formátumát előíró reguláris kifejezés.

Elemi típus megadása - példa

Példa: egy *életkor* típus létrehozását:

```
<simpleType name="eletkor">  
  <restriction base="integer">  
    <minInclusive value="0" />  
    <maxInclusive value="130" />  
  </restriction>  
</simpleType>
```

Attribútum megadása

Az attribútumokat az *attribute* segítségével adhatjuk meg, ahol meg kell adni az attribútum:

- *nevét*,
- a *típusát*, és hogy
- *kötelező* (required) vagy *opcionális* (optional), és esetleg van-e *alapértelmezett* értéke.

Attribútum megadása

Attribútum megadása:

```
<xss:attribute name="név" type="típus" params />
```

Megadható paraméterek:

- *use*: kötelező or opcionális (értéke: required, optional)
- *fixed*: rögzített érték,
- *default*: alapértelmezési érték.

A befoglaló jelölő elem leírásában kell megadni a gyerekelemek után.

Attribútum megadása - példa

Példa: szak esetében csak a szakot kell megadni szöveggel, (string) és van egy evf attribútuma is.

```
<xs:element name="szak">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="evf" type="xs:integer" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

XML schema – beépített típusok - megszorítások

Lehetőség van megszorításokat is alkalmazni a beépített típusokkal, ill. új típusokat is definiálhatunk.

Példa: a kredit értéknek legyen minimum és maximum értéke.

Korábban: <xs:element name="kredit" type="xs:integer"> – e helyett

XML schema – beépített típusok - megszorítások

Példa:

```
<xs:element name="age">
<xs:simpleType>
    <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="120"/>
    </xs:restriction>
</xs:simpleType>
</xs:element>
```

XML schema – beépített típusok - megszorítások

Szöveghossz megszorítások:

Teljes hossz: xs:length, megadása: xs:value

Példa:

```
<xs:restriction base="xs:string">  
    <xs:length value="10"/>  
</xs:restriction>
```

XML schema – beépített típusok - megszorítások

Alsó, felsőhatár:

xs:minLength, xs:maxLength

Példa:

```
<xs:restriction base="xs:string">
    <xs:minLength value="2"/>
    <xs:maxLength value="10"/>
</xs:restriction>
```

XML schema - beépített típusok – felsorolás

Felsorolás:

Csak bizonyos előre meghatározott értékekből vehetnek fel egyet.

Alakja: xs:enumeration, value="érték"

Példa:

```
<xs:restriction base="xs:string">
    <xs:enumeration value="Opel"/>
    <xs:enumeration value="Toyota"/>
    <xs:enumeration value="BMW"/>
</xs:restriction>
```

XML Schema – Saját típusok

Az XML lehetővé teszi *Saját típusok* létrehozását.

Egy *típussal* kapcsolatban *korlátozhatjuk az értéktartományt* vagy
egy *lista formájában* megadhatjuk a *felvehető értékek halmazát*,
stb.

A deklarációt az `xs:simpleType`-al kell kezdeni:

```
<xs:simpleType name="újTípus">  
    típusmeghatározás  
</xs:simpleType>
```

Hivatk.: `<xs:element name="elemnév" type="újTípus"/>`

XML schema – Saját típusok

Vagy

```
<xs:element name="elemnév">
    <xs:simpleType>
        Típusmeghatározás
    </xs:simpleType>
</xs:element>
```

A típusmeghatározás lehet:

- megszorítás (`xs:restriction`) ,
- unió (`xs:union`) vagy
- lista (`xs:list`)

XML schema – Saját típusok

Példa: a *hallgatók* és a *kurzus* ID-nak olyan *string* azonosítókat fogadunk el, amelyek legalább 5 és legfeljebb 10 hosszúak.

```
<kurzus id="GEIAL332-B">
  <nev>Adatkezelés XML környezetben</nev>
  <kredit>5</kredit>
  <hely>XXXII</hely>
  <idopont>Kedd 12:00-13:30</idopont>
  <oktato>Dr. Bednarik László</oktato>
</kurzus>
```

```
<xs:simpleType name="idTípus">
  <xs:restriction base="xs:string">
    <xs:minLength value="5"/>
    <xs:maxLength value="10"/>
  </xs:restriction>
</xs:simpleType>
```

XML schema – Saját típusok

Példa: Ha a jóváhagyás-hoz szeretnénk egy *sajátTípus-t* definiálni, ami csak *igen* vagy *nem* lehet, akkor az *string típus* lesz, és megadott értékeket vehet fel.

```
<kurzus id="GEIAL322-B" jovahagyas="igen">
  <nev>Adatbázis rendszerek I.</nev>
  <kredit>5</kredit>
  <hely>In101</hely>
  <idopont>Kedd 08:00-09:30</idopont>
  <oktato>Dr. Kovács László</oktato>
</kurzus>
```

```
<xs:simpleType name="igenNemTípus">
  <xs:restriction base="xs:string">
    <xs:enumeration value="igen"/>
    <xs:enumeration value="nem"/>
  </xs:restriction>
</xs:simpleType>
```

XML Schema – értéktartomány megszorítások

Értéktartomány megszorítások:

- xs:minInclusive – a legkisebb megengedett érték,
- xs:minExclusive – a legkisebb megengedett érték (a határ nem számít bele),
- xs:maxInclusive – a legnagyobb megengedett érték,
- xs:maxExclusive – a legnagyobb megengedett érték (a határ nem számít bele).

XML schema – felsorolás – saját típus

Példa:

```
<xs:simpleType name="nemTipus">
    <xs:restriction base="xsd:string">
        <xs:enumeration value="férfi" />
        <xs:enumeration value="nő" />
    </xs:restriction>
</xs:simpleType>

<!-- megfelelő típusok ("ferfi", "nő") -->
```

XML schema – beépített típus - lista

Listák:

Egy *elem* vagy *attribútum* egymástól szóközzel elválasztott értékeket vehet fel.

Alakja: xs:lista, itemType: típus

Pl.:

```
<xs:simpleType>
  <xs:list itemType="xs:byte"/>
    <!--123 45 56 31 -->
</xs:simpleType>
```

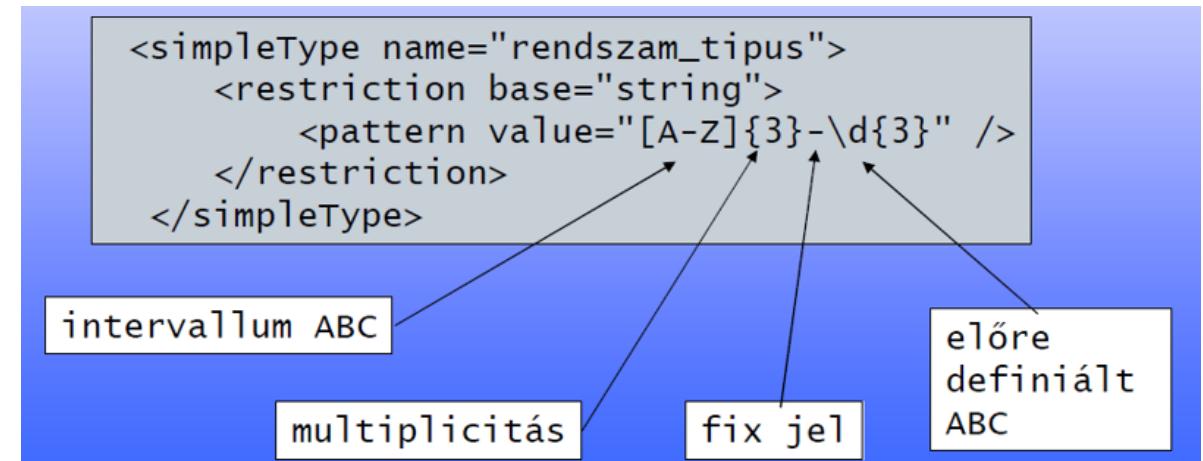
XML schema – beépített típus - reguláris kifejezés

Nagyon hasznos megszorítás. Egy adott sablon szerinti felépítésű értékeket fogad el.

Alakja: xs:pattern, value="sablon"

Mi használhatók a sablonban:

- . (pont) – bármely karakter
- \d – bármely számjegy
- \D – bármely nem számjegy
- \s – bármely üres karakter, etc.



Forrás: KovácsL

XML schema reguláris kifejezés – saját típus

Példa: reguláris kifejezést (pattern) - pontosan leírhatjuk egy karakterszorozatban használható karakterek listáját ill. sorrendjét.

Példa:

```
<xs:simpleType name="szemelyiSzamTipus">
    <xs:restriction base="xsd:string">
        <xs:pattern value="[A-Z]{2}
\d{6}"/>
    </xs:restriction>
</xs:simpleType>
<!-- megfelelő típusok pl. ("SP 225588", "KC 147896", ...) -->
```

Összetett jelölő elemek megadása

Összetett típusúnak nevezzük azon elemeket, melyek *gyermekelemeket, elemjellemzőket, vagy karakteres adatokat* tartalmaznak egyidejűleg.

- **általános összetett típus** (`anyType`) használata esetén az elem tartalmazhat bármilyen *jellemzőt, elemet és karakteres adatot*,
- **saját összetett típus** létrehozása akár *névtelenül*, akár *nevesített* (a típust az egész sémában használhatjuk, mint új típus, típushoz a mi új származtatott típusunk nevét írjuk) módon.

Összetett jelölő elemek megadása

Összetett típusok létrehozása: xs:complexType

Az összetett típusokat *négy* csoportba sorolhatjuk:

- *Üres elemek* (attribútumokat tartalmaznak).
- *Csak elemeket tartalmazó elemek.*
- *Vegyes tartalmú elemek* (elemek+ attribútumok).
- *Sorozatok és választási listák.*

Összetett jelölő elemek megadása – üres elem példa

Üres elem megadása: speciális összetett elem.

Nem kell deklarálunk gyermekelemeket, tehát a *ComplexType* elemnek nem lesznek *element típusú* gyermekelmei.

Létrehozása: xs : complexType és xs : complexContent elemekkel.

```
<xs:element...><xs:complexType/><xs:element>
    <xsd:element name="BR">
        <xsd:complexType>
            <xsd:complexType>
        </xsd:complexType>
    </xsd:element>
```

Összetett jelölő elemek megadása

Gyerekelemeket tartalmaz.

Meg kell adni a *gyerekelemek struktúráját*:

- *Szekvencia* (kötött elemsorrenddel)

<xs:sequence> . . . </xs:sequence>

- *Tetszőleges sorrend* (felsorolás kötetlen sorrend):

<xs:all> . . . </xs:all>

- *Opció lista* (választó lista):

<xs:choice> . . . </xs:choice>

Összetett jelölő elemek megadása – sequence példa

Egyszerűen az xs : complexType segítségével tehetjük meg.

Példa:

```
<xs:element name="kurzus">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="nev" type="xs:string"/>
      <xs:element name="kredit" type="xs:integer">
      <xs:element name="hely" type="xs:string"/>
      <xs:element name="idopont" type="xs:string"/>
      <xs:element name="tanar" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Összetett jelölő elemek megadása - vegyes elemek megadása

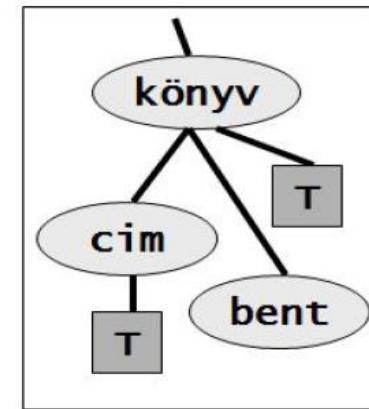
Szöveg és gyerekelementet is tartalmazó elem (mixed) leírásához a szerkezetet egy `<xs:complexType mixed="true" />` elembe kell befoglalni.

Vegyes elem megadása:

speciális összetett elem

```
<xs:element><xs:complexType mixed="true">
<xs:element>
```

```
<xs:element name="könyv">
  <xs:complexType mixed="true">
    <xs:sequence>
      <xs:element name="cím" type="xs:string"/>
      <xs:element name="bent">
        <xs:complexType/>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



```
<könyv>
  ISBN123
  <cím>
    Dada
  </cím>
  <benn/>
</könyv>
```

Összetett jelölő elemek megadása - példa

Vegyes tartalmú elemek (elemeket és attribútumokat tartalmaz).

Létrehozása: xs:complexType és xs:simpleContent segítségével.

Egyes esetekben (szöveg + gyermekelem) használhatjuk az xs:complexType esetében a mixed="true" paramétert is.

Példa:

```
<description>
    Megvalósítás<date lang=„angol”>06.10.22</date>
</description>
```

Összetett jelölő elemek megadása – sorozatok példa

Sorozatok

A sorozatok *gyermekelemek olyan listája*, amely azok pontos *megjelenési sorrendjét* írja elő.

Alakja: xs : sequence

Az adott sorrendben felsoroljuk az elemeket benne.

Összetett jelölő elemek megadása – választás példa

Választások

Ha olyan elemeket akarunk létrehozni, amely választható elemek közül *csak egyet tartalmazhat*, akkor

`xs:choice-al` tehetjük meg.

Összetett jelölő elemek megadása – min-maxOccurs

A *gyerekelemeknél* korlátozható az *előfordulási számosság*.

Az elemhez *minimum* és *maximum* előfordulási darabszámkorlát rendelhető.

```
<xs:element ...  
minOccurs=n1  
maxOccurs=n2 >  
...  
</xs:element>
```

```
<xs:element name="konyv">  
  <xs:complexType>  
    <xs:sequence>  
      <xs:element name="cím" type="xs:string"  
        minOccurs="2" maxOccurs="unbounded"/>  
    </xs:sequence>  
    <xs:attribute name="kod" type="xs:int"/>  
  </xs:complexType>  
</xs:element>
```

Összetett jelölő elemek megadása – min-maxOccurs

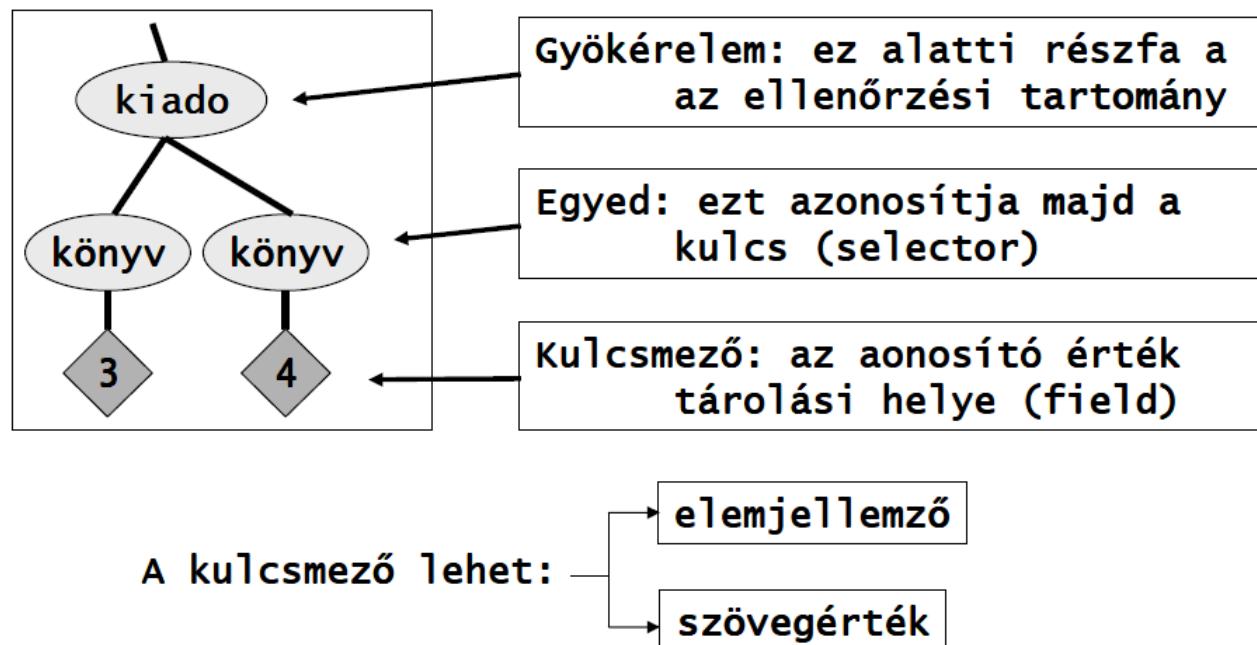
A minOccurs értéke 0 vagy 1 lehet, a maxOccurs tetszőleges egész.

```
<xs:element name="kurzus">
  <xs:complexType>
    <xs:all>
      <xs:element name="nev" type="xs:string"/>
      <xs:element name="kredit" type="xs:integer">
      <xs:element name="hely" type="xs:string"/>
      <xs:element name="idopont" type="xs:string" minOccurs="0"/>
      <xs:element name="oktato" type="xs:string" maxOccurs="5"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

Kulcs integrási elem - kulcs megadása

Kulcs megszorítás megadása

Kulcs: egy részfán belüli egyediség vizsgálatra szolgál

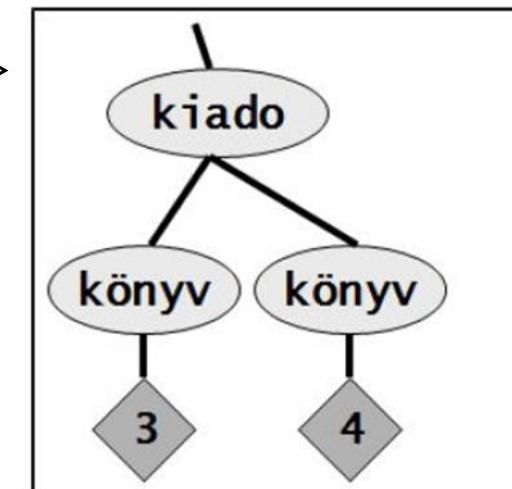


A kulcsfeltételnek egyedi azonosítója van

Kulcs integrási elem - kulcs megadása

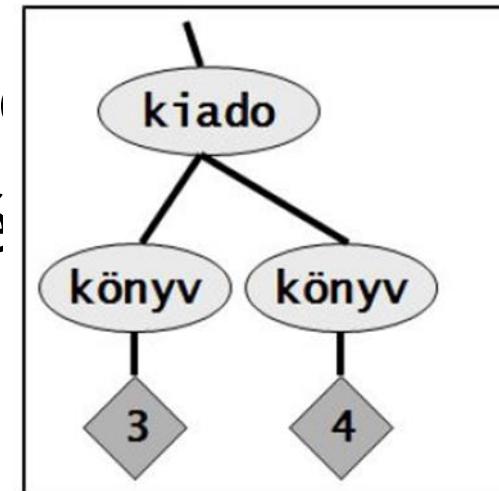
A kulcs megadás szintaktikája:

```
<xs:element name="nev" type="tipus" >  
  ><xs:key name="megkötés neve" >  
    <xs:selector xpath="eleresi_ut1" />  
    <xs:field xpath="eleresi_ut2" />  
  </xs:key>  
</xs:element>
```



Kulcs integráció elem - kulcs megadása

- A selector elemben szereplő eleresi ut1 jelöli azon elemek elérését, - az egyedek szimbolizálják.
- Ezen elemeknek van *egyedi értekkel bíró tulajdonságaiuk*
- Ezen tulajdonságokat a field elemmel lehet meghatározni.
- Az elérési útvonalaknál a selector esetében a gyökérrel minden dokumentumhoz kötődik.
- A field esetében a selector elem lesz a gyökérrel minden dokumentumhoz kötődik.



r.

Kulcs integráció elem - összefoglalás

Kulcs: *egy részfán belüli egyediség vizsgálatra szolgál.*

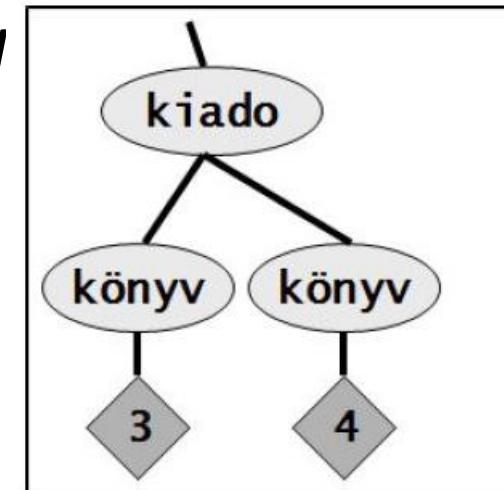
- *Gyökérelem:* ez alatti részfa az ellenőrzési tartomány.
- *Egyed:* ezt azonosítja a kulcs (selector)
- *Kulcsmező:* az azonosító értéke, tárolási helye

A kulcsot a szerkezetleírást tartalmazó

`<xsd:complexType />` elem után kell

A kulcsfeltételnek *egyedi azonosítója* van.

A kulcsmező lehet *elem* és *attribútum* is.



Hivatkozási kulcs elem - keyref

Idegenkulcs: a *keyref* integritási elemet kell megadni a sémában.

A *hivatkozott kulcsot* az *azonosítón* keresztül jelöli ki (*refer*).

Hasonló elemekre épül, mint a kulcs.

Az idegenkulcs feltételnek *egyedi azonosítója* van.

```
<xs:element name="nev" type="tipus" >
    <xs:keyref refer="„kulcs neve" name="megkötés neve" >
        <xs:selector xpath="elérési_út1" />
        <xs:field xpath="elérési_út2" />
    </xs:key>
</xs:element>
```

Idegen kulcs megadása - összefoglalás

Idegenkulcs: egy részfán belül a *hivatkozás érvényességét vizsgálja*.

A *hivatkozott kulcsot* az azonosítón keresztül jelöli ki (refer).

Hasonló elemekre épül, mint a kulcs.

Az idegenkulcs feltételeinek *egyedi azonosítója* van.

```
<xs:element name="nev" type="tipus" >
    <xs:keyref refer=„kulcs neve” name="megkötés neve" >
        <xs:selector xpath="elérési_út1" />
        <xs:field xpath="elérési_út2" />
    </xs:key>
</xs:element>
```

Idegen kulcs megadása - összefoglalás

Idegenkulcs megadása:

```
<xs:keyref refer=kulcsnév name=megkötésnév>
    <xs:selector xpath=elérési_út1 />
    <xs:field xpath=elérési_út2 />
</xs:key>
```

```
<xs:element name="nev" type="tipus" >
    <xs:keyref refer=„kulcs neve” name="megkötés neve" >
        <xs:selector xpath="elérési_út1" />
        <xs:field xpath="elérési_út2" />
    </xs:key>
</xs:element>
```

Hivatkozások használata – elemre és attribútumra kijelöléssel

Az XML-ben elég mély egymásba ágyazási *hierarchia* alakulhat ki, azért az XSD is elég bonyolult és mély lehet.

Célszerű bizonyos részeket kiszervezni, és azokra csak hivatkozásokat megadni.

Pl. a *kurzusfelvetel*-t két nagyobb részre bontjuk, a *hallgato*-ra és a *kurzusok*-ra.

Hivatkozások használata – elemre és attribútumra kijelöléssel

Egy elem, elemjellemző megadható kijelöléssel is

```
<xs:element ref="elemnév" >
```

```
<xs:attribute ref="elemjellemzőnév" >
```

Csak globális elemre vagy elemjellemzőre lehet hivatkozni

Név nem rendelhető a hivatkozó elemhez, a hivatkozott elem nevét veszi fel

```
<xs:element ref="segéd"></xs:element>
```

Forrás: KovácsL

Hivatkozások használat - példa

```
<xs:element name="seged">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="cim" type="xs:string"/>
      <xs:element name="ar" type="xs:integer"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="masik">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="irsz" type="xs:int"/>
      <xs:element name="varos," type="xs:string"/>
      <xs:element ref="seged"/></xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```



```
</masik>
<irsz>
</irsz>
<varos>
</varos>
<seged>
<cim>
</cim>
<ar>
</ar>
</seged>
</masik>
```

ER modell konverziója XMLSchema-ra

Konverziós szabályok

egyed ⇒ **elem**

elemi tulajdonság ⇒ **szöveg elem**

kulcs tulajdonság ⇒ **elemjellemző + kulcs megkötés**

összetett tulajdonság ⇒ **elemeket tartalmazó gyerekelem**

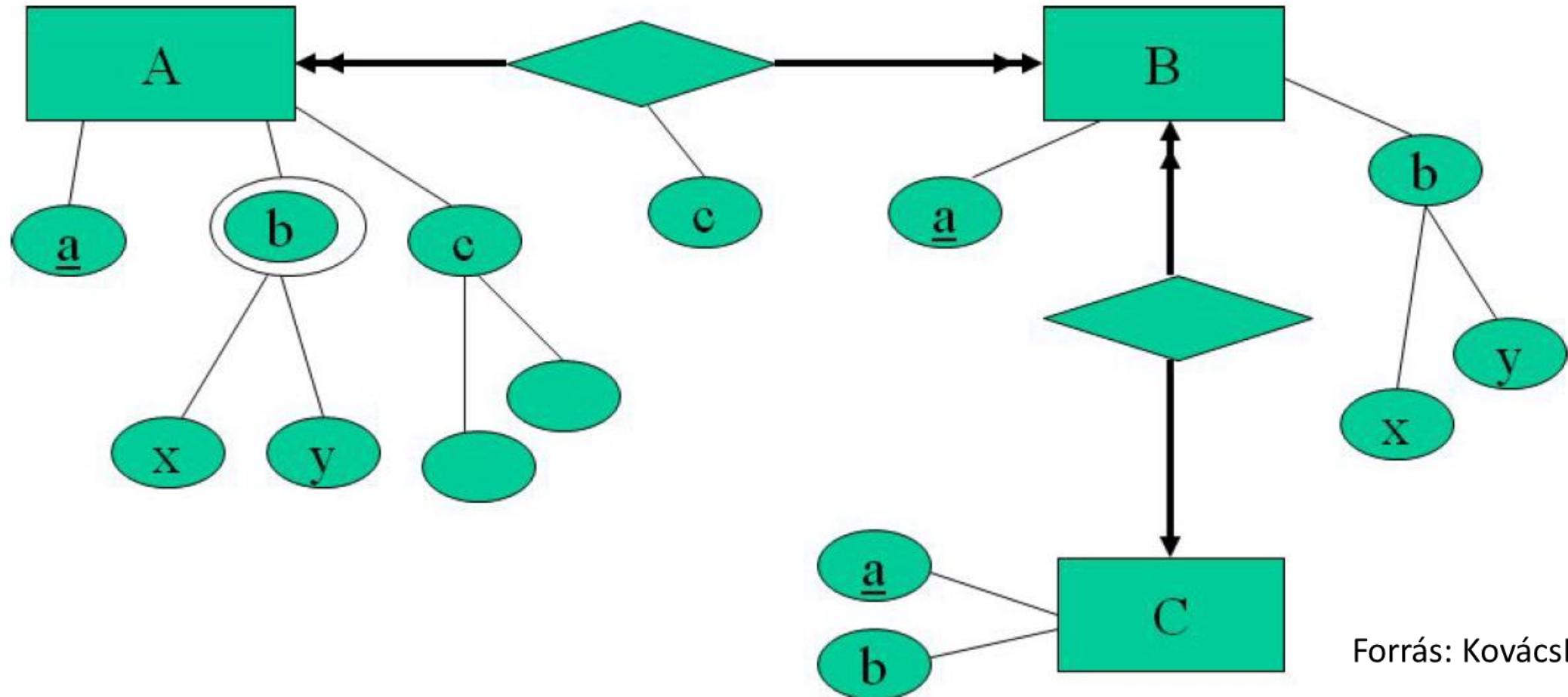
többértékű tulajdonság ⇒ **gyerekelem, ismétlődéssel**

kapcsoló tulajdonság ⇒ **elemjellemző + idegen kulcs megkötés**

1:N kapcsolat ⇒ **elemjellemző + kulcs + idegen kulcs megkötés**

N:M kapcsolat ⇒ **külön kapcsoló elem és idegen kulcsok minden két oldalra**

ER modell konverziója XMLSchema-ra



ER modell konverziója XMLSchema-ra

```
?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="fo">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="A" maxOccurs="unbounded" >
        <xs:complexType mixed="true">
          <xs:sequence>
            <xs:element name="B" type="xs:string"/>
            <xs:element name="C" type="xs:string"/>
          </xs:sequence>
          <xs:attribute name="a1" type="xs:int"/>
        </xs:complexType>
      </xs:element>

      <xs:element name="B" maxOccurs="unbounded" >
        <xs:complexType mixed="true">
          <xs:attribute name="b1" type="xs:int"/>
        </xs:complexType>
      </xs:element>
```

ER modell konverziója XMLSchema-ra

```
<xs:element name="C" maxOccurs="unbounded" >
    <xs:complexType mixed="true">
        <xs:attribute name="c1" type="xs:int"/>
    </xs:complexType>
</xs:element>

<xs:element name="AC" >
    <xs:complexType mixed="true">
        <xs:attribute name="ac1" type="xs:int"/>
        <xs:attribute name="ac2" type="xs:int"/>
    </xs:complexType>
</xs:element>
</xs:sequence>

</xs:complexType>
<xs:key name="K1">
    <xs:selector xpath="A"/>
    <xs:field xpath="@c1"></xs:field>
</xs:key>
```

ER modell konverziója XMLSchema-ra

```
<xs:key name="K2">
    <xs:selector xpath="B"/>
    <xs:field xpath="@b1"></xs:field>
</xs:key>

<xs:keyref refer="K1" name="K11">
    <xs:selector xpath="AC"></xs:selector>
    <xs:field xpath="@AC1"></xs:field>
</xs:keyref>

<xs:keyref refer="K2" name="K21">
    <xs:selector xpath="BC"></xs:selector>
    <xs:field xpath="@BC1"></xs:field>
</xs:keyref>

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

XML Schema feladat – Modularizálás

Ez az *objektumorientált programozási elvekre emlékeztet*, amelyben kis építőkockákból (alaptípusok - egyszerű típusok) építünk fel nagyobb blokkokat (osztályok, struktúrák - összetett típusok).

Felfoghatjuk úgy is, hogy felül létrehozzuk az *elemek, attribútumok absztrakt leírását*, a *komplex típusokban* pedig *példányosítjuk őket, konkrét előfordulásokat hozunk létre belőlük.*"[2]

Felhasznált irodalom

- [1] Kovács László: Adatkezelés XML környezetbe
- [2] Jeszenszky Péter: XML, DE, 2019.
- [3] NetAcademia-tudástár: Az XML Schema, 2000-2003, NetAcademia Kft.
- [4] Dr. Adamkó, Attila: Fejlett Adatbázis Technológiák