

XML Schema speciális elemei

Az előadás anyaga

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

Témakör kérdései

1. *Elemi jelölő elemek megadása*
2. *Összetett jelölő elemek megadása*
3. *Attribútum megadása*
4. *Kulcs integritási elem*
5. *Hivatkozási kulcs elem*
6. 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,)

Az XML – ismételés

„Helyesen formált XML dokumentum formátumon kívül még megemlíthetjük:

- *Az XML, mint metanyelv*: felhasználható nyelvek definiálásra, melyet a *címkekészlet* (tag) és *struktúra* (szókincs és nyelvtan) *határozza meg* (DTD, XML Schema).
- *Az XML, mint szabványcsalád*: több szervezet által is támogatott alapszabványokból épül fel (W3C).
- *Az XML, mint technológia*.

Az XML – ismételés

Az XML, mint technológia, sokféle alkalmazási területen nyújt eszközöket és módszereket a feladatok megoldásához.

Technológia, mert tartalmaz szabványokat és termékeket strukturált dokumentumok készítésére, feldolgozására és megjelenítésére.

Felhasználási területek:

- webes megjelenés (szerver és kliens oldali transzformációk),
- adatcsere (formátum, transzformáció),

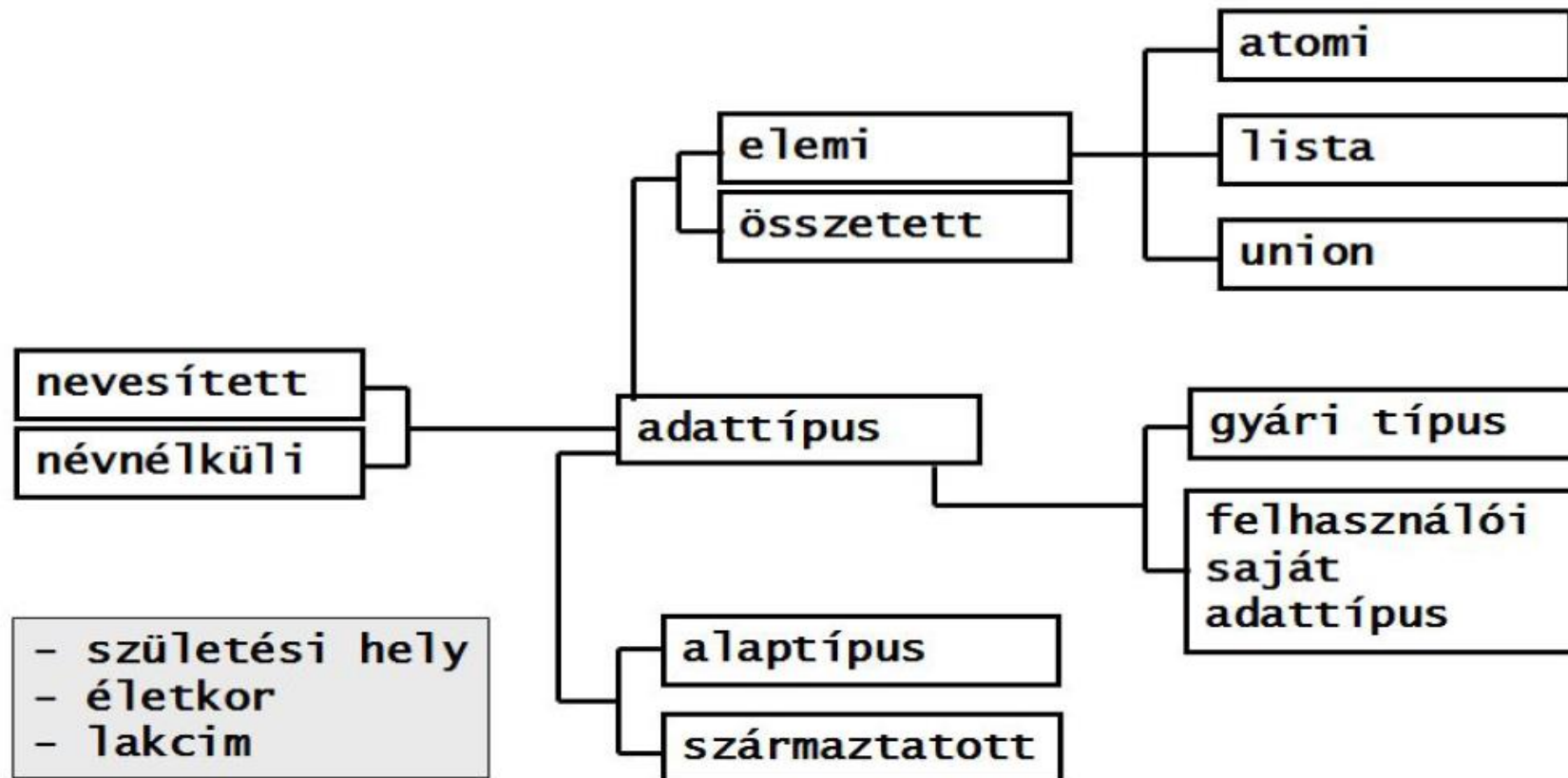
Az XML – ismételés

- szövegek reprezentációja és feldolgozása,
- szövegszerkesztők dokumentum formátuma (OpenOffice, MS Office),
- Web 2.0
- technikai dokumentációk nyelvezete,
- szoftverek konfigurálása,
- felhasználói interfészek definiálása,
- EU önéletrajzok készítése (Europass).”

Az XMLSchema speciális elemei

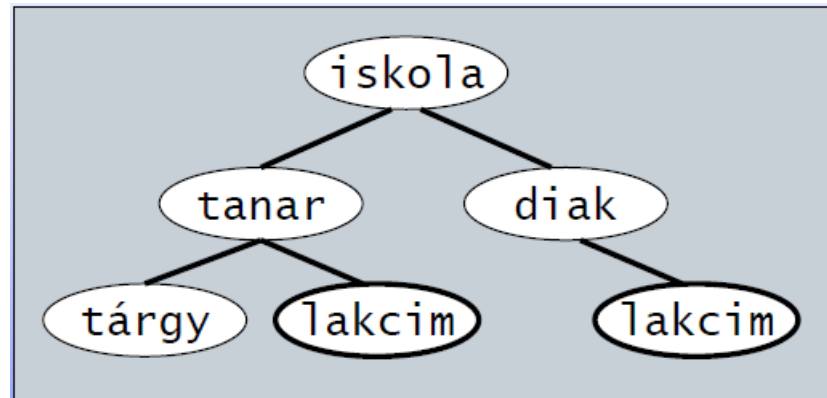
XMLSchema típusrendszere - emlékeztető

„XMLSchema típusrendszere (egyszerűsítve)

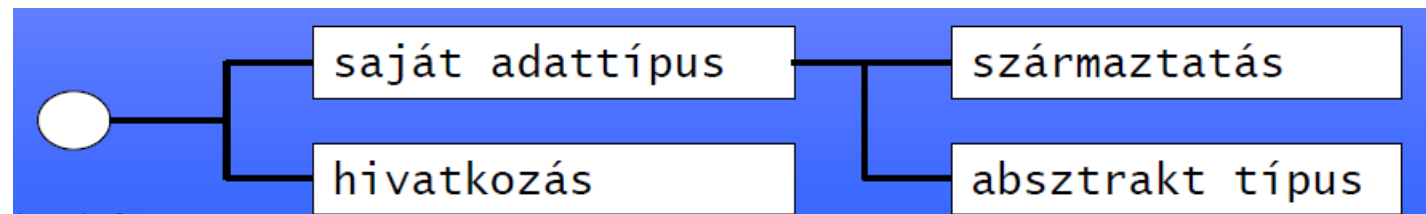


Saját típus használatának előnyei

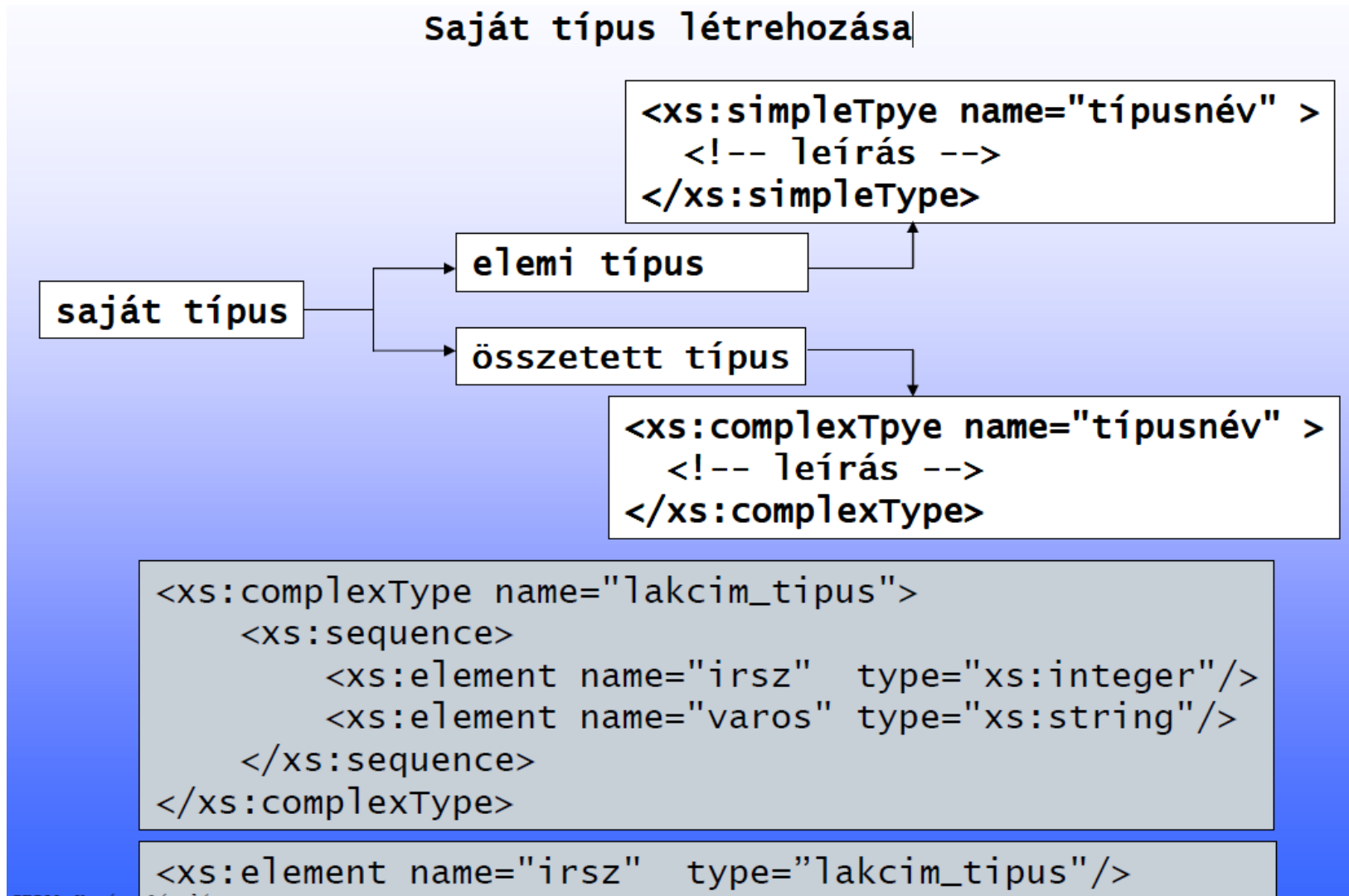
*Egy adott szerkezetű **elem** több helyen is előfordulhat a fában.*



Újrahasznosíthatóság elve: a megírt kód más kontextusban újrafelhasználható.



Saját típus létrehozása



Saját típus létrehozása

Saját típusok létrehozhatók:

- **elemi típusokból** (`simpleType`) vagy
- **összetett típusokból** (`complexType`) való származtatással.

Saját típus létrehozása – elemi típusok

Elemi típusok *származtatása:*

Listaképzés

```
<xs:simpleType name="új_név">  
    <xs:list itemType="elemi_típus" />  
</xs:simpleType>
```

Unionképzés

```
<xs:simpleType name="új_név">  
    <xs:union  
memberTypes="elemi_típusok_listája" />  
</xs:simpleType>
```

Saját típus létrehozása - elemi típusok

Megszorítás

```
<xs:simpleType name="új_név">  
  <xs:restriction base="elemi_típus">  
    <!-- Megszorítások -->  
  </xs:restriction>  
</xs:simpleType>
```

Saját típus létrehozása – alaptípusból saját típus

Lehetőség van **saját típus létrehozására** úgy, hogy az egyes *alaptípusokból egy saját típust származtatunk.*

```
<xsd:element name="EV">
  <xsd:simpleType>
    <xsd:restriction base="xsd:gYear">
      <xsd:minInclusive value="1954"/>
      <xsd:maxInclusive value="2003"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Saját típus létrehozása – megszorítás

`Restriction` kulcsszó segítségével történik a *származtatás*, úgy, hogy a `restriction` elem `base` jellemzőjének adjuk meg azt a típust, amelyből *származtatni szeretnénk a saját típusunkat*.

A `restriction` elem gyermekelemeiként megadhatjuk az új típusra vonatkozó megszorításokat.

Globális és lokális elemek – két fogalom

Mely elemek lehetnek gyökér elemek a kapcsolt dokumentumban

Globális elem: a <schema> gyökér alatt helyezkednek el

Lokális elem: más elem alatt helyezkednek el

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

  <xs:element name="fo">
    <xs:sequence>
      <xs:element name="a1" ...>
    </xs:sequence>
  </xs:element>
</xs:schema>
```



```
<?xml ...>
<fo>
  ...
</fo>
```

```
<?xml ...>
<a1>
  ...
</a1>
```

Gyökér elem csak globális elem lehet

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="seged"></xs:element>
```

Hivatkozások használat_1

```
<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>
```

Elemi típusok származtatása

származtatott elemi típus

megszorítás

listaképzés

unionképzés

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

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

```
<xs:element name="dlista" type="dolgozok_típus"/>  
  
<xs:simpleType name="dolgozok_típus">  
  <xs:list itemType="xs:string"/>  
</xs:simpleType>
```

Elemi típusok származtatása1

Megszorítás: más típus értékkészletének korlátozása

```
<simpleType name="ujnév">  
  <restriction base="elemi_típus" >  
    <!-- megszorítások -->  
  </restriction>  
</simpleType>
```

Az alaptípus lehet saját típus is

```
<xs:simpleType name="eletkor_tipus">  
  <xs:restriction base="xs:integer">  
    <xs:maxInclusive value="120"/>  
  </xs:restriction>  
</xs:simpleType>  
  
<xs:element name="eletkor" type="eletkor_tipus"/>
```

Megszorítás típusok

`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

`fractionDigits`: 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

Adattípus származtatása - megszorítással

Példában a `minInclusive` és `maxInclusive` adattípus-tulajdonságokat alkalmazzuk a *beépített integer adattípusra*, amelyekkel *az alaptípus értékterét* egy tartományra korlátozzuk:

Példa: Az adattípus értékterét az 1, ..., 100 egész számok alkotják.

```
<xs:simpleType name="percent">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="100"/>
  </xs:restriction>
</xs:simpleType>
```

Megszorítás típusok_1

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

```
<simpleType name="rendszam_tipus">  
  <restriction base="string">  
    <pattern value="[A-Z]{3}-\d{3}" />  
  </restriction>  
</simpleType>
```

intervallum ABC

multiplicitás

fix jel

előre
definiált
ABC

Összetett típusok származtatása

Összetett típusok származtatása

- A **származtatás** a *tartalmat meghatározó elemeken* keresztül történik, **létrehozható**:
 - *megszorítással*, vagy
 - *kibővítéssel*.

Összetett típusok származtatása

származtatott összetett típus

megszorítás

kibővítés

A származtatás a tartalom meghatározó elemen keresztül történik

```
<complexType name="ujnév">  
  <complexContent>  
    <restriction base="...">  
      <!-- megszorítások -->  
    </restriction>  
  </complexContent>  
</complexType>
```

```
<complexType name="ujnév">  
  <complexContent>  
    <extension base="...">  
      <!-- új elemek -->  
    </extension>  
  </complexContent>  
</complexType>
```

Összetett típusok származtatása

`complexType.simpleContent`: *attribútummal történő kiegészítés, csak **szövegértékű típust eredményezhet.***

```
<xs:complexType name="név">
  <xs:simpleContent>
    <xs:extension base="elemi_típus">
      <!-- Attribútumok -->
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Összetett típusok származtatása_1

complexType - simpleContent

```
<xs:complexType name="Ar">  
  <xs:simpleContent>  
    <xs:extension base="xs:integer">  
      <xs:attribute name="valuta" type="xs:string"/>  
    </xs:extension>  
  </xs:simpleContent>  
</xs:complexType>
```

Elemjellel történő kiegészítés

Csak szöveértékű típust eredményezhet

Összetett típusok származtatása_2

Új elemekkel,
elemjellemezőkkel
való kiegészítés.

complexType - complexContent

Az új elemek, elemjellemezők szerepelnek

```
<xs:complexType name="alap">
  <xs:sequence>
    <xs:element name="varos" type="xs:string"/>
    <xs:element name="utca" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="ujcim">
  <xs:complexContent>
    <xs:extension base="alap">
      <xs:sequence>
        <xs:element name="ajto" type="xs:integer"/>
      </xs:sequence>
      <xs:attribute name="id" type="xs:string"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Összetett típusok származtatása

Struktúrát definiálni **komplex elemtípusokkal** lehet.

A *komplex elemtípus* két csoportra osztható:

- *egyszerű* és
- *összetett tartalommal* rendelkező.

Mindkettőnek lehet *attribútuma*, de csak az *összetett tartalommal* rendelkezőnek lehet *gyermekeleme*.

Összetett típusok származtatása

Egyszerű tartalommal rendelkező definiálása:

```
<xs:complexType name="address">  
  <xs:simpleContent>  
    <xs:attribute name="city" type="xs:string" />  
  </xs:simpleContent>  
</xs:complexType>
```

Összetett típusok származtatása

Összetett tartalommal rendelkező típus definiálása:

```
<xs:complexType name="address">  
  <xs:complexContent>  
    <xs:sequence>  
      <xs:element name="street" type="xs:string"/>  
      <xs:element name="streetnumber" type="xs:integer"/>  
    </xs:sequence>  
  </xs:complexContent>  
</xs:complexType>
```

Komplex elemtípusoknál meg kell adni egy *kompozitort*.

A kompozitor mondja meg, hogyan kell kezelni a gyermekelemeket.

Összetett típusok származtatása

Három kompozitor létezik:

- *sequence*: gyermekelemeknek abban a sorrendben kell megjelenniük a dokumentumban, mint ahogyan a sémában deklaráltak,
- *choice*: sorrend tetszőleges,
- *all*: csak az egyik gyermekelem jelenhet meg.

Elemcsoportok definiálása

Definiálhatók **elemcsoportok** (substitutionGroup), melyek

- *egymást helyettesítő,*
- *azonos szerkezetű, de eltérő azonosító nevű elemeket* tartalmaznak.

A csoport egyik tagja lesz az azonosító.

Példa:

Lásd: *csoport.xml; csoport.xsd*

Elemcsoportok definiálása

Elemcsoport (helyettesítési) mechanizmus lehetővé teszi a *példányokban* adott *globális elem* más *globális elemekkel* történő helyettesítését.

Globális elemnél előforduló *element* elemekhez megengedett a *substitutionGroup* tulajdonság, értékeként egy globális elem nevét kell megadni.

A *substitutionGroup* tulajdonság értékeként adott nevű elemet helyettesítheti a példányokban az az elem, amelynek deklarációjában megjelenik a tulajdonság.

Elemcsoportok definiálása

- A *helyettesítési csoport* azokat az elemeket jelenti, amelyek *egy adott elemet* helyettesíthetnek.
- A helyettesítő elem *típusa* meg kell, hogy egyezzen a *helyettesítendő elem típusával*, vagy annak típusából származtatott típus kell, hogy legyen.

A helyettesíthetőség *tranzitív tulajdonság*:

azaz, ha a *B* elem helyettesítheti az *A* elemet, a *C* elem pedig a *B* elemet, akkor a *C* elem egyben az *A* elemet is helyettesítheti.

Elemcsoportok definiálása - példa

A példányokban a *surname* és *lastName* elem helyettesítheti a *familyName* elemet, azaz bárhol, ahol előfordulhat a *familyName* elem, megengedettek a *surname* és *lastName* elemek is.

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

```
<xs:element name="surname" type="xs:string"  
substitutionGroup="familyName"/>
```

```
<xs:element name="lastName" type="xs:string"  
substitutionGroup="familyName"/>
```

Elemcsoportok definiálása

Egymást helyettesítő elemek

Azonos szerkezet, eltérő azonosító nevek

A csoport egyik tagja lesz az azonosító

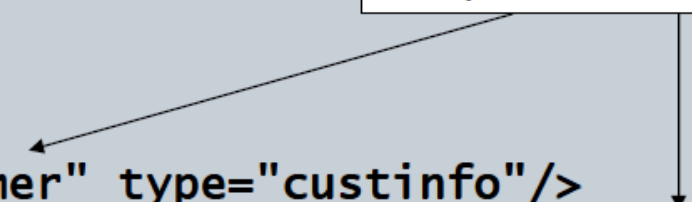
Csak globális elemek között hajtható végre

```
<xs:element ... substitutionGroup="nev">
```

```
<xs:complexType name="custinfo">  
  <xs:sequence>  
    <xs:element ref="name"/>  
  </xs:sequence>  
</xs:complexType>
```

csoportazonosító

```
<xs:element name="customer" type="custinfo"/>  
<xs:element name="kunde" substitutionGroup="customer"/>  
<xs:element name="vevo" substitutionGroup="customer"/>
```



Absztrakt elem definiálása

Globális elemeknél előforduló *element* elemekhez adható meg a logikai értékű *abstract tulajdonság*, amelynek *default értéke false*.

Olyan *globális elemeket* nevezzük *absztrakt elemeknek*, amelyek deklarációjában az *abstract* tulajdonság értéke *true*.

Absztrakt elemek nem fordulhatnak elő a *példányokban*, az ilyen elemekhez tipikusan egy *helyettesítési csoportot* használunk, amelynek tagjai helyettesíthetik az adott *absztrakt elemet*.

Absztrakt elem definiálása

Példa:

Lásd: *absztrak.xml*; *absztrak.xsd*

Olyan csoportazonosító elem, melynek csak valamely helyettesítője szerepelhet a dokumentumban

```
<xs:element name="nn" ... abstract="True">
```

```
<xs:element name="name" type="xs:string" abstract="True" />
<xs:element name="navn" substitutionGroup="name"/>
<xs:element name="nev" substitutionGroup="name"/>

<xs:complexType name="custinfo">
  <xs:sequence>
    <xs:element ref="nev"/>
  </xs:sequence>
</xs:complexType>
```

Absztrakt elem definiálása - példa

Az *uri* globális elem *absztrakt elem*, amely a *példányokban* *nem fordulhat* elő, helyette mindenhol a *helyettesítési csoport* tagjait,

a *http-uri* és *ftp-uri* elemet kell használni.

Az előbbiben *http://*, az utóbbiban pedig *ftp://* kezdőszeletű URI adható meg kizárólag.

Névtér kezelése

Alapesetben a dokumentumhoz *egyetlen névtérhez tartozik*, a **névtér** paraméterei a séma gyökérelemében állíthatók be:

Alapértelmezett névtér beállítása: `targetNamespace`

- **Elemekre vonatkozó beállítás:** `elementFormDefault`
(`qualified`, `unqualified`)
- **Attribútumokra vonatkozó beállítás :**
`attributeFormDefault` (`qualified`, `unqualified`)
- `unqualified` = csak a gyökérnél él a névtér,
- `qualified` = a gyökér alatti elemeknél is él a névtér.

Névtér kezelése - összefoglalás

Alap esetben a dokumentum egyetlen névtérhez tartozik

A névtér paraméterei a séma gyökérelemében állíthatók be

`targetNamespace` : default névtér beállítása

`elementFormDefault` : elemekre vonatkozó beállítás

`attributeFormDefault`: jellemzőkre vonatkozó beállítás

`qualified` : a gyökér alatti elemek is megkapják

`unqualified` : csak a gyökérnél él a névtér

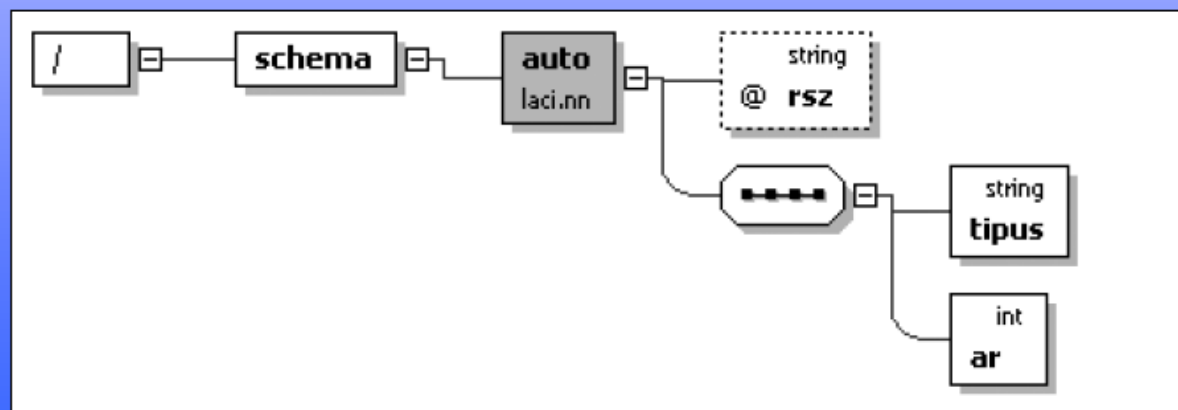


alapértelmezett érték

Default névtér kijelölése - targetNamespace

```
<?xml version="1.0" ?>  
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"  
            targetNamespace="http://iit.uni-miskolc.hu">  
  
  <!-- .... ide jön a séma részletezése .... -->  
  
</xs:schema>
```

Csak a gyökér elem veszi fel ezt a névteret



Megjegyzések felvitele

Megjegyzés típusai:

- fejlesztőnek szóló
- feldolgozó programnak szóló

`<xs:annotation> ... </xs:annotation>`

```
<xs:annotation>
  <xs:documentation xml:lang="Hungarian">
    verziószám: 2.3
    Fejlesztő: K.L.
    dátum: 2007.07.11
  </xs:documentation>
  <xs:appinfo> ... </xs:appinfo>
</xs:annotation>
```

Több séma modul kezelése

Több séma modul kezelése

Külső séma modul beintegrálása

Moduláris felépítés: több rész-séma egyesítése

1. Azonos munkanévtér esetén (INCLUDE)

```
<xs:include schemaLocation="file:...">
```

2. Azonos munkanévtér, verziókövetés (REDEFINE)

```
<xs:redefine schemaLocation="file:...">
```

3. Eltérő munkanévtér (IMPORT)

```
<xs:import schemaLocation="file:..."  
  namespace="file:...">
```

ezen mechanizmussal lehet vegyes névtér eloszlást kialakítani

Séma importálása

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="laci.nn" elementFormDefault="unqualified"
  attributeFormDefault="qualified"
  xmlns:kl="kk.ss">
  <xs:import namespace="kk.ss" schemaLocation="xsd22.xsd"/>
  <xs:element name="auto">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="tipus" type="xs:string"
          form="qualified"/>
        <xs:element ref="kl:ar"/>
      </xs:sequence>
      <xs:attribute name="rsz" type="xs:string"
        use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

befogadó séma

Séma importálása

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="kk.ss" elementFormDefault="qualified">
  <xs:element name="ar" >
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Penznem" type="xs:string"/>
        <xs:element name="Ertek" type="xs:int"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

bejövő séma

A beillesztési névtérnek meg kell egyezni a sémában megadott alapnévtérrel

ER modell konverziója XMLSchema-ra

Konverziós szabályok

egyed \Rightarrow elem

elemi tulajdonság \Rightarrow szöveg elem

kulcs tulajdonság \Rightarrow elemjellemző + kulcs megkötés

összetett tulajdonság \Rightarrow elemeket tartalmazó gyerekelem

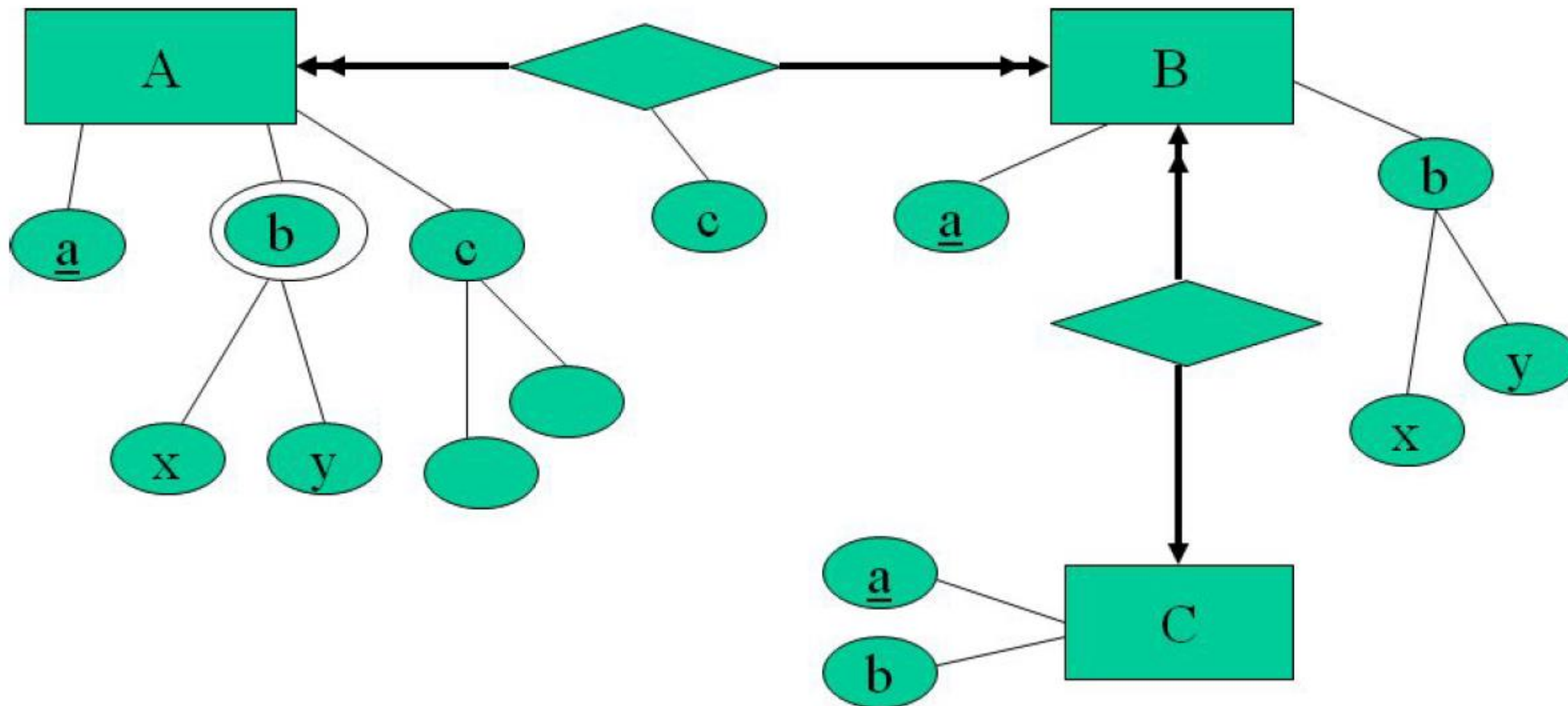
többértékű tulajdonság \Rightarrow gyerekelem, ismétlődéssel

kapcsoló tulajdonság \Rightarrow elemjellemző + idegen kulcs megkötés

1:N kapcsolat \Rightarrow elemjellemző + kulcs + idegen kulcs
megkötés

N:M kapcsolat \Rightarrow külön kapcsoló elem és idegen kulcsok
mindkét 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>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

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:key>

<xs:keyref refer="k1" name="k11">
  <xs:selector xpath="AC"/>
  <xs:field xpath="@AC1"/>
</xs:keyref>

<xs:keyref refer="k2" name="k21">
  <xs:selector xpath="BC"/>
  <xs:field xpath="@BC1"/>
</xs:keyref>

</xs:element>

</xs:schema>
```

Felhasznált irodalom

[1] Kovács László: Adatkezelés XML környezetbe.

<http://moodle.iit.uni-miskolc.hu/login/index.php>

[2] NetAcademia-tudástár: Az XML Schema, 2000-2003, NetAcademia Kft.

[3] Dr. Adamkó, Attila: Fejlett Adatbázis Technológiák.

[4] Jeszenszky Péter: XML, DE, 2019.