Séma szabványok

Az előadás anyaga

Prof. Dr. Kovács László: Adatkezelés XML környezetbe

jegyzete alapján készült el

Séma szabványok

Az előadás anyaga

Prof. Dr. Kovács László: Adatkezelés XML környezetbe

jegyzete alapján készült el

Témakör kérdései

1. Séma szerepe
2. Sémák főbb típusai
3. DTD séma áttekintése
4. *Elem séma* megadása DTD-ben
5. *Attribútum séma* megadása DTD-ben
6. *DTD speciális* elemei
7. ER modell konverziója DTD-re
8. Relációs modell konverziója DTD-re

Igényelt kompetenciák

- Igényelt séma típus kiválasztása
- DTD séma készítése
- ER/relációs séma konverziója DTD-re
- Környezet: XML szerkesztő (Oxygen, EditIX, Eclypse, ….)

XML modell szerepe

„Az XML dokumentumok szerepe: egy *általános adatcsere formátum biztosí*tása a különböző platformon futó programok között.

- Az *XML részben strukturált formátumú*, mert az *elemek rögzített szerkezetűek.*
- Viszont más elemei, mint pl. a *szövegrészek* és az *elemnevek* szabadon alakíthatók.
- A *szabad név* és *szerkezet alakítás* teszi *általánossá a rendszert.*

Az XML modell szerepe

Az XML másik sajátossága, hogy *önleíró formátumú*.

Az XML egy *önleíró nyelv*, mivel tartalmazza az *adatokat* és a hozzájuk tartozó *metaadatokat*, amelyek meghatározzák az *adatok struktúráját* is egy helyen.

Ez a *metaadat* megtalálható az *elemek nevében* és *attribútumaiban,* valamint magában az *XML dokumentum hierarchiájában* is. Pl.: lásd az *orarend.xml*

Az XML modell szerepe

Az *önleíró nyelv* azt is jelenti, hogy ***adott*** *az egyes adatelemek jelentése* a *befoglaló elemekkel* – így nem kell *külön sémaleíró állomány a jel*entés feltárására.

Ez azonban veszélyeket is hordoz magában.

Ezért XML dokumentum előállításakor már korábban jó lenne észlelni esetleg *hibás szerkezetet és tartalmat*.

Ehhez egy külön szerkezetre, *séma leíró részre van szükség*.

XML sémák

XML sémakezelése:

- megadása opcionális,
- *laza kapcsolat* a *séma és XML file között*.

XML séma típusok:

- *Elemi: DTD* - Nyelvtanalapú nyelvek.
- *Haladó*: XMLSchema - Objektumorientált nyelvek.
- *Speciális*: Schematron, Relax NG - Szabályalapú nyelvek.

DTD modell

A *XML szerkezetének ellenőrzésére* szolgáló egyik legelterjedtebb módszer a

DTD (Document Type Denition) mechanizmus.

DTD szerepe: az XML dokumentumok *szerkezetének korlátozása*.

Egy XML dokumentumot *validáltnak, ellenőrzöttnek* nevezünk, ha az teljesíti a megadott *DTD* (vagy más sémaleíró) *követelményeit.*

DTD modell - séma célja

DTD meghatározza:

- milyen *elemek* értelmezettek,
- milyen *elemjellemzők* használhatók,
- milyen az *egyes elemek belső szerkezete* (befoglalt elemek),
- az alkotó *elemek számosság korlátozásai*,
- *értékkorlátozások* bevezetése,
- *egyszerűsítő szimbólumok* definiálása.

DTD modell előnye, hátránya

DTD séma *előnye*, hogy *egyszerű*, ezért készültek hozzá *validálók, ellenőrző programok*.

Ezáltál a *gyakorlatban is „széles” területeken* építenek rá.

DTD séma *hátrányai*:

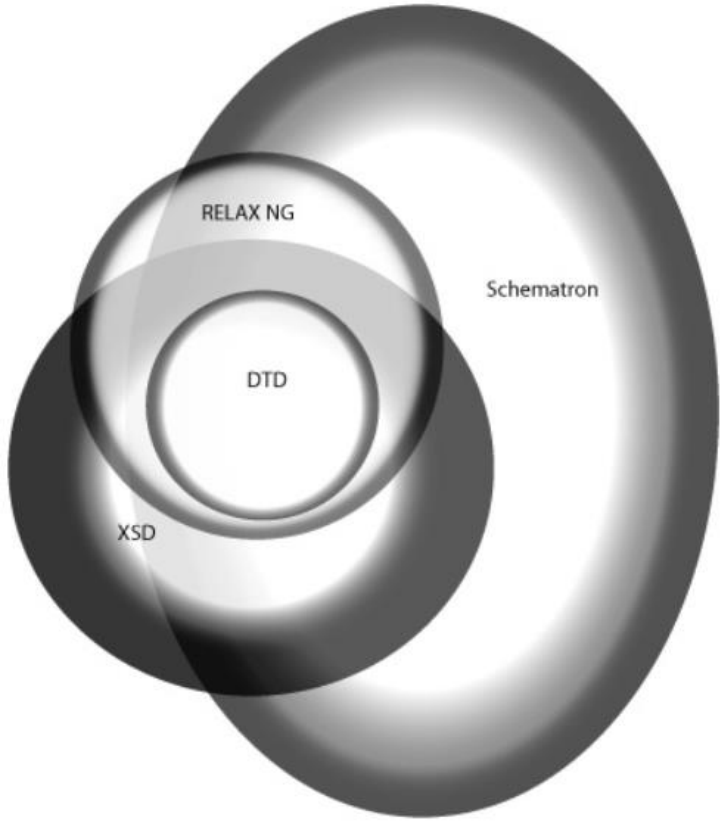
- túl egyszerű,
- nem illeszkedik az XML világra,
- nem támogatja többek között a *névtereket*, a *típusosságat.*

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

DTD séma jellemzői

- elemi *szerkezeti elemek* használ,
- korlátozott *integritási elemek* (típusai: elsődleges kulcs (PK), idegen kulcs (FK), egyedi értékek (UN), nem lehet üres (NN), értékellenőrzés (C))
- globális nevek,
- *noname típusok*,
- van entity,
- nincs *névtér kezelés*,
- nincs *adattípus megkötés*.

XMLSchema jellemzői - összehasonlítás

- *összetett szerkezeti elemek*,
- *statikus* megkötések,
- *gazdag integritási* elemek,
- van *adattípus megkötés*,
- *lokális nevek*,
- *nevesített típusok*,
- *típusok származtatása*,
- van *névtér kezelés*.

DTD jellemzői

DTD fő jellemzői:

- SGML nyelvből öröklődött, nem XML formátumú.
- Az XDM fa szerkezetét adja meg.
- A sémát meg kell adni az XML file elején.
- A séma *belső és külső tárolású* lehet.

DTD séma megadási módjai – belső deklaráció

Hivatkozást mindig XML dokumentum *elején kell elhelyezkednie.*

A hivatkozást egy *megjegyzés jellegű elemben* helyezzük el.

Belső séma megadási mód:

```
<!DOCTYPE gyökérelem_neve [ séma_leírás ]>
```

```
<!DOCTYPE kocka [
  <!ELEMENT kocka (kocka+) >
]>
```

DTD séma megadása – külső deklaráció

A fájl kiterjesztése: **DTD**

Külső séma megadási mód:

```
<!DOCTYPE gyökérelem_neve SYSTEM "külső állomány neve">
```

A kulcsszó `SYSTEM` vagy `PUBLIC` elnevezésű lehet.

A `SYSTEM` akkor használatos, akkor az egy *személyhez* vagy *szervezethez* kötött – ez egy URI takar.

DTD séma megadása – külső deklaráció

A `PUBLIC`, amikor a `DTD`-t egy *szabványalkotó testület hozta létre*, vagy egyszerűen csak a *nagyközönség rendelkezésére* áll.

DTD séma megadása – hivatkozás

Belső deklaráció hátránya:

- minden XML tartalmazza a DTD-t, ami nagyobb méretű lesz,
- a DTD redundánsan szerepel az összes XML-ben,
- ha módosítjuk a DTD-t, akkor minden XML-ben ki kell cserélni.

Külső deklaráció: érdemes ezt használni -

```
<!DOCTYPE jegyzet SYSTEM „jegyzet.dtd">
```

DTD elemei

DTD elemei:

- element,
- attribute,
- entity,
- notation.

```
<!DOCTYPE kocka [
  <!ELEMENT kocka (kocka+) >
]>
```

Element séma megadása

Az **elem** megadásakor megadjuk a *nevet* és a *belső szerkezet* sémáját:

Elem szerkezet általános alakja:

```
<!ELEMENT elemnév (szerkezet)>
```

ELEMENT – elem-típus deklaráció (element type declaration)

Element séma megadása

- Az **elemek** (elements) az XML szerkezet-leírás építőköve.
- Minden elem, ami egy *XML dokumentumban* van megfelel a *DTD egy elemtípusának*.
- Egy elemtípus tartalmazza az *elemtípus nevét*, és a *tartalmára vonatkozó szerkezeti (típus) leírást*.

```
<!ELEMENT elemnév (szerkezet)>
```

Element séma megadása - szerkezet megadásának főbb típusai

- `EMPTY` : üres elem (nincs tartalma), *egytagú tartalom elem*, ebben az esetben a címke nyitó, ill. záró párja között nem található információ (pl. HTML-ben a
)

```
<üres_elem></üres_elem>
<üres_elem />
```

DTD-ben definiálása: `<!ELEMENT üres_elem EMPTY>`

- `#PCDATA` (parsed data, azaz olyan adattartalom, amit az XML-parser feldolgoz), csak *szöveget tartalmazó* elem,

```
<!ELEMENT from (#PCDATA)>
```

Pl.: <from>Ez egy karakter sorozat.</from>

Element séma megadása

- `ANY`: tetszőleges tartalmú elem, nem köti meg a lehetséges tartalmat (lehet *szöveg, elem,* vagy *vegyes* tartalom):

```
<!ELEMENT tetszleges_elem ANY>(szerkezet)
```

csak gyerekelem(eket) tartalmazó elem,

Elem szerkezet általános alakja:

```
<!ELEMENT elemnév (szerkezet)>
```

Pl.: `<!ELEMENT jegyzet (to,from,heading,body)>`

Element séma megadása

- `(#PCDATA | szerkezet)*` : szöveget és gyerek elemeket is tartalmazó elem (MIXED).

```
<!ELEMENT VEGYES (#PCDATA, GYERMEKELEM)*>
```

Element séma megadása

Az *elemtípus* név egy szabályos név – ez egy *elemtípus n*eve.

- A **#PCDATA** (Parsed Character DATA) *karakter tartalmat jelöl*, de ez *tartalmazhat szerkezetleírót* (entitás-referencia).

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

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

<!ELEMENT jegyzet (to,from,heading,body)>

    <!ELEMENT to (#PCDATA)>
    <!ELEMENT from (#PCDATA)>
    <!ELEMENT heading (#PCDATA)>
    <!ELEMENT body (#PCDATA)>
```

Számosság megadása

A *szerkezet* megadásánál meg kell adni:

- milyen *típusú gyerekelemek fordulhatnak* elő,
- milyen lehet azok *sorrendje*.

Ehhez egyrészt:

- *számosság jelző szimbólumokat* használunk a *szerkezeti csoport végén*,
- másrészt a *csoporton belüli előfordulások* is tehetők megkötések.

Számosság megadása

`*` : az *elem* vagy *egyszer sem*, vagy *egyszer*, vagy akár *többször is előfordulhat*,

`+` : az *elem legalább egyszer*, de akár *többször is előfordulhat,*

`?` : az elem *legfeljebb egyszer fordulhat* elő (tehát lehet, hogy egyszer sem),

`(szerkezet)` : pontosan *egy előfordulás*.

Struktúra megadása

Struktúra megadása:

- `Szekvencia:(e1, e2, e3):` elemek szekvenciája.

Pl.: `<!ELEMENT jegyzet (to,from,heading,body)>`

Ha több elem közül kell választani egyet, akkor a **|** jellel adhatjuk meg, pl.: `oktato` vagy `oraado`

- `(e1 | e2 | e3 | ...)`: csak az egyik elem fordul elő a felsoroltak közül.

Element séma megadása - példa

Az *elemeket* a bennük található *tartalommal* definiáljuk.

Ez a *tartalom további elemek* és a rájuk *vonatkozó megkötések lehetnek.*

Megköthető a benn foglalt *elemek listája* (szekvencia) vagy *választási lehetőség* (choice). Példa:

a) `<!ELEMENT spec (front, body, back)>`

A `"spec"` nevű elem belsejében a `"front"`, `"body"` és `"back"` elemeknek kell lenniük, ebben a sorrendben.

Element séma megadása - példa

b) *Választási lehetőség* (choice).

```
<!ELEMENT div1 (p | list | note)>
```

A "div1" nevű elem belsejében a "p", "list" vagy "note" elem található (egy és csak egy közülük)

Element séma megadása - példa

Megkötések tehetők az *elemek előfordulási számára*.

Példák:

a) `<!ELEMENT spec (front, body, back?)>`

A "`spec`" nevű elem belsejében egy darab "`front`" és egy darab "`body`" elemnek kell lennie.

Ezen felül egy "`back`" elem is lehetséges (0 vagy 1-szer).

b) `<!ELEMENT div1 (head, div2*)>`

A "`div1`" elem belsejében a "`head`" elem után tetszőleges számú "`div2`" elem lehet (0 vagy 1 vagy több)

Element séma megadása - példa

c) `<!ELEMENT div1 (head, div2+)>`

A `"div1"` elem belsejében a `"head"` elem után legalább egy vagy több `"div2"` elem lehet (*legalább* 1 vagy több)

A megkötések listája zárójelezhető.

d) `<!ELEMENT div1 (head, (p | list | note)*, div2*))>`

A `"div1"` elem belsejében először egy `"head"` elemnek kell jönnie, majd `"p", "list"` és `"note"` elemek következhetnek.

Opcionálisan az utolsóelemek `"div2"` típusúak lehetnek, de el is maradhatnak.

Element séma megadása - példa

Kevert tartalom: az elem *tartalma karaktersorozatok* és *beágyazott elemek keveréke* lehet.

A beágyazott elemek sorrendje tetszőleges, de típusuk előírható. Példa:

```
<!ELEMENT p (#PCDATA | a | ul | b | i| em)*>
```

A `"p"` elem belsejében *karaktersorozatok*, valamint *tetszőleges számú* és *sorrendű* `"a", "ul", "b", "i"` és `"em"` elemek lehetnek.

Element séma megadása - összefoglalás

Operátor	Példa	Jelentés
+	név+	a „név” nevű elem egy vagy több alkalommal szerepel
*	név*	az elem nulla vagy több alkalommal szerepel
?	név?	az elem nulla vagy egy alkalommal szerepel
nincs	név	az elem pontosan egy alkalommal szerepel (azaz nem ismételhető)
,	név1,név2	a felsorolt elemek ilyen sorrendben, egymás után szerepelnek
|	név1|név2	a két megadott elem közül az egyik, és csakis az egyik szerepel
( )	(név1,név2)|(név3,név4)	a zárójelek lehetővé teszik a többi operátor hatáskörének meghatározását. A példában vagy a „név1” majd „név2” elemek szerepelnek ilyen sorrendben az elemen belül, vagy a „név3” majd „név4” elem.

Attribútum séma megadása - elemtípusok

Az *elemjellemzők* esetén az általános forma a következő:

Általános szerkezet:

```
<!ATTLIST elemnév attribútum adattípus megkötések>
```

A parancsban az *elemnév* a *befoglaló elem neve*.

Az ad*attípus* az attribútum *értékének jelleget* mutatja.

Csak az érték jellegét határozhatjuk meg.

Attribútum séma megadása - elemtípusok

Az egyes jellemzőknek *három típusa* lehetséges:

- *karakterlánc* - szöveges érték,
- *token típus* - mely *tartalma bizonyos korlátok közé* van szorítva,
- *felsorolás típusú* jellemzőknél *megadhatunk egy értékhalmazt*, amelyből a végső jellemzőértek kiválasztható.

<!ATTLIST elemnév attribútum adattípus megkötések>

Attribútum séma megadása - adattípusok

Elemjellemző típusok - *Adattípusok:*

Karakterlánc típus

- `CDATA`: szöveges érték,

Token típus

- `ID`: *egyedi azonosítót* tartalmaz a jellemző (unique), azaz két jellemzőnek nem lehet egyforma tartalmú ID típusa,
- `IDREF`: egy ID értékét tartalmazza - referencia (mutató) egy már létező ID típusú jellemzőre, értéke csak egy létező ID típusú jellemző értéke lehet.

Attribútum séma megadása - adattípusok

- `IDREFS`: *egyidejűleg több*, már létező azonosítóra hivatkozhatunk, referenciákat szóközzel választjuk el egymástól egy tulajdonságon belül. Példa:

```
<!ATTLIST GYERMEK szemelyi_szam ID #REQUIRED szulo_szem_sz IDREFS #REQUIRED>
...
<GYERMEK szemelyi_szam="18456" szulo_szem_sz="22022 25183" />
```

- `ENTITY`: egy **egyed** szimbólum értéket tárolja az attribútum,
- `NOTATION`: egy *külső objektumra utalást* tartalmaz az érték

<!ATTLIST elemnév attribútum adattípus megkötések>

Attribútum séma megadása - megkötések

Elemjellemző megkötések formátuma:

- #REQUIRED: az attribútum érték *megadása kötelező*,
- #FIXED: egy rögzített értékkel rendelkezik az *elemjellemző,*
- #IMPLIED: nem kötelező értéket megadni,
- érték: az alapértelmezési érték kijelölésére szolgál.

```
<!ELEMENT tanar (nev, szak)>
<!ELEMENT tantargy (tnev, osztaly) >
<!ATTLIST tanar kod ID #REQUIRED>
<!ATTLIST tantargy kod ID #REQUIRED>
<!ATTLIST tantargy oktato IDREF #IMPLIED
```

Attribútum séma megadása - példa

Az attribútum lehet *kötelező* vagy *opcionális*: `#REQUIRED`, `#IMPLIED`, `#FIXED`. Példa:

a)
```
<!ATTLIST termdef
     id ID #REQUIRED
     name CDATA #IMPLIED>
```

A "`termdef`" elemnek két attribútuma van: az "`id`" nevű, amely *kötelezően megadandó* és `ID` adattípusú, valamint a "`name`" nevű, amelyik *tetszőleges karaktersorozat* lehet és *opcionális.*

Attribútum séma megadása - példa

b) A `"list"` elemnek egy attribútuma van, melynek neve `"type"`.

Ennek értéke `„golyók", „elrendel"` vagy `„szójegyzék"` lehet, az `„elrendel"` érték alapértelmezett

```
<!ATTLIST list
          type (golyók|elrendel|szójegyzék)
„elrendel">
```

DTD speciális elemei – Egyed szimbólumok

Az *egyedek* az XML-ben, *mint konstansok működnek*.

Azokat az adatokat (elemek, jellemzők, DTDk) amelyeket gyakran használunk, egyedek segítségével beilleszthetjük a dokumentumba (ahányszor) és így a dokumentum méretét is csökkenthetjük.

Esetleges változások esetén elegendő magát az *egyed deklarációt megváltoztatni*.

DTD speciális elemei – Egyed szimbólumok

Az *egyed szimbólumok* lehetnek:

- *általános egyedek:* bármilyen XML vagy nem XML típusú adatot tartalmazhatnak:
  - *értelmezett egyedek* - tartalmazhatnak bármilyen XML típusú adatot:
    - belső értelmezett egyedek,
    - külső általános értelmezett egyedek.

DTD speciális elemei – Egyed szimbólumok

- *nem értelmezett egyedek*: tartalmazhatnak bármilyen nem XML típusú adatot:
  - ➢ külső nem értelmezett egyedek általános deklarációja:

```
<!NOTATION jelölésnév SYSTEM erőforrás>
<!ENTITY egyednév SYSTEM erőforrás NDATA
jelölésnév>
```

DTD speciális elemei – Egyed szimbólumok

- *Paraméter egyedek* DTD-ket tartalmaznak, melyek mindig értelmezettek.
  - *belső paraméter* egyedek deklarálása:

    ```
    <!ENTITY % név érték>
    ```
  - *külső paraméter* egyedek

    ```
    <!ENTITY % név SYSTEM erőforrás>
    ```

DTD speciális elemei – Egyed szimbólumok

Entitások: az XML dokumentumok felépítésénél az *újra felhasználható tartalom készítésére* alkalmas.

Entitás deklarációs formája:

```
<!ENTITY entitás_név „érték”>
```

Az *entitás neve betűvel* vagy *aláhúzással kezdődhet* és betűket, számjegyeket, elválasztó karaktert, aláhúzást és pontot tartalmazhat.

Entitás értéke lehet tetszőleges XML tartalom (azaz címkékkel ellátott szöveg).

DTD speciális elemei – Egyed szimbólumok

Entity definiálása

- **Belső** : `<!ENTITY szimbólumnév “érték">`
- **Külső**: `<!ENTITY szimbólumnév SYSTEM "fájl">`

Szimbólum felhasználása: < > ... &szimbólumnév; ... </ > rövidítést.

```
<!ELEMENT
kartya (nevjegy, lakcim)>
...
<!ENTITY KJ "Katona Jozsef">
```

```
<kartya>
 <nevjegy> &KJ; </nevjegy>
 <lakcim>
  <postai>P u 32</postai>
  <telefon>23-64</telefon>
 </lakcim>
</kartya>
```

```
<kartya>
  <nevjegy>
    Katona Jozsef
  </nevjegy>
  <lakcim>
   <postai>
     P u 32
   </postai>
   <telefon>23-64
   </telefon>
  </lakcim>
</kartya>
```

DTD speciális elemei – Egyed szimbólumok

Külső: `<!ENTITY szimbólumnév SYSTEM "fájl">`

Abban különböznek a *belső általános értelmezett* egyedektől, hogy az egyedek tartalma *külső fáljban* van elhelyezve.

Alakja: `<!ENTITY nev SYSTEM erőforrás>`

ahol, *nev* az egyed neve, amivel hivatkozni fogunk rá, az erőforrás pedig a fájl elérési útja (URI).

nev.xml

```
Michael<KOZEPSONEV>J</KOZEPSONEV>Fox
```

DTD speciális elemei - **paraméteres entity**

A DTD támogatja **paraméter entity** használatát: *a definíciós részben használható.*

A *százalékjel* jelzi, hogy *paraméter entitásról* van szó, amit a DTD-ben alkalmazunk, *egyszerűsíthetik a DTD készítését*.

Entity definiálása: belső: `<!ENTITY % name “value">`

Külső: `<!ENTITY % szimbólum SYSTEM "állomány">`

Paraméterszimbólum felhasználása:

`<!DOCTYPE ... %szimbólumnév; ... >`

DTD speciális elemei - **paraméteres entity**

Példa:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % CIM "(postai, telefon?)">
<!ENTITY % KIEG "e-mail" >
<!ENTITY KJ "Katona Jozsef">
<!ELEMENT kartya (nevjegy, cim , %KIEG;?)>
<!ELEMENT nevjegy (#PCDATA)>
<!ELEMENT cim %CIM; >
<!ELEMENT postai (#PCDATA) >
<!ELEMENT telefon (#PCDATA) >
<!ELEMENT %KIEG; (#PCDATA) >
```

```
<kartya>
  <cim>
   <postai>
   </postai>
   <telefon>
   </telefon>
  </cim>
  <e-mail>
  </e-mail>
</kartya>
```

DTD speciális elemei - **paraméteres entity**

Belső entity definiálása: `<!ENTITY % name “value">`

ahol *name* az egyed neve, melynek érvényes kell lenni, *value* pedig a jellemző értéke.

Példa: létrehozunk egy *ember* nevű belső paraméteregyedet, majd beillesztettük a DTD-be.

DTD speciális elemei - **paraméteres entity**

Külső entity:

```
<!ENTITY % szimbólum SYSTEM "állomány">
```

ahol *szimbólum* az egyed neve, amivel hivatkozni fogunk rá, *erőforrás*, pedig a *külső fájl elérési útvonala*.

DTD speciális elemei – notation (elnevezések)

`NOTATION:` Külső feldolgozó programot azonosító.

```
<!NOTATION név SYSTEM feldolgozó>
```

Az elnevezések (notation) lehetővé teszik típusok (fájl típusok, adattípusok, stb.) deklarációját.

E*lemjellemző*höz

vagy *egyedhez* kötődik.

```
<!DOCTYPE d [
   <!ELEMENT kep EMPTY>
   <!ATTLIST kep nev CDATA #REQUIRED>
   <!ATTLIST kep tipus NOTATION (gif | jpg) "gif">
   <!NOTATION gif  SYSTEM "gifviewer.exe">
   <!NOTATION jpg  SYSTEM "jpgviewer.exe">
]>

<d>
   <kep nev="a.dd" tipus="gif" />
</d>
```

DTD speciális elemei - feltételes definíció létrehozása

A *paraméter szimb*ólumok másik lehetséges alkalmazási területe **a feltételes definíció** létrehozása.

Vannak értelmezők, amelyek támogatják a *feltételes értelmezési szekciók használatát*, azaz a *definíció egyes részei kihagyhatók* vagy *bevonhatók* a *séma felépítésébe*.

Ha a feltétel `INCLUDE`, akkor a feltételes szekció megjelenik a DTD-ben, ha `IGNORE`, akkor nem.

DTD speciális elemei- **feltételes definíció létrehozása**

Feltételes séma elemek

A *kihagyandó rész* megadása:

```
<![IGNORE[ schema definition]]>
```

A *bevonandó rés*z behatárolása:

```
<![INCLUDE[ schema definition]]>
```

DTD speciális elemei - példa

A példában csak a *könyv* és a *CD értékét* kell beállítani, hogy változtassunk a létrehozott séma felépítésen.

A példa esetében a *könyv* egyed fog létrejönni és *a CD* egyed pedig nem.

A *paraméter szimbólumok* segítségével, tehát egy korlátozott rugalmasságot vihetünk be a séma definícióba.

```
<!ENTITY % konyv "INCLUDE" >
<!ENTITY % cd "IGNORE" >

 <![%konyv;[
   <ELEMENT konyv (cim, kiado,..)>
   <ATTLIST konyv isbn ID #REQUIRED>
 ]]>
```

Névterek használata a DTD-ben

Névterek használata DTD-ben nagyon egyszerű.

Az *egyes elemek*, ill. *jellemzők neve* előtt fel kell tüntetni a *névteret kettősponttal* elválasztva. Példa:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE EMBEREK
[
 <!ELEMENT EMBEREK (szulo:EMBER+)>
 <!ELEMENT szulo:EMBER EMPTY>
 <!ATTLIST EMBEREK xmlns:szulo CDATA #REQUIRED>
 <!ATTLIST szulo:EMBER szulo:szul_ev CDATA #IMPLIED>
]
>

<EMBEREK xmlns:szulo="http://www.iit.uni-miskolc.hu">
<szulo:EMBER szulo:szul_ev="1985" />
</EMBEREK>
```

DTD fogalmak – adattípusok (összefoglalás)

- *CDATA:* Az attribútum *csak karakteres adat* lehet. Ezt az adattípust az értelmező nem dolgozza fel, változatlan formában átengedi az ellenőrzésnél.
- *ENTITY:* Az attribútumban *egy entitásra vonatkozó hivatkozás* található.
- *ID:* Az attribútum *egyedi azonosító*, mely a dokumentum *egy meghatározott pontját adja* meg.

DTD fogalmak - összefoglalás

- *IDREF:* Az attribútum *referenciát* tartalmaz egy ID-re, mely a DTD egy más pontján van deklarálva.
- *IDREFS* : Olyan, mint az IDREF, de itt az *attribútum ID-k egy listáját tartalmazza.*
- *NOTATION* : Az attribútum értéke egy *NOTATION* adattípus, melynek deklarációja a DTD egy másik pontján helyezkedik el.

DTD korlátai

- A szintaktikája *nem felel meg az XML szintaktikának*.
- Nem lehet megkötést adni a *szövegértékekre.*
- Nem támogatja a különböző *adattípusok kezelését*, kevés *integritási feltételt támogat*.
- Nem elég *rugalmas a tartalom modell*.
- Nem tud *névtereket kezelni*.
- Nincs *öröklődés,* így nem alkalmas moduláris sémafejlesztésre.
- Hiányos *idegenkulcs* opció, *korlátozott az ID és az IDREF*.

DTD mintapéldák

Mintapéldán keresztül nézzük meg a DTD használatát – *Oxygen szerkesztő* segítségével.

A séma ellenőrzése a szerkesztőben a *Document* menüpont *Validation* alpontjából hívható meg közvetlenül.

A beállítástól függően a szerkesztés alatt, automatikusan is végrehajtható a séma ellenőrzése.

ER modell konverziója DTD-re

A konverziós szabályok:

egyed ⇒ elem

elemi tulajdonság ⇒ #PCDATA típusú gyerek elem

kulcs tulajdonság ⇒ ID típusú jellemző

összetett tulajdonság ⇒ elemeket tartalmazó gyerekelem

többértékű tulajdonság ⇒ gyerekelem, ismétlődéssel

1:N kapcsolat ⇒ IDREF típusú jellemző a gyerek elemnél

N:M kapcsolat ⇒ egy kapcsolóelem létrehozása, melynek van két IDREF típusú jellemzője a kapcsolt elemekre

kötelező kapcsolat ⇒ az IDREF jellemző REQUIRED típusú

DTD mintapéldák

1. feladat

Bl_orarend.xml

Bl_orarend.dtd

2. feladat

- *kurzusfelvetel.xml*
- *kurzusfelvetel.dtd*

DTD mintapéldák

Ez a példa egy *üzenet* leíró XML dokumentum:

```
<?xml version="1.0" ?>
<uzenet>
        <kitol> Peter </kitol>
        <kinek> Feri </kinek>
        <tema> kerdes </tema>
        <leiras> mikor jossz focizni? </leiras>
</uzenet>
```

DTD mintapéldák

A **séma leírása**:

```
<!DOCTYPE uzenet [
    <!ELEMENT uzenet (kitol, kinek, tema, leiras)>
    <!ELEMENT kitol (#PCDATA) >
    <!ELEMENT kinek (#PCDATA) >
    <!ELEMENT tema (#PCDATA) >
   <!ELEMENT leiras (#PCDATA) >
]>
```

DTD mintapéldák – nem illeszkedő

Egy *lehetséges nem illeszkedő XML dokumentum*, melyben

a kötelezően előírt *tema* elem nem szerepel:

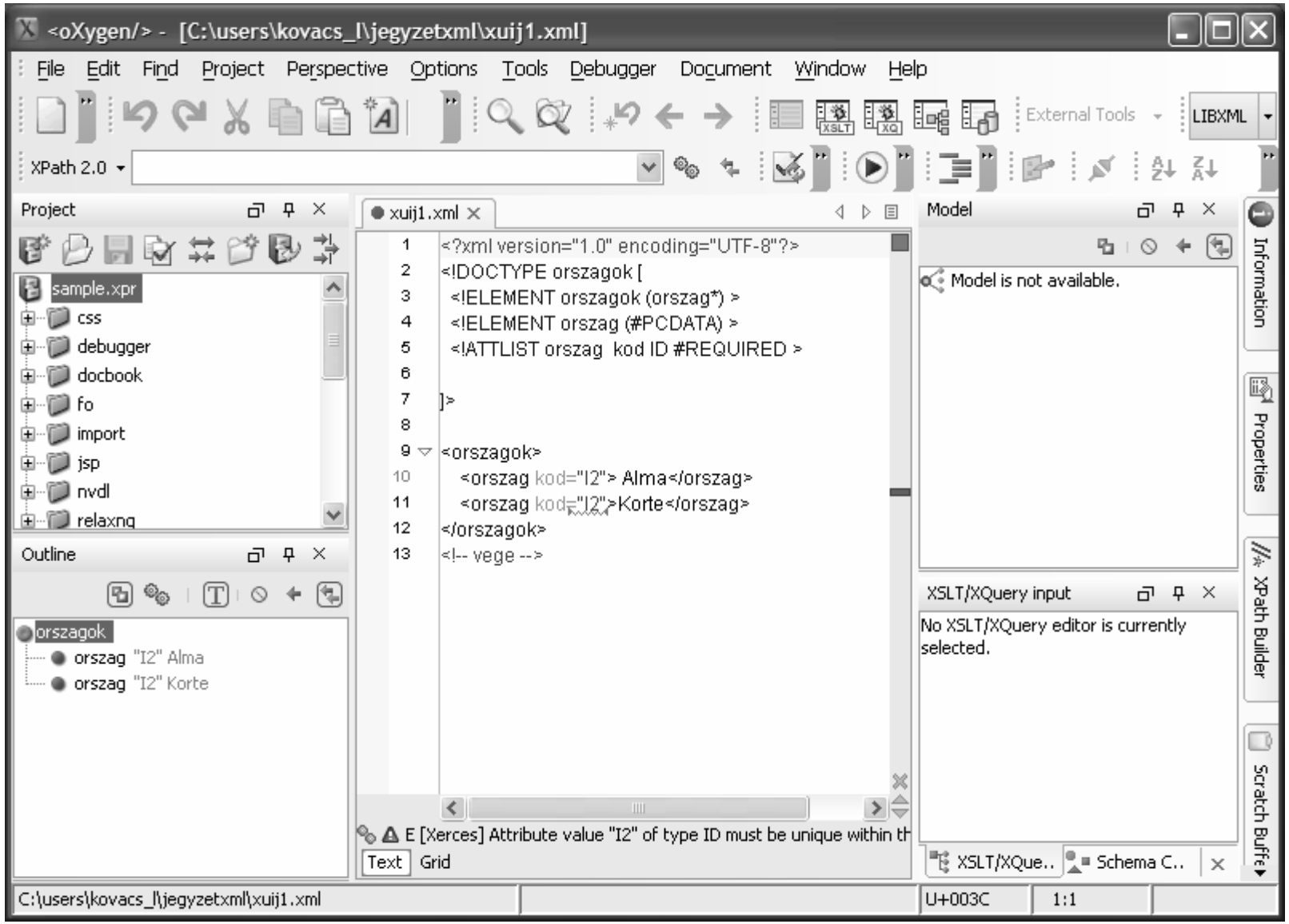
```
<?xml version="1.0" ?>
<uzenet>
   <kitol> Peter </kitol>
   <kinek> Feri </kinek>
   <leiras> mikor jossz focizni? </leiras>
</uzenet>
```

DTD minták - TV műsorújság egyszerűsített leírására szolgál.

```
<!DOCTYPE TVmusor [
      <!ELEMENT TVmusor (ado+)>
      <!ELEMENT ado (leiras, nap*) >
      <!ELEMENT leiras (PCDATA)>
      <!ELEMENT nap (datum, (szunet | musorszam+)+) >
      <!ELEMENT datum (#PCDATA) >
      <!ELEMENT szunet EMPTY >
      <!ELEMENT musorszam  (cim, kezdes, leiras?) >
      <!ELEMENT cim (#PCDATA) >
      <!ELEMENT kezdes (#PCDATA) >
      <!ATTLIST TVmusor datum CDATA #REQUIRED>
      <!ATTLIST ado tulaj CDATA #IMPLIED>
      <!ATTLIST musorszam korhatar
            (N |A14 | A16 | A18) #REQUIRED>
      <!ENTITY LK "Lancos Kotta kiado">
 ]>
```

Forrás: KovacsL

Oxygen XML szerkesztő



Oxygen XML szerkesztő

Főbb jellemzők:

- több OS plattformon fut
- XML, XMLSchema, DTD, XSLT, WSDL, XQuery,
  HTML, XHTML, CSS támogatás, XInclude, XPath
- validáció (DTD, XMLSchema)
- vizuális logikai modell nézet
- környezet érzékeny szerkesztői HELP
- külső adatbázis kapcsolat
- minta generálása
- különböző megjelenitő felületek

(Web Services Description Language, **WSDL**)

Tervezési megfontolások

Hogyan tervezzünk meg egy sémát?

A sématervezésnél az egyik legjobb példakép a *relációs adatbázismodell.*

A relációs adatbázisoknál a viszonylag egyszerűbb a *strukturális rész*, *egyszintű tárolási sémát* kell létrehozni.

Az elemi szinten a *mezők és az*

összefogó szinten a *relációk helyezkednek* el.

Tervezési megfontolások – relációs séma

Például: legyen egy *adatbazis* nevű séma

DOLGOZO**[kod** number(3) primary key, **nev** char(20), **fizetes** number(8)]

relációs sémát *első körben* egy DTD sémára konvertálhatjuk.

```
<!DOCTYPE adatbazis [
    <!ELEMENT adatbazis (DOLGOZO)>
    <!ELEMENT DOLGOZO (kod, nev, fizetes)>
    <!ELEMENT kod (#PCDATA)>
    <!ELEMENT nev (#PCDATA)>
    <!ELEMENT fizetes (#PCDATA)>
]>
```

Tervezési megfontolások - példa

A létrehozott séma *bizonyos tulajdonságokban eltér a relációs sémától*

Pl.: nincs *kulcsérték ellenőrzés* és a *fizetés mező* is tetszőleges *szöveges értéket* vehet fel.

Ez azt jelenti, hogy a DTD nem hordozza magába mindazon *integritási megkötést*, amit a *relációs modell* hordoz.

Ezért több *funkcionális és szintaktikai hiányosságok* vannak a DTD-ben.

DOLGOZO[kod number(3) primary key, nev char(20), fizetes number(8)]

Tervezési megfontolások - példa

Mivel kimaradt a *PK megkötés*, ezért a *kód mezőt* attribútumként vesszük fel – második kör.

A módosított séma alakja:

```
<!DOCTYPE adatbazis [
<!ELEMENT adatbazis (DOLGOZO) >
<!ELEMENT DOLGOZO (kod, nev, fizetes)>
<!ATTLIST DOLGOZO kod ID #REQUIRED >
<!ELEMENT nev (#PCDATA) >
<!ELEMENT fizetes (#PCDATA) >
]>
```

Így viszont **sérül, hogy minden mező azonos sémaszinthez tartozik**.

Tervezési megfontolások

Megoldás: minden mezőt *attribútumba* visszük be.

```
<!DOCTYPE adatbazis [
     <!ELEMENT adatbazis (DOLGOZO) >
     <!ELEMENT DOLGOZO (kod, nev, fizetes)>
     <!ATTLIST DOLGOZO kod ID #REQUIRED >
     <!ATTLIST DOLGOZO nev CDATA #IMPLIED >
     <!ATTLIST DOLGOZO fizetes CDATA #IMPLIED >
]>
```

Ez viszont egybesűríti a elemeket, így a szinteltérés újból jelentkezik - 1NF relációs sémáról átterünk N1NF sémákra.)”

Felhasznált irodalom

- Kovács László: Adatkezelés XML környezetbe
- Jeszenszky Péter: XML, DE, 2019.