Az XML-DTD áttekintése

dr. Kovács László

Az XML-DTD modell szerepe

Az XML dokumentumok szerepe egy általános adatcsere formátum biztosítása a különböző plattformon futó programok között. Az XML részben struktúrált formátumú, hiszen egyes részei, mint a tagok, elemek rögzített szerkezetűek, míg más elemei, minmt például a szövegrészek vagy elemnevek szabadon alakíthatók. A szabad név és szerekezet alakítás teszi általánossá a rendszert. Az XML másik sajátossága, hogy önleíró formátmumú. Vagyis, az egyes adatelemek jelentése a befoglaló elemekkel adott, azaz nem kell külön sémaleíró állomány a jelentés feltárására. A nagy formai és tartalmi szabadság azonban veszélyeket is hordoz magával. Gondoljunk például arra, amikor egy XML formátuban adjuk meg a szállás rendelési adatokat egy szálloda információs rendszerében. A szállás rendelési témakörnek is meglesznek a saját egyedi fogalmai, adatelemei melyeket a névterek segítségével azonosíthatunk a feldolgozó programnál. Mivel a feldolgozó program az egyszerűbb implementáció végett már rögzített adatelemeket vár, a bejövő XML állománynak tartalmaznia kell az igényelt elemeket. Ha hiányzik vagy tévesen megadott egy elem, akkor a kezelő program nem tudja értelmezni az adatokat. Minél később derül ki ez a hiba, annál nagyobb veszteséget jelent. Emiatt jó lenne már korábban, esetleg már az XML dokumentum előállításakor észlelni a hibás szerkezetet és tartalmat. Ehhez viszont egy külön szerkezet, séma leíró részre van szükség.

A XML szerkezetének ellenőrzésére szolgáló egyik legelterjedtebb módszer a DTD (Document Type Definition) mechanizmus. Ez a mechanizmus már az SGML-ben megjelent, onnan került tovább az XML-be. A DTD szerepe az XML dokumentumok szerkezetének korlátozása. A DTD leírás egy sémaleírásnak tekinthető. Ez a sémaleírás a relációs sémához hasonló szerepet tölt be, de annál jóvabb lazább szereppel bír. Egy DTD leírásnak rendszerint több XML dokumentum is megfelel. Egy XML dokumentumot validáltnak, ellenőrzöttnek nevezünk, ha az teljesíti a megadott DTD (vagy más sémalíró) követelményiet.

A DTD esetében a séma leírása történhet az XML dokumentum elején és

egy külön sémalíró állományban is. A séma legfontosabb célja, hogy megadja

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

Az DTD ugyan sokhelyen használatos, nem ezt tekintik a jövő sémaleíró nyelvének. A DTD ugyanis számos olyan korláttal rendlekezik, ami akadályozza rugalmas felhasználását. Ezen megkötöttségekre a későbbiekben visszatérünk. A DTD fő erőssége egyrészt a múltjában, másrészt az egyszerűségében rejlik. Mivel a DTD az SGML-ből jött át, már régóta többen használják, igen elterjedtté vált az elmúlt időben. Az egyszerűsége miatt igen hamar készültek hozzá tartozó validálók, ellenőrző programok. Ezáltal a gyakorlati alkalmazásokban is széles területeken építenek rá. Az ellene felhozott érvek legfontosabb elemei: túl egyszerű, nem illeszkedik az XML világra, nem támogatja többek között a névtereket, a típusosságot. A DTD-t követő újabb sémaszabvány az XMLSchema lesz, amit a következő fejezetben veszünk át.

A DTD szintaktikája

A DTD leírásra utaló hivatkozásnak a viszgálandó XML dokumentum elején kell elhelyezkednie. A hivatkozást egy megjegyzés jellegű elemben helyezzük el. Ha a sémaleírás egy külső állományban van, akkor a hivatkozás alakja:

```
<!DOCTYPE gyökérelem_neve SYSTEM "külsö állomány neve"’>
```

Ha az XML dokumentum belsejében adjuk meg a szerketetre vomnatkozó megkötést, akkor ahhoz a

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

formátum használjuk.

A belső sémalírásban az alábbi szerkezeti komonensek szerepelhetnek:

- elem
- elem jellemző

- egyed szimbólumok

Az elem megadásakor megadjuk a nevét és a belső szerkezetét, sémáját:

```
<!ELEMENT elem_név szerkezet>
```

A szerkezet megadásának főbb típusai:

- EMPTY : üres elem, egytagú tartalom elem
- (#PCDATA) : szöveg értéket tartalmazó elem
- ANY : tetszőleges tartalom megengedett
- (szerkezet) : gyerekelemeket tartalmazó elem (ELEMENT-ONLY típus)
- (#PCDATA | szerekezet)* : vegyes, szöveget és gyerek elemeket is tartalmazó elem (MIXED típus). Ügyelni kell arra, hogy ebben az esetben csak az itt megadott formátumot fogadja el az értelmező, nem lehet például a * sem elhagyni.

Tovább részletezve a szintaktikát, az elem-szerkezet megadása következik. Itt kellmegadni, hogy milyen típusú gyerekelemek fordulhatnak elő, s milyen lehet azok előfordulási sorrendje. Ehhez egyrészt számosság jelző szimbólumokat használunk a szerkezeti csoport végén, másrészt a csoporton belüli előfordulások is tehetők megkötések. A számosságjelzők értékei:

- ()* : tetszőleges számú előfordulás
- ()+ : egy vagy több előforulás
- ()? : nulla vagy egy előfordulás
- () : pontosan egy előfordulás

A csoporton belüli viszonyt megadó elemek:

- e_1, e_2 : az elemek szekvenciája
- $e_1|e_2$: az egyik elem fordul elő a két elemből

Az elemjellemzők esetén az általános formula alakja a következő:

```
<!ATTLIST elem jellemzö_neve tipus  alapérték>
```

A parancsban az elem a befoglaló elem neve. A típus a jellemző értékének jellegét mutatja. Itt nem lehet programozási típusszinten pontosítani az elemhet tartozó értéket (tehát nem lehet megadni, hogy dátum típusú a jellemző). Csak az érték jellegét határozhatjuk meg. Az értelmezett jelleg típusok:

- CDATA : szöveges érték
- $(e_1|e_2)$: értéklista
- ID : egyedi azonosító értéket tartalmaz a jellemző, hasonlít a kulcs integritási megkötéshez
- IDREF : egy ID értékét tartalmazza, hasonlít az idegen kulcs integritási megkötéshez
- IDREFS : több ID értékeit tartalmazza
- NMTOKEN : a jellemző egy XML azonbosító nevet tartalmaz, tehát nem kezdődhet pl. számmal a tárolt érték
- ENTITY : egy egyed szimbólum értékét tárolja a jellemző
- NOTATION : egy külső objektumra utalást tartalmaz az érték (ez lehet például egy külső feldolgozó program azonosítója)

Az ID típusú elemjellemző kulcs szerepét mutatja az is, hogy egy elemnél csak egyetlen ID típusú jellemző szerepelhet.

Az alapértéket megadó paraméterrész azt szabályozza, hogy az elemjellemzőnek kötelező-e értékkel rendelkeznie vagy sem. A paraméter lehetséges értékei:

- #REQUIRED : kötelező explicit értéket tartalmaznia
- #FIXED érték : egy rögzített értékkel rendelkezik a jellemző. A rögzített értéket a FIXED kulcsszó után kell megadni.
- #IMPLIED : nem kötelező értéket megadni
- érték : az alapértelmezési érték kijelölséére szolgál, ilyenkor az IMPLIED típust fogja használni az értelmező

Az elemek és elemjellemzők mellett szerepeltethetők még a DTD leírásban a ritkábban használatos egyed-szimbólumok. Az egyed szimbólumok valamely hosszabb szövegrész rövidítésére, alias névvel való ellátására szolgál. Az egyed-szimbólumok definiálása történhet belső és külső forrásból. A külső forrás esetén egy külső állományból kerül beolvasásra a helyettesítési érték. Belső definíció esetén az XML állományban adjuk meg a szimbólumhoz tartozó értéket. A belső definíció alakja:

```
<!ENTITY szimbólum "érték">
```

A kapcsolódó XML dokumentumban az érték helyett használhatjuk a

```
<  > ... &szimbólum; ... <  >
```

rövidítést. Külső definíció esetén az érték helyén a tartalmazó leíró állomány szerepel:

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

A fenti egyedszimbólumok mellett a DTD szabvány támogatja az úgynevezett paraméter szimbólumok használatát is. A paraméter szimbólom olyan szimbólum, amelyet a DTD definíciós részben lehet használni. Vagyis ezzel a mechanizmussal paraméteres séma definíciót lehet végrehajtani. A séma definícióban egy szimbólum szerepelhet, s annak aktuális tartalamától függ az érvényesnek tekintett séma. A paraméter szimbólum létrehozásának parancsa:

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

A létrehozott szimbólumot a DTD definíciós részben lehet felhasználni a következő módon:

```
<!DOCTYPE ... [
   %szimbólum;
>
```

Alapértelmezésben a szimbólumegy teljes elemet vagy jellemzőlistát tartalmazhat. Erre példa az alábbi kis minta:

```
<!DOCTYPE fo [
   <!ENTITY elem2 "<!ELEMENT al1 EMPTY>

   <!ELEMENT fo (#PCDATA | al1)* >
   %elem2;
>
```

A fenti mechanizmus előnye, hogy segítségével külső állományokból egyes elemek külön is átemelhetők. Ekkor csak az igényelt elemek definíciója jön át, nem szükséges a teljes definíciós állomány alkalmazása.

A paraméter szimbólumok másik lehetséges alkalmazási területe a feltételes definíció létrehozása. Egyes értelmezők támogatják a feltételes értelmezési szekciók használatát, azaz a definíció egyes részei kihagyhatók vagy bevonhatók a séma felépítsébe. A kihagyandó rész megadása:

```
<![IGNORE[
  séma definíció
 ]]>
```

A bevonandó rész behatárolása:

```
<![INCLUDE[
  séma definíció
 ]]>
```

A paraméter szimbólumot ekkor arra lehet használni, hogy kijelöljük, aktuálisan mely részt kell bevonni vagy elhagyni. Az ide vonatkozó sémarészlet:

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

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

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

A példában csak a cd és konyv szimbólumok értékét kell beállítani ahhoz, hogy változtassunk a létrehozott séma felépítésén. A példa esetében a konyv egyed fog létrehönni és a cd egyed pedig nem. Ugyancsak bizonyos értelmezők támogatják azt a lehetőséget, hogy a paraméter szimbólum ne egy teljes elem vagy jellemző tagot helyettesítsen, hanem annak csak egy részét. Ekkor létrehozható olyan elem definíció, melynél például csak a gyerek elem neve fog paraméterként szerepelni:

```
<!ENTITY % gyerek "cim" >

<!ELEMENT konyv (%gyerek;, ar)>
<!ATTLIST konyv isbn ID #REQUIRED>
```

A paraméter szimbólumok segítségével tehát egy korlátozott rugalmasságot vihetünk be a sémadefinícióba.

A megismert séma elemekből építkezve alakítható a helyes DTD leírás. A helyességhez nem elegendő, hogy helyes építőelemeket használjuk, fontos azok egymáshoz való viszonya is. A helyes DTD séma az alábbi megkötéseken alapszik:

- az elem definíciók között szerepelni kell a gyökérelemnek, ami megegyezik a DOCTYPE-nál megadott azonosító névvel
- minden elem csak egyszer szerepelhet
- minden gyerekelem hivatkozáshoz kell léteznie egy elem definíciónak
- egy elem-jellemzőhöz több definíciós sor is tartozhat, melyek közül az a sorrendben első fog érvényesülni.

A DTD séma lényegi elemeit egy formális felírással célszerű összefogni. Indulásként vegyünk egy $\Sigma(\Omega, \Psi)$ szignatúrát, ahol Ω elem azonosító nevek halmazát jelöli, s Ψ elemjellemzők azonosító neveinek halmazát adja meg.

A DTD séma egy $\Delta(E, A, P, R, r)$ ötössel írható le, melyben

- E: az elemek véges halmaza, ahol $E \in \Omega$
- A: az elemjellemzők véges halmaza, ahol $A \in \Psi$
- P: az elemek szerkezetét leíró leképezés. Minden $\epsilon \in E$ elemre teljesül, hogy

  $$P(\epsilon) = \alpha$$

  ahol

  $$\alpha = S \vee \perp \vee \epsilon' \vee \alpha | \alpha \vee \alpha, \alpha$$

  A fenti kifejezésben az elemek jelentése:

  - S: egy szövegérték
  - $\perp$: üres elem

  - ϵ': tartalom elem
  - $|$: opcionális elemek
  - ,: szekvencia

- R: az elemek leképzése az A hatványhalmazára (az elemek jellemzőit jelöli ki)
- r: a gyökérelem

A sémát meg kell tudni különböztetni az XML dokumentumtól. A dokumentum formálisan egy $T(V, \Lambda, \Pi, \Sigma, \Upsilon)$ ötössel adhatjuk meg, ahol

- V: csomópontok halmaza
- Λ: A V-t az Ω-ba leképező típus kijelölési függvény
- Π: A V-t az $S \cup V*$-ba való leképezés, amely a gyerekelemeket adja vissza és a $*$ operátor a Kleene lezártat szimbolizálja
- Σ: A $V \times \Psi$-t az S szöveghalmazba rendelő leképezés
- Υ: A T XML fa gyökér elemét kijelölő elem

A fenti formális felírásra alapozva, egy T dokumentum akkor illeszkedik a Δ sémára, ha

- a Λ leképezés értékkészlete részhalmaza az E-nek
- a $\Pi(v), v \in V$ értéke egy S-beli érték, ha $P(\Lambda(v)) = S$
- a $\Pi(v), v \in V$ értéke egy $s_1, s_2, ...$ értéklista, ha $P(\Lambda(v))$ magába foglalja a $s_1, s_2, ...$ értéklistát.
- a Σ leképezésnek összhangban kellállnia az R leképezéssel.

Mintapéldák

A következőkben héhány mintapéldán keresztül mutatjuk be a DTD használatát. A példákat az Oxygen szerkesztőben próbáltuk ki. A séma ellenőrzése a szerkesztőben a Document menüpont Validation alpontjából hívható meg közvetlenül. A beállítástól függően a szerkesztés alatt, automatikusan is végrehajtható a séma ellenőrzés.

Az első példa egy üzenet leíró sémát és XML fát ad meg. A séma leírása:

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

Egy lehetséges illeszkedő XML fa:

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

Egy lehetséges nem illeszkedő XML dokumentumot is bemutatunk, melyben a kötelezően előírt téma elem nem szerepel.

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

A következő példa egy 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">
 ]>
```

Egy illeszkedő mintaállomány:

```
  <?xml version="1.0" ?>
<TVmusor datum="2006.02">
    <ado tulaj = "&LK;" >
        <leiras>  TV 3 </leiras>
        <nap>
            <datum>hetfo</datum>
            <musorszam korhatar="A16">
                <cim>Reggeli móka</cim>
                <kezdes>10.00</kezdes>
            </musorszam>
            <musorszam korhatar="N">
                <cim>Az okos róka és a romlott párizsi</cim>
                <kezdes>11.50</kezdes>
            </musorszam>
        </nap>
       <nap>
           <datum>kedd</datum>
           <musorszam korhatar="N">
               <cim>A szappan amotizációja</cim>
               <kezdes>10.30</kezdes>
               <leiras>makrogazdasági ismeretek</leiras>
           </musorszam>
       </nap>

    </ado>
</TVmusor>
```

Harmadik példánkban egy iskolai tanár - tantárgy nyilvántartást végző dokumentum sémát mutatunk be.

```
<!DOCTYPE  iskola [
<!ELEMENT iskola (tanar+, tantargy+)>
<!ELEMENT tanar (nev, szak)>
```

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

A sémában szerepelnek ID és IDREF típusú elemjellemzők is. A dokumentum ellenőrző rutin ezesetben vizsgálja az ID értékeinek egyediségét és a hivatkozási értékek helyességét is.

Egy érvényes dokumentum:

```
  <?xml version="1.0" ?>
<iskola>
    <tanar kod="T1">
        <nev>Béla Herceg</nev>
        <szak>Biológia</szak>
    </tanar>
    <tanar kod="T2">
        <nev>Ütvefúró Lajos</nev>
        <szak>Technika</szak>
    </tanar>

    <tantargy kod="O1" oktato="T1">
        <tnev>Hálózatok biológiája</tnev>
        <osztaly>5C</osztaly>
    </tantargy>

</iskola>
```

Tekintsük itt is néhány érvénytelen dokumentumot. A következő példában a második tanar elem nem felel meg a sémaelőírásoknak, ugyanis a tantárgy elem után már nem állhat tanár elem.

```
  <?xml version="1.0" ?>
<iskola>
    <tanar kod="T1">
```

```
        <nev>Béla Herceg</nev>
        <szak>Biológia</szak>
    </tanar>

    <tantargy kod="O1" oktato="T1">
        <tnev>Hálózatok biológiája</tnev>
        <osztaly>5C</osztaly>
    </tantargy>

    <tanar kod="T2">
        <nev>Ütvefúró Lajos</nev>
        <szak>Technika</szak>
    </tanar>

</iskola>
```

A következő példában az a hiba, hogy az ID mező értéke ”1”, amely egy NCName típusú érték. Az azonosító csak azonosító név formátumú lehet, nem kezdődhet számjeggyel az értéke.

```
  <?xml version="1.0" ?>
<iskola>
    <tanar kod="T1">
        <nev>Béla Herceg</nev>
        <szak>Biológia</szak>
    </tanar>
    <tanar kod="T2">
        <nev>Ütvefúró Lajos</nev>
        <szak>Technika</szak>
    </tanar>

    <tantargy kod="1" oktato= "T1">
        <tnev>Hálózatok biológiája</tnev>
        <osztaly>5C</osztaly>
    </tantargy>

</iskola>
```

Tervezési megfontolások

A XML dokumentumok sémájának megtervezése ma még sokkal kevésbé szabályozott folyamat, mint az adatbázisok sémájának meghatározása. A sématervezésnél az egyik legjobb példakép a relációs adatbázismodell. A relációs adatbázisoknál a viszonylag egyszerűbb a struktúrális rész, egyszintű tárolási sémát kell létrehozni. Az elemi szinten a mezők és az összefogó szinten a relációk helyezkednek el. Például a

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

relációs sémát első körben egy

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

DTD sémára konvertálhatjuk. Az már első pillantásra is látszik, hogy ugyan a kapcsolódó XML dokumentum be tudja fogadni a reláció tartalmát, de a létrehozott séma bizonyos tulajdonságokban eltér a relációs sémától. Így például nincs kulcsérték ellenőrzés és a fizetés mező is tetszőleges szöveges értéket is felvehet. Tehát a DTD nem hordozza mindazon integritási megkötést, amit a relációs modell hordoz.

A funkcionális hiányosságok bemutatása mellett több más kritikát kapott a DTD modell, melyek nagyban segítették egy erőteljesebb sémaleíró, az XMLSchema kidolgozását. A következőkben megemlítjük a DTD néhány hiányosságát, amelyek ismerete nem lesz haszontalan a XML adatkezelés és a sématervezés során.

Elsőként nézzük, mik azok az általános funkcionális és szintaktikai hiányosságok, amiket a leggyakrabban felhoznak a DTD ellen:

- a DTD szintaktikája nem felel meg az XML szintaktikának, emiatt a DTD részre nem lehet alkalmazni az XML-hez készített általános feldolgozókat.
- nem lehet megkötést adni a szövegértékekre
- nem támogatja a különböző adattípusok kezelését (csak szöveges adat jelemik meg)

- igen szűk a támogatott integritási feltételek köre
- nem elég rugalmas a tartalom modell
- nem tudja kezelni a névtereket
- túlságosan szoros a kapcsolat ax XML 1.0 szabvánnyal, jobb lenne egy független séma leíró rész
- nem alkalmas a moduláris sémafejlesztése (nincs öröklés, specializáció)
- nehezenen megoldható a DTD sémák automatikus dokumentálása
- nincs elég mechanizmus a parametrikus sémák kialakítására
- túl laza az ID és az IDREF mechanizmus, nem felel meg a valós követelményeknek

Ezen igények mindegyike fontos szempont lehet egy fejlesztés során. Hiszen a moduláris séma megléte esetén könnyen és megbízhatóan lehetne egy korábban elkészült általános sémát specializálni egy részproblémakörre. Ezáltal könnyebben megvalósítható lenne a sémák megosztása és újrafelhasználása is.

A tervezés során további szempont a séma egyértelműsége. A DTD ezen a téren is hagy bizonyos játékteret a fejlesztőnek. Az egyik leggyakrabban feltett és ide vonatkozó kérdés: egy adott tulajdonságot elemként vagy elemjellemzőként tároljunk-e le. Mivel mindkét mód alapvetően alkalmas a kapcsolt értékek tárolására, a döntéskor bizonyos finomabb viselkedési igényeket kell figyelembe venni. Az elem választás jellemzői:

- lehet struktúrálni
- a tulajdonság értéke megjelenhet elemnévként is
- létre lehet hozni üres elemet is

Az elemjellemző főbb tulajdonságai:

- elemi értékű
- lehet egyediséget megkövetelni
- lehet hivatkozási integritási szabályt hozzá rendelni
- alapértelmezési érték köthető hozzá

A relációs séma példánkra visszatérve az első konverziós változatnál kimaradt PRIMARY KEY megkötést behozhatjuk, ha a kod mezőt elemjellemzőként vesszük fel. 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) >
]>
```

Ebben az esetben viszont azt lehet felhozni, hogy a relációs sémában megszokott egyenállúság, vagyis az hogy minden mező azonos sémaszinthez tartozik, itt megsérült. Mint látható, itt a kod és a név mező eltérő sém komponenssel került leírásra. Megoldás lehet az is, hogy minden mezőt elem jellemzőbe viszünk 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 a megoldás viszont nagyon egybesűriti az elemeket, s a szinteltérés megintcsak be fog jönni, ha a 1NF relációs sémáról áttérünk N1NF sémákra.

Korábban említettük, hogy az ID és IDREF mechanizmusok az elsődleges kulcs és az idegen kulcs integritási elemeknek felel meg. Ez azonban csak egy közelítő megfeleltetés. A legnagyobb eltérés abban van, hogy a DTD-ben az ID mező egyediségét nem egy megadott elem előfordulásainak halmazában vizsgálja, hanem az XML dokumentum összes ID elem-jellemzőinek halmazában. Vagyis az ID olyan kulcsot jelent, amelynek egyedisége nem reláció hatáskörű, hanem adatbázis hatáskörű. Így például a

```
<!DOCTYPE  iskola [
<!ELEMENT iskola (tanar+, diak+)>
<!ELEMENT tanar (nev, szak)>
<!ATTLIST tanar kod ID #REQUIRED>
<!ELEMENT diak (nev, osztaly)>
<!ATTLIST diak kod ID #REQUIRED>
<!ELEMENT nev (#PCDATA) >
<!ELEMENT szak (#PCDATA) >
<!ELEMENT osztaly (#PCDATA) >
]>
```

séma esetében egy megadott azonosító érték, mint például K1, nem szerepelhet egyidejűleg tanár és diák azonosítóként. Tehát az alábbi XML dokumentum nem felel meg a sémának:

```
<?xml version="1.0" ?>
<iskola>
  <tanar kod="K1">
     <nev>Nagy Lajos</nev>
     <szak>orosz</szak>
  </tanar>
  <tanar kod="K2">
     <nev>Tamsik Anna</nev>
     <szak>német</szak>
  </tanar>
  <tanar kod="K3">
     <nev>Varga Ilona</nev>
     <szak>angol</szak>
  </tanar>
  <diak kod="K1">
     <nev>Balogh Imre</nev>
     <osztaly>4C</osztaly>
  </diak>
  <diak kod="K2">
     <nev>Bene Sándor</nev>
     <osztaly>4A</osztaly>
  </diak>
</iskola>
```

Ez a fajta kulcs értelmezést sérti az egyedek függetlenségének elvét. Hiszen ekkor egy diák kódjának meghatározásakor olyan kódot sem lehet választani, amely már a tanároknál előfordult. Ez egyfajta függőséget jelent a két mennyiség között.

Hasonlóan torzult az IDREF működése a relációs modellben megszokott idegenkulcshoz viszonyítva. A relációs modellben az idegenkulcsnál adott egy hivatkozott egyed, s az értéknek a hivatkozott kulcsok értékhalmazába kell esnie. A DTD szemléletében az IDREF típusú értéknek a dokumentum ID értékek halmazából kell kikerülnie. A fő eltérés a relációs modellhez képest, hogy így nem tudjuk előírni, milyen elemtípusra mutasson a hivatkozás. A DTD csak azt tudja ellenőrizni, hogy valamilyen elemre mutasson, de az elem típusát nem lehet korlátozni. Például a

```
<!DOCTYPE  iskola [
```

```
<!ELEMENT iskola (tanar+, diak+)>
<!ELEMENT tanar (nev, szak)>
<!ATTLIST tanar kod ID #REQUIRED>
<!ELEMENT diak (nev, osztaly)>
<!ATTLIST diak kod ID #REQUIRED>
<!ATTLIST diak konzulens IDREF #IMPLIED>
<!ELEMENT nev (#PCDATA) >
<!ELEMENT szak (#PCDATA) >
<!ELEMENT osztaly (#PCDATA) >
]>
```

esetében az a helyes szemantikai értelmezés, hogy a konzulens elemjellemző egy tanár elemet jelöl ki. A DTD viszont csak azt tudja előírni, hogy valamilyen tanár vagy diák elemre mutasson a hivatkozás. Emiatt a séma alapján helyesnek kell tekinteni az alábbi XML dokumentumot:

```
<?xml version="1.0" ?>
<iskola>
  <tanar kod="K1">
     <nev>Nagy Lajos</nev>
     <szak>orosz</szak>
  </tanar>
  <tanar kod="K2">
     <nev>Tamsik Anna</nev>
     <szak>német</szak>
  </tanar>
  <tanar kod="K3" >
     <nev>Varga Ilona</nev>
     <szak>angol</szak>
  </tanar>
  <diak kod="D1" konzulens="K1">
     <nev>Balogh Imre</nev>
     <osztaly>4C</osztaly>
  </diak>
  <diak kod="D2" konzulens="D1">
     <nev>Bene Sándor</nev>
     <osztaly>4A</osztaly>
  </diak>
</iskola>
```

Ez a hiányosság jelentősen csökkenti az XML dokumentumok helyesség ellenőrzési képességét.

A DTD séma jellemzésénél érdemes arra is kitérni, hogy léteznek olyan, formálisan helyes DTD sémaleírások, melyekez nem tartozik megvalósítható XML dokumentum. Az ellentmondás oka, hogy az XML dokumentumokat végesnek tételezzük fel, s létezheznek olyan DTD sémák, melyeket csak végtelen XML dokumentumok elégítenek ki. Belátható, hogy a DTD akkor eredményez végtelen XML dokumentumot, ha a sémában kötelező, nem kikerülhető ciklusok fordulnak elő. Azaz egy elem rekurzívan és kötekezően tartalmazza magát minden érvényes XML dokumentumban.

Példaként vegyük az alábbi sémát:

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

Ekkor a kocka elem kötelezően tartalmaz egy újabb kocka elemet. Ezt a sémát nem lehet véges sok egymásba ágyazott kocke elem megvalósítani. Természetesen a ciklus több elemen keresztül is záródhat:

```
<!DOCTYPE kocka [
  <!ELEMENT kocka (szelet+) >
  <!ELEMENT szelet (pozicio, kocka) >
  <!ELEMENT pozicio (#PCDATA) >
]>
```

Ha nem kötelező a ciklus, akkor találhatunk véges méretű illeszkedő XML dokumentumot. Például a

```
<!DOCTYPE kocka [
  <!ELEMENT kocka ((#PCDATA) |kocka)* >
]>
```

esetében a

```
<?xml version="1.0" ?>
<kocka>
  <kocka>
     <kocka>
        Natasa
     </kocka>
  </kocka>
</kocka>
```

egy érvényes XML dokumentum lesz.

A DTD tervezésének egy további problémája, hogy nem lehet vele egyértelműen figyelembe venni a valós értékek közötti funkcionális függőségeket. A funkcionális függőség (FD) az adatbázis tervezés egyik fontos alapeleme, hiszen segítségével lehet a felesleges redundanciát és anomáliákat megszüntetni. Az XML dokumentum hierarchikus jellegéből következően a redunancia esélye igen nagy. Vegyük például azt az esetet, amikor egy tantárgyakat, hallgatókat és a jelentkezésket tartjuk nyilván egy dokumentumban. Az egyik séma megvalósítási út az ID és IDREF elemjellemzők használata:

```
<!DOCTYPE iskola[
  <!ELEMENT iskola (targy*, diak*) >
  <!ELEMENT targy (nev, kredit) >
  <!ATTLIST targy kod ID #REQUIRED>
  <!ELEMENT nev (#PCDATA) >
  <!ELEMENT kredit (#PCDATA) >
  <!ELEMENT diak (nev, oszaly) >
  <!ATTLIST diak kod ID #REQUIRED>
  <!ATTLIST diak targya IDREF #IMPLIED>
  <!ELEMENT osztaly (#PCDATA) >
]>
```

A kapott séma problémája, hogy így egy diák csak egy tárgyhoz kapcsolódhat. A modellünkben ezzel szemben mi azt tesszük fel, hogy több tárgyat is tud hallgatni az illető. Ehhez átalakítjuk a sémát:

```
<!DOCTYPE iskola[
  <!ELEMENT iskola (targy*, diak*) >
  <!ELEMENT targy (nev, kredit) >
  <!ATTLIST targy kod ID #REQUIRED>
  <!ELEMENT nev (#PCDATA) >
  <!ELEMENT kredit (#PCDATA) >
  <!ELEMENT diak (nev, oszaly, targya*) >
  <!ATTLIST diak kod ID #REQUIRED>
  <!ELEMENT targya EMPTY>
  <!ATTLIST targya tkod IDREF #IMPLIED>
  <!ELEMENT osztaly (#PCDATA) >
]>
```

Ez már jobb megoldás, de ekkor meg az előbb emített problémába ütközük: ez a DTD leírás nem tudja garantálni, hogy a megadott IDREF érték valóban

egy tárgy elemre mutasson. Emiatt lehetne azt az utat is választani, hogy behozzuk a hivatkozott elemeket a hivatkozó elemek alá. Most például a tárgy elem alá helyezzük le az ott tanuló diákokat:

```
<!DOCTYPE iskola[
  <!ELEMENT iskola (targy*) >
  <!ELEMENT targy (nev, kredit, diak*) >
  <!ATTLIST targy kod ID #REQUIRED>
  <!ELEMENT nev (#PCDATA) >
  <!ELEMENT kredit (#PCDATA) >
  <!ELEMENT diak (kod,nev, osztaly) >
  <!ELEMENT osztaly (#PCDATA) >
  <!ELEMENT kod (#PCDATA) >
]>
```

Ekkor azonban egy diák elem többször is előfordulhat az XML dokumentumban, mint az alábbi példa mutatja, hiszen egy diák több tárgyat is hallgathat:

```
<?xml version="1.0" >
<iskola>
  <targy kod="T1">
    <nev> Adatbázisok </nev>
    <kredit> 5 </kredit>
    <diak>
      <kod> H1 </kod>
      <nev> Kalap Ede </nev>
      <osztaly> 4A </osztaly>
    </diak>
    <diak>
      <kod> H2 </kod>
      <nev> Nagy Ferenc </nev>
      <osztaly> 4A </osztaly>
    </diak>
  </targy>
  <targy kod="T2">
    <nev> Hálózatok </nev>
    <kredit> 5 </kredit>
    <diak>
      <kod> H1 </kod>
      <nev> Kalap Ede </nev>
```

```
      <osztaly> 4A </osztaly>
    </diak>
    <diak>
      <kod> H3 </kod>
      <nev> Sós László </nev>
      <osztaly> 4B </osztaly>
    </diak>
  </targy>
</iskola>
```

A mintában a következő hiányosságokat fedezhetjük fel:

- egy diák adait több helyen is ismétlődnek, redundancia lépett fel
- a diákhoz a redundancia miatt nem lehet ID kulcs mezőt rendelni, elveszik a diák elemek egyedisége
- a diák elem alá behozott kód mező nem tudja átvenni a kulcs szerepét, hiszen nem lehet az egyediségét biztosítani.

References

[1] SAX ...