XML alapú adatbázis-kezelés (Katona Endre diái alapján)

Adatstruktúrák:

- Digitális kép, hang: teljesen strukturálatlan
- A web (linkek): részben strukturált
- Relációs: teljesen strukturált

Motiváció:

- Ismeretlen adatstruktúrák fogadása. (Teljesen strukturált adatokat csak a séma ismeretében tudjuk beolvasni.)
- Különféle forrásból eredő adatok integrálása.
- Rugalmasabb adatstruktúrák alkalmazása.

A relációs modell megközelítése

A relációs modell külön kezeli az adatszerkezet leírását és magukat az adatokat.

Relációséma:

Vásárlás (eladó, vevő, egységár, mennyiség)

Adatok:

Kovács	Tóth	2120	18
Kiss	Nagy	150	4500
Szabó	Kiss	980	3

Önleíró adatstruktúra

Tétel = (Eladó = ’Kovács’, Vevő = ’Tóth’, Egységár = 2120,
Mennyiség = 18)

Tétel = (Eladó = ’Kiss’, Vevő = ’Nagy’, Egységár = 150,
Mennyiség = 4500)

Tétel = (Eladó = ’Szabó’, Vevő = ’Kiss’, Egységár = 980,
Mennyiség = 3)

HTML-szerű leírás (XML)

```
<Vásárlások>
  <Tétel>
    <Eladó> Kovács </Eladó>
    <Vevő> Tóth </Vevő>
    <Egységár> 2120 </Egységár>
    <Mennyiség> 18 </Mennyiség>
  </Tétel>
  <Tétel>
    <Eladó> Kiss </Eladó>
    <Vevő> Nagy </Vevő>
    <Egységár> 150 </Egységár>
    <Mennyiség> 4500 </Mennyiség>
  </Tétel>
</Vásárlások>
```

XML

HTML (HyperText Markup Language)

XML (Extensible Markup Language):
a HTML általánosításának tekinthető. Az XML szabvány fejlesztője: W3C = World Wide Web Consortium
http://www.w3.org/

SGML (Standard Generalized Markup Language):
metanyelv, amely adat- és dokumentumcsere céljára szolgáló nyelvek (pl. HTML) definiálására alkalmas.
Az SGML szabványt 1988-ban publikálták.
Az XML az SGML leszűkített változata.

Az XML elemei

XML-deklaráció: `<? xml version="1.0" ... ?>`

Kezdőcímke: `<címkenév>`

Zárócímke: `</címkenév>`

A címkenévben kis- és nagybetű különböző!

Elem: kezdő- és zárócímkepár által leírt egység.

Attribútum: `<címkenév attr="érték">`

Kommentár: `<!-- ... -->`

Hierarchikus felépítés: minden elemnek egy szülője és több gyermeke lehet.

```
<? xml version="1.0" ?>
<könyvtár>
  <könyv>
    <szerző> Jókai </szerző>
    <cím> Aranyember </cím>
  </könyv>
  <könyv>
    <szerző> Sályi </szerző>
    <szerző> Szelezsán </szerző>
    <cím> Adatbázisok </cím>
  </könyv>
  <olvasó>
    <név> Nagy László </név>
    <lakcím>
      <város> Pécs </város>
      <utca> Kő u. </utca>
      <házszám> 35 </házszám>
    </lakcím>
  </olvasó>
</könyvtár>
```

Példa: könyvtári nyilván-tartás

Szintaktikai szabályok

A jól formált (well-formed) XML szabályai:

- Minden kezdőcímkéhez tartozik zárócímke.
- Az elemek nem átfedőek.
- Csak egy gyökérelem van.
- Attribútumértékek idézőjelek között szerepelnek.
- Egy elemnek nincs két azonos nevű attribútuma.
- Címke belsejében nem szerepelhet megjegyzés vagy feldolgozásra vonatkozó utasítás.
- Elemben és attribútumban a < és & karakterek csak speciális kódolással (escape-szekvenciában) fordulhatnak elő.

Az XML-dokumentum két típusa

Önálló dokumentum:

```
<? xml version="1.0" standalone="yes" ?>
<címke>
   ...
</címke>
```

Helyes (valid) dokumentum:

szabályrendszer leírását (DTD) feltételezi.

A DTD fogalma

DTD = Document Type Definition:
a dokumentum felépítési szabályainak leírása.

Elemdeklaráció a DTD-ben:

<!ELEMENT elemnév (komponensek)>

A DTD lehet

- külön fájlban,
- az XML dokumentum elején.

Külön fájlban elhelyezett DTD

Az XML fájl:

```
<? xml version="1.0" standalone="no" ?>
<!DOCTYPE minta SYSTEM "minta.dtd">
<címke>
   ...
</címke>
```

A minta.dtd fájl:

```
<? xml version="1.0" ?>
<!ELEMENT elemnév (komponensek)>
   ...   ...
<!ELEMENT elemnév (komponensek)>
```

XML fájlban elhelyezett DTD

```
<? xml version="1.0" standalone="yes" ?>
<!DOCTYPE gyökércímke [
  <!ELEMENT elemnév (komponensek)>
    ...
  <!ELEMENT elemnév (komponensek)>
]>
<címke>
  ...
</címke>
```

```
<? xml version="1.0" ?>
<könyvtár>
  <könyv>
    <szerző> Jókai </szerző>
    <cím> Aranyember </cím>
  </könyv>
  <könyv>
    <szerző> Sályi </szerző>
    <szerző> Szelezsán </szerző>
    <cím> Adatbázisok </cím>
  </könyv>
  <olvasó>
    <név> Nagy László </név>
    <lakcím>
      <város> Pécs </város>
      <utca> Kő u. </utca>
      <házszám> 35 </házszám>
    </lakcím>
  </olvasó>
</könyvtár>
```

Milyen DTD-vel írjuk le a könyvtári nyilvántartás szabályait?

Példa: könyvtári nyilvántartás DTD-je

```
<? xml version="1.0" ?>
<!ELEMENT könyvtár (könyv*, olvasó*)>
<!ELEMENT könyv (szerző+, cím)>
<!ELEMENT szerző (#PCDATA)>
<!ELEMENT cím (#PCDATA)>
<!ELEMENT olvasó (név, lakcím)>
<!ELEMENT név (#PCDATA)>
<!ELEMENT lakcím (város, utca, házszám)>
<!ELEMENT város (#PCDATA)>
<!ELEMENT utca (#PCDATA)>
<!ELEMENT házszám (#PCDATA)>
```

Jelölések

#PCDATA: szövegkonstans (Parsed Character Data, parse = elemez)

***** jelentése "nulla vagy több", tetszőleges számú ismétlés, beleértve a nullaszorost is.

+ jelentése "egy vagy több", tetszőleges számú ismétlés, de legalább egy.

? jelentése "nulla vagy egy".

| jelentése: kizáró vagylagos kapcsolat, például **`(#PCDATA | (város, utca, házszám))`**

A féligstrukturált adatmodell

Féligstrukturált adatmodell (semistructured data model):

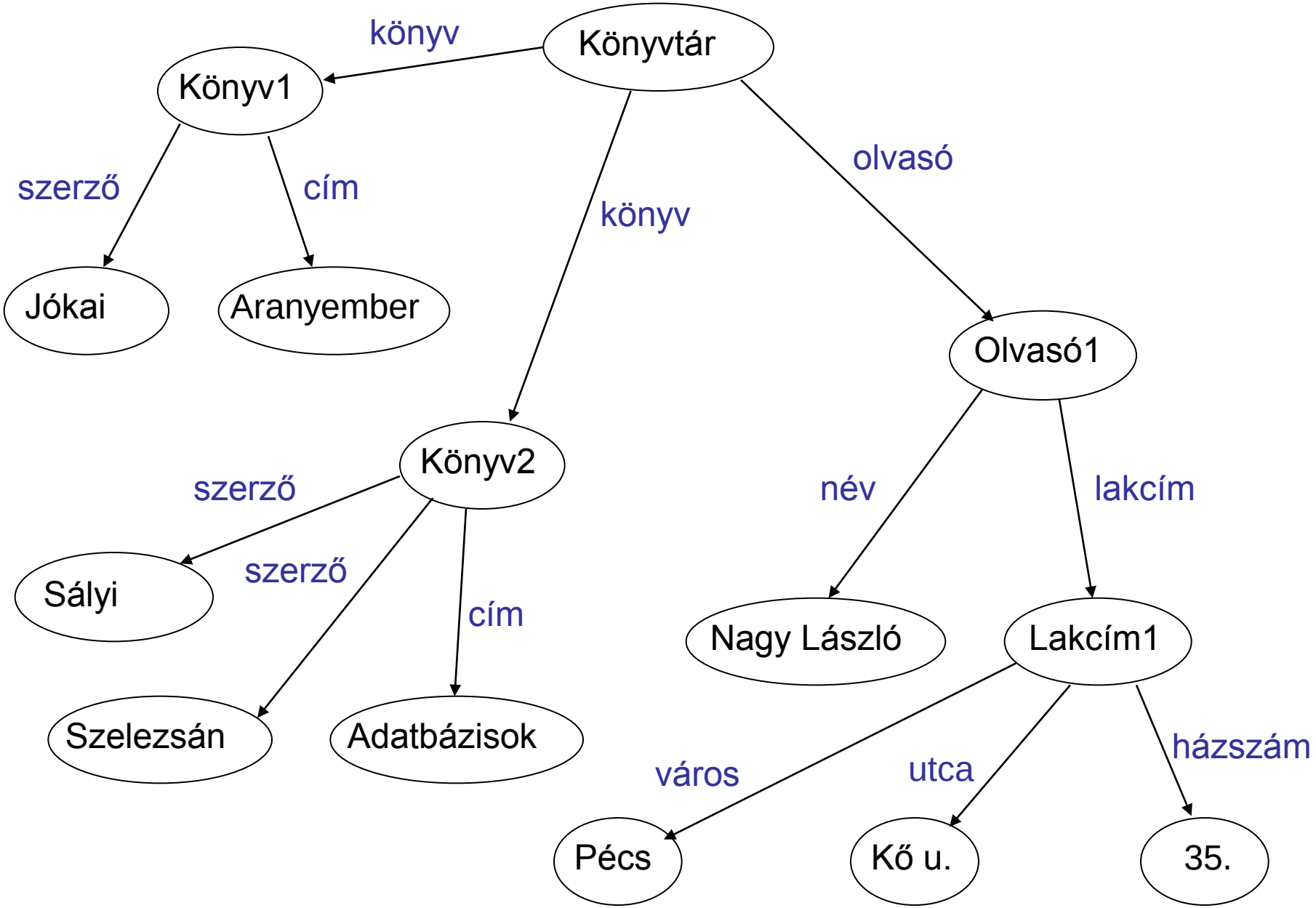
- Önleíró adatstruktúra (nincs külön séma és tartalom).
- Az XML elméleti alapjának tekinthető.

Grafikus ábrázolás: címkézett gráf

- ***Csomópont:*** adatpéldány, XML-elemnek felel meg (levélen atomi adat, közbülső csomópontban összetett adat)
- ***Él:*** adatpéldányok közötti viszony (tartalmazás vagy kapcsolat)

```
<? xml version="1.0" ?>
<könyvtár>
  <könyv>
    <szerző> Jókai </szerző>
    <cím> Aranyember </cím>
  </könyv>
  <könyv>
    <szerző> Sályi </szerző>
    <szerző> Szelezsán </szerző>
    <cím> Adatbázisok </cím>
  </könyv>
  <olvasó>
    <név> Nagy László </név>
    <lakcím>
      <város> Pécs </város>
      <utca> Kő u. </utca>
      <házszám> 35 </házszám>
    </lakcím>
  </olvasó>
</könyvtár>
```

Példa: könyvtári nyilván-tartás

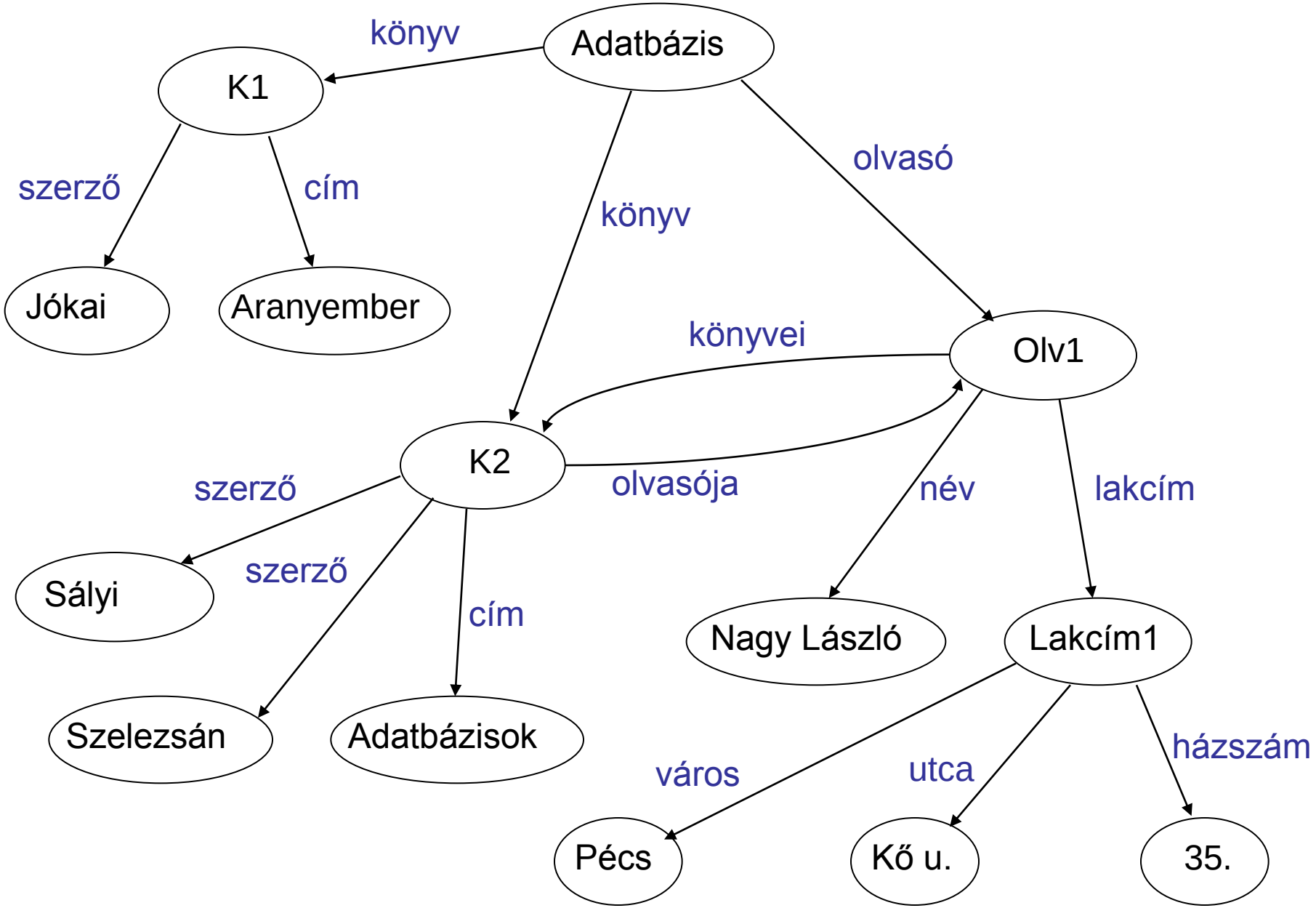

Könyvtár
könyv
Könyv1
szerző
cím
Jókai
Aranyember
könyv
olvasó
Olvasó1
Könyv2
szerző
szerző
cím
Sályi
Szelezsán
Adatbázisok
név
lakcím
Nagy László
Lakcím1
város
utca
házszám
Pécs
Kő u.
35.

Kapcsolatok kezelése

Probléma: a kölcsönzést (könyv – olvasó kapcsolat) hogyan kezeljük?

Megoldás féligstrukturált modellben: nyilak.

Megoldás XML-ben: attribútumok.


Adatbázis
könyv
K1
szerző
cím
Jókai
Aranyember
könyv
olvasó
könyvei
Olv1
K2
olvasója
szerző
Sályi
szerző
cím
Szelezsán
Adatbázisok
név
lakcím
Nagy László
Lakcím1
város
utca
házszám
Pécs
Kő u.
35.

```
<? xml version="1.0" ?>
<könyvtár>
  <könyv könyvId="K1">
    <szerző> Jókai </szerző>
    <cím> Aranyember </cím>
  </könyv>
  <könyv könyvId="K2" olvasója="Olv1">
    <szerző> Sályi </szerző>
    <szerző> Szelezsán </szerző>
    <cím> Adatbázisok </cím>
  </könyv>
  <olvasó olvasóId="Olv1" könyve="K2">
    <név> Nagy László </név>
    <lakcím>
      <város> Pécs </város>
      <utca> Kő u. </utca>
      <házszám> 35 </házszám>
    </lakcím>
  </olvasó>
</könyvtár>
```

Példa: könyvtári nyilván-tartás kölcsön-zéssel

(1:1 kapcsolat)

```
<? xml version="1.0" ?>
<könyvtár>
  <könyv könyId="K1" olvasója="Olv1">
    <szerző> Jókai </szerző>
    <cím> Aranyember </cím>
  </könyv>
  <könyv könyId="K2" olvasója="Olv1">
    <szerző> Sályi </szerző>
    <szerző> Szelezsán </szerző>
    <cím> Adatbázisok </cím>
  </könyv>
  <olvasó olvasóId="Olv1" könyvei="K1, K2">
    <név> Nagy László </név>
    <lakcím>
      <város> Pécs </város>
      <utca> Kő u. </utca>
      <házszám> 35 </házszám>
    </lakcím>
  </olvasó>
</könyvtár>
```

Példa 1:N kapcsolatra

N:M kapcsolat hasonlóan

Attribútumok megadása DTD-ben

ATTLIST elemnév attribútumdefiníciók

Attribútumdefiníció: attribútumnév típus alapértelmezés

Típusok:

- CDATA: karakteres adat.
- ID: egyértelmű elemazonosító.
- IDREF vagy IDREFS: hivatkozás azonosítóra.

Alapértelmezés:

- Alapértelmezett érték megadása, például "0".
- #REQUIRED: az attribútum megadása kötelező.
- #IMPLIED: az attribútum megadása nem kötelező.

Példa: könyvtári DTD attribútumokkal

```
<? xml version="1.0" ?>
<!ELEMENT könyvtár (könyv*, olvasó*)>
<!ELEMENT könyv (szerző+, cím)>
<!ATTLIST könyv könyvId ID #REQUIRED
                olvasója IDREF #IMPLIED>
<!ELEMENT szerző (#PCDATA)>
<!ELEMENT cím (#PCDATA)>
<!ELEMENT olvasó (név, lakcím)>
<!ATTLIST olvasó olvasóId ID #REQUIRED
                 könyvei IDREFS #IMPLIED>
<!ELEMENT név (#PCDATA)>
<!ELEMENT lakcím (város, utca, házszám)>
<!ELEMENT város (#PCDATA)>
<!ELEMENT utca (#PCDATA)>
<!ELEMENT házszám (#PCDATA)>
```

Az SQL-szabvány XML-támogatása

SQL:2003 szabvány: XML támogatás (SQL/XML)

Oracle 9i-ben jelent meg az XML támogatás (XML DB).

XMLType: XML adattípus, tábla oszlopához rendelhető.
XML-t kezelő metódusok használhatók.
Belül CLOB-ként tárolódik.

SQL/XML függvények

Az alábbi függvények az SQL-szabványban szerepelnek, és az Oracle is támogatja azokat.

Lényegében ***relációs adattábla → XML*** konverziót támogatnak.

- **XMLElement**: XML-elemet hoz létre.
- **XMLForest**: XML-elemek sorozatát hozza létre.
- **XMLAgg**: elemcsoportot tartalmazó XML-elemet hoz létre (SQL összesítő függvények mintájára).
- **XMLConcat**: XMLType típusú elemek sorozatából egyesített sorozatot hoz létre.
- **XMLSequence**: az XMLConcat inverze.

Az XMLElement függvény

A létrehozandó elem:

<elemnév attr1="érték1" attr2="érték2" ... >

tartalom

</elemnév>

XML-elem létrehozása:

XMLELEMENT (elemnév

[, XMLATTRIBUTES (érték1 AS attr1, érték2 AS attr2, ...)]

[, tartalom]])

Példák XMLElement használatára

1. példa:

```
SELECT XMLELEMENT(proba,'szöveg') AS p
  FROM dual;
```

Eredmény:

```
<PROBA>szöveg</PROBA>
```

2. példa (kisbetű-nagybetű megkülönböztetése)*:*

```
SELECT XMLELEMENT("Próba",' szöveg ') AS p
  FROM dual;
```

Eredmény:

```
<Próba> szöveg </Próba>
```

Példák XMLElement használatára

3. példa:

```
SELECT XMLELEMENT(proba,
  XMLATTRIBUTES('ertek1' as "attr1"),
  'szöveg') AS p FROM dual;
```

Eredmény:

```
<PROBA attr1="ertek1">szöveg</PROBA>
```

Példák XMLElement használatára

4. példa: Könyv (könyvszám, szerző, cím)

```
SELECT XMLELEMENT(konyv,
  XMLATTRIBUTES(k.konyvszam AS "konyvszam"),
  k.szerzo ||': '||k.cim) AS result
FROM konyv k;
```

Eredmény:

```
<KONYV konyvszam="1121">
  Sályi: Adatbázisok</KONYV>
<KONYV konyvszam="3655">
  Radó: Világatlasz</KONYV>
<KONYV konyvszam="2276">
  Karinthy: Így írtok ti</KONYV>
<KONYV konyvszam="1782">
  Jókai: Aranyember</KONYV>
```

Példák XMLElement használatára

5. példa:

```
SELECT XMLELEMENT( konyv,
  XMLATTRIBUTES(k.konyvszam AS "ksz"),
  XMLELEMENT(szerzo, k.szerzo),
  XMLELEMENT(konyvcim, k.cim) ) AS result
FROM konyv k;
```

Eredmény:

```
<KONYV ksz="1121"><SZERZO>Sályi</SZERZO>
  <KONYVCIM>Adatbázisok</KONYVCIM></KONYV>
<KONYV ksz="3655"><SZERZO>Radó</SZERZO>
  <KONYVCIM>Világatlasz</KONYVCIM></KONYV>
<KONYV ksz="2276"><SZERZO>Karinthy</SZERZO>
  <KONYVCIM>Így írtok ti</KONYVCIM></KO
<KONYV ksz="1782"><SZERZO>Jókai</SZERZO>
  <KONYVCIM>Aranyember</KONYVCIM></KONYV>
```

Az XMLForest függvény

XML-elemsorozat létrehozása:

XMLFOREST (kifejezés1 [AS elemnév1],
kifejezés2 [AS elemnév2], ...)

A létrehozott elemek:

<elemnév1> érték1 </elemnév1>
<elemnév2> érték2 </elemnév2>
...

Megjegyzések:

- A "kifejezés1" aktuális értéke "érték1".
- Ha "kifejezés1" egy tábla oszlopneve, akkor az oszlopnév lesz az "elemnév".

Példák XMLForest használatára

1. példa:

```
SELECT XMLFOREST
  (konyvszam AS ksz, szerzo, cim) AS lista
FROM konyv k;
```

Eredmény:

```
<KSZ>1121</KSZ>
<SZERZO>Sályi</SZERZO>
<CIM>Adatbázisok</CIM>
<KSZ>3655</KSZ>
<SZERZO>Radó</SZERZO>
<CIM>Világatlasz</CIM>
...
```

Példák XMLForest használatára

2. példa:

```
SELECT XMLELEMENT( konyv,
  XMLATTRIBUTES(k.konyvszam AS "ksz"),
  XMLForest(szerzo, cim) ) AS lista
FROM konyv k;
```

Eredmény:

```
<KONYV ksz="1121"><SZERZO>Sályi</SZERZO>
  <CIM>Adatbázisok</CIM></KONYV>
<KONYV ksz="3655"><SZERZO>Radó</SZERZO>
  <CIM>Világatlasz</CIM></KONYV>
<KONYV ksz="2276"><SZERZO>Karinthy</SZERZO>
  <CIM>Így írtok ti</CIM></KONYV>
...
```

Az XMLAgg függvény

XML-aggregációs függvény:
XMLAGG (xml_elem [ORDER BY oszlop])

A létrehozott adatstruktúra:
A lekérdezett xml_elem-ek sorozatát egyetlen XML-elemként adja vissza.

...

Az SQL összesítő függvények (AVG, SUM,...) és GROUP BY mintájára működik.

Példák XMLAgg használatára

1. példa: Dolgozó (adószám, név, lakcím, osztálykód)

```
SELECT XMLELEMENT ( osztaly, XMLAGG(
  XMLELEMENT (dolgozo, d.nev||': '||d.lakcim)
  ORDER BY nev) ) AS lista
FROM dolgozo d WHERE osztalykod='3';
```

Eredmény:

```
<OSZTALY>
<DOLGOZO>Fekete: Pécs, Hegy u.5.</DOLGOZO>
<DOLGOZO>Kiss: Pápa, Kő tér 2.</DOLGOZO>
<DOLGOZO>Nagy: Pécs, Cső u.25.</DOLGOZO>
</OSZTALY>
```

Példák XMLAgg használatára

2. példa:

```
SELECT XMLELEMENT(
  osztaly,
  XMLATTRIBUTES(osztalykod AS osztkod),
  XMLAGG(
  XMLELEMENT(dolgozo, d.nev||': '||d.lakcim))
  ) AS lista
FROM dolgozo d GROUP BY osztalykod;
```

Példák XMLAgg használatára

2. példa eredménye:

```
<OSZTALY OSZTKOD="1">
  <DOLGOZO>Tóth: Tata, Tó u.2.</DOLGOZO>
  <DOLGOZO>Kovács: Vác, Róka u.1.</DOLGOZO>
  <DOLGOZO>Takács: Győr, Pap u.7.</DOLGOZO>
</OSZTALY>
<OSZTALY OSZTKOD="2">
  <DOLGOZO>Kovács: Pécs, Vár u.5.</DOLGOZO>
  <DOLGOZO>Török: Pécs, Sas u.8.</DOLGOZO>
</OSZTALY>
<OSZTALY OSZTKOD="3">
  <DOLGOZO>Kiss: Pápa, Ko tér 2.</DOLGOZO>
  <DOLGOZO>Fekete: Pécs, Hegy u.5.</DOLGOZO>
  <DOLGOZO>Nagy: Pécs, Cső u.25.</DOLGOZO>
</OSZTALY>
```