Adatkezelési műveletek az XML nyelvben

– XQuery - XML dokumentumok lekérdezésére szolgáló lekérdezőnyelv
– kifejező ereje az OQL nyelvével egyenértékű.
– más nyelvek: XML-QL, XQL, Lorel, de jelenleg valószínű, hogy az XQuery lesz a szabványos,
– teljes specifikációja a http://www.w3.org/TR/xquery/
– az XQuery több korábbi nyelvre alapul, az XPath elérési-út kifejezéseit használja az XML dokumentum részeinek a visszaküldésére. XQuery magába foglalja a teljes XPath-t.

– az XQuery csak lekérdezésre ad lehetőséget.
– adatmódosításra nincs lehetőség egyelőre csak a DOM (Document Object Model) segítségével.
– fejlesztés alatt van egy XUpdate nevű nyelv, mely valószínű szabvány lesz.
– az XQuery nem XML dokumentumokkal dolgozik, mivel az XML dokumentumok karaktersorok, úgy voltak kitalálva, hogy az ember számára olvashatók legyenek.
– az XQuery az XML dokumentumok egy átalakított, absztakt formájával dolgozik.
– egy XML dokumentum absztrakt formája az XQuery adatmodell egy példánya.

Az XPath nyelv

– Az XPath lehetőséget ad egy XML dokumentumban való navigációra, figyelembe véve a félig-struktúrált adat gráfként való ábrázolását és az útakat a gráfban.

– Az XQuery nyelv, az XSLT és az XPointer is használja az XPath által nyújtott lehetőségeket.

– Az XQuery adatmodellben (hasonlóan az XPath esetében) egy XML dokumentum egy fa szerkezettel van ábrázolva.

– Minden csomópontnak van egy azonósítója, mely megkülönbözteti a többi csomóponttól, még ha teljesen egyformák is.

Az XPath adatmodell

- *adatok szekvenciája*.

Egy *adat* a következők egyike:

- dokumentum (document)
- csomópont (element)
- szöveges (text)
- tulajdonság (attribute)
- névterület (namespace)
- vezérlési utasítás (processing intruction)
- megjegyzés (comment)

a. *Dokumentumok.* Ezek XML dokumentumot tartalmazó fájlok, esetleg lokális elérési útvonalukkal vagy egy URL-lel jelölve.

b. *Elemek.* Ezek XML elemek, beleértve a nyító jelölőjüket, a megfelelő zárójelölőjüket és mindent, ami e kettő között szerepel. (pl. egy féligstruktúrált adat fáját, melyet egy XML dokumentum reprezentál.)

c. *Attribútumok*. Ezek a nyitó jelölőkben találhatók.

Példa.

```
10                                          //  egyszerű típusú adatok:
”tíz”                                             // egész szám
10.0                                              // szöveg
<Számrendszer bázis = ”8”>                        // valós szám
   <Digit>1</Digit>                         // csomópont, amelynek
   <Digit>2</Digit>                         // típusa „elem”
</Számrendszer>                             // 2 gyerekelem
@valt=”10”                                  // 1 attribútum
                                            // attribútum csomópont
```

Dokumentum csomópontok

- XPath által feldolgozható dokumentumok fájlok
- minden XPath lekérdezés egy dokumentumra hivatkozik
- dokumentum csomópontot készíteni egy fájlból:

  doc(*fájlnév*)
- a fájlnak XML dokumentumnak kell lennie.
- a fájlt megadhatjuk lokális névvel, illetve egy URL-lel is

Példa:

doc(”filmek.xml”)
doc(”/usr/sally/data/filmek.xml”)
doc(”infolab.stanford.edu/~hector/filmek.xml”)

Egy XPath kifejezés eredménye:

- csomópontok egy csoportja,
- logikai érték, szám vagy szöveg.

Függvények

doc() - egy URI (Universal Resource Identifier) által megadott dokumentumot ad vissza, pontosabban egy dokumentum csomópontot.

collection() - akkor használhatjuk, ha egy adatbázist akarunk azonosítani egy lekérdezéshez, mely csomópontok kollekcióját adja meg.

példa: Legyen a következő könyvészetet leíró példa, mely egy bibl.xml nevű állományban van tárolva.

```
<bibliográfia>
 <könyv év="1994">
  <cím>TCP/IP Illustrated</cím>
  <szerző>
     <vnév>Stevens</vnév>
     <knév>W.</knév>
   </szerző>
   <kiadó>Addison-Wesley</kiadó>
   <ár>65.95</ár>
 </könyv>
 <könyv év="1992">
  <cím>Advanced Programming in the
            UNIX Environment</cím>
```

```
  <szerző>
    <vnév>Stevens</vnév>
    <knév>W.</knév>
  </szerző>
  <kiadó>Addison-Wesley</kiadó>
  <ár>65.95</ár>
</könyv>
<könyv év="2000">
  <cím>Data on the Web</cím>
  <szerző>
    <vnév>Abiteboul</vnév>
    <knév>Serge</knév>
  </szerző>
```

```
  <szerző>
    <vnév>Buneman</vnév
    <knév>Peter</knév>
  </szerző>
  <szerző>
    <vnév>Suciu</vnév>
    <knév>Dan</knév>
  </szerző>
  <kiadó>Morgan Kaufmann</kiadó>
  <ár>39.95</ár>
</könyv>
```

```
<könyv év="1999">
  <cím>The Economics of Technology and
           Content for Digital TV</cím>
  <szerkesztő>
    <vnév>Gerbarg</vnév>
    <knév>Darcy</knév>
    <társulat>CITI</társulat>
  </szerkesztő>
  <kiadó>Kluwer Academic
              Publishers</kiadó>
  <ár>129.95</ár>
 </könyv>
</bibliográfia>
```

```
<!ELEMENT bibliográfia (könyv* )>
<!ELEMENT könyv (cím, (szerző+ |
          szerkesztő+ ), kiadó, ár)>
<!ATTLIST könyv év CDATA #REQUIRED >
<!ELEMENT szerző (vnév, knév)>
<!ELEMENT szerkesztő (vnév, knév,
            társulat)>
<!ELEMENT cím (#PCDATA )>
<!ELEMENT vnév (#PCDATA )>
<!ELEMENT knév (#PCDATA )>
<!ELEMENT társulat (#PCDATA )>
<!ELEMENT kiadó (#PCDATA )>
<!ELEMENT ár (#PCDATA )>
```

doc("bibl.xml")- megadja az egész XMLdokumentumot

XPath elérési-út kifejezései (path expressions)

– az elérési út egy XML dokumentum adott csomópontjához vezető út megadására szolgál
– tipikusan egy dokumentum gyökerével kezdődik és egy szekvenciát tartalmaz, amely jelölőkből és törtvonalakból (/) áll
– $/T_1/T_2/\ldots/T_n$.
– kiértékeljük ezt a kifejezést onnan kezdve, hogy az adatok szekvenciája egyetlen csomópontból áll: a dokumentumból.
– ezt követően feldolgozzuk minden egyes $T_1, T_2, \ldots$-t.

– egy T_i feldolgozása figyelembe veszi a szekvenciában az előző adatok feldolgozása során kapott eredményeket, amennyiben vannak ilyenek
– minden lépés kiértékelése eredményeként csomópontok sorozatát kapjuk.

Relatív útkifejezések

- az aktuális csomóponthoz vagy csomópontok szekvenciájához.
- a relatív kifejezések nem kezdődhetnek / jellel

```
<xs:keyref name = "filmRef" refers = "FilmKulcs">
    <xs:selector xpath = "Színész/szerepelBenne" />
    <xs:filed xpath = "@FilmCím" />
    <xs:filed xpath = "@év" />
</xs:keyref>
```

példa:

```
doc("bibl.xml")/bibliográfia/könyv
```

– a doc függvény megnyitja a bibl.xml állományt,
– /bibliográfia szelektálja a dokumentum gyökér elemét
– /könyv pedig a könyv elemeket.

példa: A

```
doc("bibl.xml")//könyv
```

– elérési-út kifejezés eredménye azonos az előző példa által adott eredménnyel.
– könyv elemeket ad vissza az adott dokumentumból, arról a szintről, ahol épp könyv elemek találhatók.

Egy XPath lépés általános formája a következő:

```
irány::csomópont_típus
[feltételes_kifejezés]
```

Irányok

Az irány, az aktuális elem és a lépéssel megjelölt csomópontok viszonyát írja le. Lehetséges irányok:

- child:: az aktuális elem gyermek eleme lesz. Ha nincs megadva irány, ez az alapértelmezett.

Pl. child::könyv, az aktuális elem könyv gyerek elemeit adja.

- descendant:: az aktuális elem összes leszármazottja (gyermekei és azoknak is a gyermekei rekurzívan).

Pl. descendant::szerző azon szerző elemeket adja, melyek az aktuális elem leszármazottai.

- parent:: az aktuális elem szülő elemét jelöli.

Pl. parent::könyv kiválasztja az aktuális elem szülő elemét, ha az egy könyv elem.

- self:: az aktuális elemet jelöli.

Pl. self::könyv az aktuális elemet adja, ha az egy könyv elem.

- descendant-or-self:: az aktuális elemet és a leszármazottait jelöli.
- ancestor:: az aktuális elem összes ősét jelöli.
- ancestor-or-self:: az aktuális elemet és az összes ősét jelöli.

- following-sibling:: az aktuális elem következő testvérét jelöli.
- preceding-sibling:: az aktuális elem előző testvérét jelöli.
- following:: az aktuális elemet követő minden elem a dokumentum sorrendjében.
- preceding:: az aktuális elemet megelőző összes elem a dokumentum sorrendjében.
- attribute:: az aktuális elem tulajdonságait jelöli.

Pl. attribute::könyv, könyv tulajdonságait adja meg az aktuális elemnek.

- namespace:: az aktuális elem névtereit jelöli ki.

Rövidítések

A leggyakrabban használt XPath kifejezéseket rövidített formában is lehet írni, ezek a következők:

(semmi) - child::
@ - attributes::
. (pont) - self::node()
.. (két pont) - parent::node()
// - descendant-or-self::node()

példa: Az eddigi példákban is a child irány elmaradt. A

```
doc("bibl.xml")/bibliográfia/könyv
```

kifejezés bővebben:

```
doc("bibl.xml")/child::bibliográfia/child::
könyv.
```

Attribútumok az útkifejezésekben

- az útvonal kifejezést egy attribútum nevével kell zárni, melyet megelőz egy @ jel

Példa: `doc("bibl.xml")/bibliográfia/könyv/`
`attribute::év`

rövidebben:

`doc("bibl.xml")/bibliográfia/könyv/@év`

Típusok

A XPath 7 féle csomópont típust különböztet meg.
három alaptípus:

- névterület
- tulajdonság
- csomópont

a többi a csomópont alaptípus speciális esete.

- szöveges
- megjegyzés
- gyökér
- vezérlési utasítás

XPath-ban minden iránynak van egy elsődleges típusa:

- ha az aktuális irány tulajdonság, akkor az elsődleges típus is tulajdonság
- ha az aktuális irány névterület, akkor az elsődleges típus is névterület
- minden más esetben az elsődleges típus csomópont.

Egy típust kétféleképpen lehet megadni:

- névterület megadásával
- csomópont típust kiválasztó utasítások segítségével.

Csomópont típusok:

- "név" – A "név" által megadott nevű tulajdonságot (attribute:: irány esetén) vagy elemeket választja ki.

Pl. child::könyv esetén a típus névvel volt megadva és az aktuális elem könyv gyerek elemeit adja meg.

- "*" – az adott iránynak megfelelő összes elemet (bármilyen nevű) kiválasztja.
- comment() – az adott iránynak megfelelő minden megjegyzés elemet kiválaszt.
- text() – az adott iránynak megfelelő minden szöveges elemet kiválaszt.
- processing-instruction() – az adott iránynak megfelelő minden vezérlési utasítást kiválaszt.

- node() – az adott iránynak megfelelő minden elemet, a típusától függetlenül, kiválaszt.

Pl. ha az aktuális egy könyv elem, a child::node() kiválasztja annak összes gyermek elemét, a típusától függetlenül.

Feltételek az útkifejezésekben

- az elérési-út kifejezés által szolgáltatott csomópontok egy részhalmazát választjuk ki.
  - a feltételes kifejezést szögletes zárójelben adjuk meg.
- összetett feltételt a feltételekkel és a köztük szereplő `or` vagy `and` operátorokkal készíthetünk.

példa: A következő elérési-út kifejezés azon szerző elemeket adja meg, melyeknek vezeték neve Stevens.

```
doc("bibl.xml")/bibliográfia/könyv/szerző
[vnév="Stevens"]
```

Egyéb feltétel:

- Egy [i] egész kizárólag a szülő i. gyerekére való alkalmazással igaz.
- Egy [T] jelölő kizárólag azokra az elemekre igaz, amelyek egy vagy több T jelölőjű gyerekelemmel rendelkeznek.
- Hasonlóan, egy [A] attribútum kizárólag azokra az elemekre igaz, amelyek rendelkeznek értékkel az A attribútumra.

Példa: A

```
doc("bibl.xml")/bibliográfia/könyv/szerző[1]
```

kifejezés minden könyv esetén az első szerzőt adja meg.

Példa: az egész dokumentum legelső szerzője:

```
(doc("bibl.xml")/bibliográfia/könyv/szerző)[1]
```

Példa:

```
1)  <? xml version=”1.0’” encoding=”utf-8”
          standalone=”yes” ?>
2)  <Filmek>
3)     <Film FilmCím = ”King Kong”>
4)        <Verzió év = ”1933”>
5)           <Színész>Fay Wray</Színész>
6)        </Verzió>
7)        <Verzió év = ”1976”>
8)           <Színész>Jeff Bridges</Színész>
9)           <Színész>Jessica Lange</Színész>
10)       </Verzió>
```

```
11)          <Verzió év= ”2005” />
12)       </Film>
13)       <Film FilmCím = ”Gumiláb”>
14)          <Verzió év= ”1984”>
15)             <Színész>Kevin Bacon</Színész>
16)             <Színész>John Lithgow</Színész>
17)             <Színész>Sarah Jessica Parker</Színész>
18)          </Verzió>
19)       <Film>
20)   </Filmek>
```

```
/Filmek/Film/Verzió[1]/@év
```

Ez az összes film első verziójának az elkészítési évét kérdezi és eredménye az ”1933” ”1984” szekvencia.

példa. `/Filmek/Film/Verzió[Színész]`
- három `Verzió` elemet szolgáltat.
- `[Színész]`, a „van legalább egy Színész gyerekeleme” feltételt interpretálja.

Tulajdonság értékére is vonatkozhat feltétel.
– a tulajdonság nevét a @ kell megelőzze.

példa: A

```
doc("bibl.xml")/bibliográfia/könyv/[@év=2000]
```

Vagy:

```
doc("bibl.xml")/bibliográfia/*/[@év=2000]
```

a * bármely elem nevét helyettesíti.

Az XQuery lekérdező nyelv

– az XPath nyelv nem elegendő az XML dokumentumok lekérdezésére, csak egy címzési lehetőség.
– az XQuery az XPath elérési-út kifejezéseit használja az XML dokumentum részeinek a visszatérítésére.
– a FLWOR kifejezések az SQL nyelv SELECT parancs helyét veszik át.
– dokumentum, elem és tulajdonság konstruktorok segítik új XML adat létrehozását.

Dokumentum, elem és tulajdonság konstruktorok

XQuery-ban lehetőségünk van XML dokumentum létrehozására:

```
document{}
```

kifejezés segítségével, mely a fenti formában egy üres dokumentumot hoz létre.

Elemeket, tulajdonságokat, megjegyzéseket XML szintaxissal hozhatunk létre.

példa: A következőkben egy dokumentum csomópontot hozunk létre, mely megjegyzést és elemeket tartalmaz.

```
document {
<?xml version="1.0" encoding="WINDOWS-1250" ?>
<!—Egy jó adatbázis könyv! —>
  <könyv év="2000">
   <cím>Database System Implementation</cím>
   <szerző>
      <vnév>Garcia-Molina</vnév>
      <knév>Hector</knév>
   </szerző>
   <szerző>
      <vnév>Ullman</vnév
      <knév>Jeffrey D.</knév>
   </szerző>
   <szerző>
```

```
        <vnév>Widom</vnév>
        <knév>Jennifer</knév>
    </szerző>
    <kiadó>Prentice Hall</kiadó>
    <ár>45</ár>
</könyv>
}
```

– Elem konstruktorban egy elem vagy egy tulajdonság értékének a generálására kapcsos zárójeleket használunk
– „{” és „}”, ún. „enclosed” kifejezést határolnak.
– A zárójelek közé egy XPath kifejezést, illetve összesítő függvényeket helyezhetünk.

```
<példa>
  <p> Ez egy lekérdezés. </p>
  <eg> doc("bibl.xml")//könyv[1]/cím </eg>
  <p> Itt a fenti lekérdezés eredménye.</p>
  <eg>{ doc("bibl.xml")//könyv[1]/cím }</eg>
</példa>
```

Az eredménye a fenti lekérdezésnek:

```
<példa>
  <p> Ez egy lekérdezés. </p>
  <eg> doc("bibl.xml")//könyv[1]/cím </eg>
  <p> Itt a fenti lekérdezés eredménye.</p>
  <eg><cím>TCP/IP Illustrated</cím></eg>
</példa>
```

```
doc("bibl.xml")//könyv[1]/cím
```

XPath kifejezés - kiértékeli a rendszer és eredménye, mely egy cím elem, a kapcsos zárójelek között levő kifejezés helyére kerül.

– az enclosed kifejezés esetén elemet bevezető és elemet záró tag generálódik.

– „enclosed” kifejezések segítségével létező XML értékeket új szerkezetben adhatunk meg.

```
<címek szám="{ count(doc('bibl.xml')//cím) }">
{
    doc("bibl.xml")//cím
   }
</címek>
```

A lekérdezés eredménye:

```
<címek szám = "4">
 <cím>TCP/IP Illustrated</cím>
 <cím>Advanced Programming in the Unix
         Environment</cím>
 <cím>Data on the Web</cím>
 <cím>The Economics of Technology and Content
          for Digital TV</cím>
</címek>
```

– tulajdonság értékét dupla idézőjel közé kell helyezni és ezért az XML állomány nevét egyszeres idézőjel közé tesszük.

Az XQuery egy másik lehetőséget is ajánl elemek és tulajdonságok létrehozására, az `element` és `attribute` kulcsszó segítségével.

példa: egy könyv elemet hoz létre az ismert szerkezettel.

```
element könyv
{
   attribute év { 2004 },
   element cím { "An Introduction to Database
                  Systems" },
   element szerző
   {
      element vnév { "Date" },
      element knév { "Chris J." }
   },
   element kiadó { "Addison Wesley Higher
                    Education " },
   element ár{ 59.95 }
}
```

FLWOR kifejezések

– Az SQL nyelv SELECT parancsához hasonlít, de nem táblákra, sorokra, oszlopokra vonatkozik, hanem **for** és **let** kulcsszó segítségével változót köt értékekhez.

– Az XQuery egy *funkcionális nyelv*, ami magában foglalja, hogy tetszőleges XQuery kifejezést használhatunk bármilyen olyan helyen, ahol az értelmezhető.

– A funkcionális tulajdonság egy kétélű fegyver. Megköveteli azt, hogy az XQuery minden operátora értelmet kapjon akkor is, amikor egynél több adatra alkalmazzuk, melynek váratlan következményei lehetnek.

– A FLWOR kifejezés neve rövidítésekből adódik:

for kulcsszó, mely egy vagy több változóhoz kifejezéseket rendel, értékpárokat létrehozva.

Általános formája:

```
for <változó_1> in <elérési-út kifejezés_1>
   ⋮
    <változó_k> in <elérési-út kifejezés_k>
```

– a változó rendre felveszi az elérési-út kifejezés minden értékét, melyek csomópontok az XML dokumentumból.

– minden ami a **for** után következik a változó összes értékére végrehajtásra kerül.

– $ jel előzi meg a változókat

let kulcsszó, mely abban különbözik a **for**-tól, hogy a változó nem egyenként veszi fel a hozzárendelt kifejezés összes értékét, hanem a kifejezés összes értékét egyszerre rendeli a változóhoz.

- Használható a **for** kulcsszóval együtt vagy nélküle is.
- Ha nincs **for** a kifejezésben, akkor a változó csak egy értéket vesz fel, mely valójában a hozzárendelt kifejezés összes értékének a sorozata;

where kulcsszó, melynek segítségével egy szűrő feltételt tehetünk a **for** és **let** által létrehozott értékpárokra;

order by kulcsszó, mely az értékpárokat adott sorrendbe rendezi;

return kucsszó, melynek segítségével az eredmény értékpárt létre tudjuk hozni.

példa: A gyakorlatban nagyrészt elérési-út kifejezést használunk, de az XPath megengedi a következő kifejezést is:

```
for $i in (1, 2, 3)
return <eredmény><i>{ $i }</i></eredmény>
```

A lekérdezés eredménye:

```
<eredmény><i>1</i></eredmény>
<eredmény><i>2</i></eredmény>
<eredmény><i>3</i></eredmény>
```

példa: A fentihez hasonló példában használjuk a **`for`** helyett a **`let`**-et. Figyeljük meg, hogy az `$i` változó egyszerre felveszi az (1, 2, 3) halmaz összes értékét.

```
let $i in (1, 2, 3)
return <eredmény><i>{ $i }</i></eredmény>
```

A lekérdezés eredménye:

```
<eredmény><i>1 2 3</i></eredmény>
```

```
for $i in (1, 2, 3)
let $j := (1, 2, 3)
return
<eredmény><i>{ $i }</i><j>{ $j }</j></eredmény>
```

A lekérdezés eredménye:

```
<eredmény><i>1</i><j>1 2 3</j></eredmény>
<eredmény><i>2</i><j>1 2 3</j></eredmény>
<eredmény><i>3</i><j>1 2 3</j></eredmény>
```

```
for $i in (1, 2, 3),
    $j in (4, 5, 6)
return
 <ered><i>{ $i }</i><j>{ $j }</j></ered>
```

Eredmény:

```
<ered><i>1</i><j>4</j></ered>
<ered><i>1</i><j>5</j></ered>
<ered><i>1</i><j>6</j></ered>
<ered><i>2</i><j>4</j></ered>
<ered><i>2</i><j>5</j></ered>
<ered><i>2</i><j>6</j></ered>
<ered><i>3</i><j>4</j></ered>
<ered><i>3</i><j>5</j></ered>
<ered><i>3</i><j>6</j></ered>
```

Pszeudokódban:

```
for i = 1 to 3 do
   for j = 4 to 6 do
      eredmény {i, j}
   endfor
endfor
```

Lássunk olyan lekérdezéseket, ahol a változó egy elérési-út kifejezés különböző értékeit veszi fel.

- Egy FLWR kifejezés eredménye – hasonlóan bármely XQuery kifejezéshez – adatok szekvenciája.
- Az adatok szekvenciáját a return-záradékhoz csatolt kifejezés állítja elő az addig előállított adatok szekvenciájából

példa: A `$b` változó értékei rendre a `bibl.xml` dokumentumban található könyv elemek.

```
for $b in doc("bibl.xml")//könyv
where $b/ár < 50
return $b/cím
```

A lekérdezés eredménye tartalmazza az 50 $-nál olcsóbb könyveket:

```
<cím>Data on the Web</cím>
```

Változók helyettesítése értékeikkel

- jelölők között, illetve egy attribútum értékeként bármilyen szöveg megengedett
- a jelölők belsejében, a szöveget kapcsos zárójelekkel kell körülvennünk.

Példa: Adjuk meg a `Film` elemek szekvenciáját úgy, hogy ezek mindegyike tartalmazza egy adott című film minden színészét, tekintet nélkül arra, hogy melyik verzióban szerepeltek.

```
let $filmek := doc(”filmek.xml”)
for $m in $filmek/Filmek/Film
return <Film FilmCím =
{$m/@FilmCím}>{$m/Verzió/Színész}</Film>
```

Példa: `Színész` elemek szekvenciája egy `Színészek` elemben legyen

```
let $Színészekeq := (
   let $filmek := doc(”filmek.xml”)
   for $m in $filmek/Filmek/Film
   return $m/Verzió/Színész
)
return <Színészek>{$Színészekeq}</Színészek>
```

Az XQuery összehasonlító operátorai

Egy hibás kísérlet arra, hogy megtaláljuk azokat, akik 123 Maple St., Malibu címen laknak

```
let $színészek := doc(”színészek.xml”)
for $s in $színészek/Színészek/Színész
where $s/Lakcím/Utca = ”123 Mapple St.” and
   $s/Lakcím/Város = ”Malibu”
return $s/Név
```

eredmény: <Név>Carrie Fisher</Név>

Az XQuery összehasonlító operátorok egy olyan csoportját biztosítja, amelyekkel csak olyan szekvenciákat tudnak összehasonlítani, amelyeknek egyedi adatuk van, és elbuknak, ha valamelyik operandus több elemből álló szekvencia. Ezek az operátorok az összehasonlítások két betűs rövidítései: `eq`, `ne`, `lt`, `gt`, `le` és `ge`

```
let $színészek := doc(”színészek.xml”)
for $s in $színészek/Színészek/Színész
where $s/Lakcím/Utca eq ”123 Maple St.” and
   $s/Lakcím/Város eq ”Malibu”
return $s/Név
```

nem szolgáltat eredményt

```
let $színészek := doc(”színészek.xml”)
for $s in $színészek/Színészek/Színész
    $l in $s/Lakcím
where $/Utca = ”123 Mapple St.” and
      $l/Város = ”Malibu”
return $s/Név

let $színészek := doc(”színészek.xml”)
for $s in
$színészek/Színészek/Színész[Lakcím/Város =
”Malibu”]
where $s/Lakcím/Utca = ”123 Mapple St.”
return $s/Név
```

Az XQuery-ben is használhatunk összesítő függvényeket, mint a count, sum, avg, min és max.

példa:

```
for $b in doc("bibl.xml")//könyv
let $c := $b/szerző
return <könyv>{ $b/cím,
  <szszám>{ count($c) } </szszám>}</könyv>
```

- a count függvény nem elemet ad vissza, mint a `$b/cím`, ezért elemet bevezető és elemet záró tagot explicit meg kell adjuk.
- a lekérdezés eredménye minden könyv esetén tartalmazza a könyv címét és szerzőinek a számát:

```
<könyv>
  <cím>TCP/IP Illustrated</cím>
  <szszám>1</szszám>
</könyv>
<könyv>
  <cím>Advanced Programming in the UNIX
         Environment</cím>
  <szszám>1</szszám>
</könyv>
<könyv>
  <cím>Data on the Web</cím>
  <szszám>3</szszám>
</könyv>
<könyv>
```

```
  <cím>The Economics of Technology and Content
          for Digital TV</cím>
  <szszám>0</szszám>
</könyv>
```

Feltételben attribútum:

példa: Keressük a 2000-ben kiadott könyvek címét.

```
for $b in doc("bibl.xml")//könyv
where $b/@év = "2000"
return $b/cím
```

A lekérdezés eredménye:

```
<cím>Data on the Web</cím>
```

példa: Keressük azon könyveket, melyeknek több mint 2 szerzőjük van.

```
for $b in doc("bibl.xml")//könyv
let $c := $b//szerző
where count($c) > 2
return $b/cím
```

A lekérdezés eredménye:

```
<cím>Data on the Web</cím>
```

Eredmény rendezése

példa: A könyvek cím szerint ábécé sorrendben jelennek meg az eredményben:

```
for $c in doc("bibl.xml")//cím
order by $c
return $c
```

példa: Rendezzük a könyv elemeket az első szerző neve szerint!

```
for $b in doc("bibl.xml")//könyv
let $a1 := $b/szerző[1]
order by $a1/vnév, $a1/knév
return $b/cím
```

Az eredmény előállításában, a **`return`** kulcsszó után is használhatunk XQuery kifejezéseket.

példa: A következő példa árajánlatokat ad meg, elemkonstruktor segítségével.

```
for $b in doc("bibl.xml")//könyv
return <ajánlat> {$b/cím, $b/ár } </ajánlat>
```

A lekérdezés eredménye:

```
<ajánlat>
  <cím>TCP/IP Illustrated</cím>
  <price>65.95</price>
</ajánlat>
<ajánlat>
  <cím>Advanced Programming in the UNIX
        Environment</cím>
  <ár>65.95</ár>
```

```
</ajánlat>
<ajánlat>
  <cím>Data on the Web</cím>
  <ár>39.95</ár>
</ajánlat>
<ajánlat>
  <cím>The Economics of Technology and Content
         for Digital TV</cím>
  <ár>129.95</ár>
</ajánlat>
```

Az ismétlődések kiszűrése

A distinct-values() függvény egy csomópont sorozat esetén különböző értékek sorozatát adja meg, az azonos értékeket kiküszöböli.

példa: A könyvészetet tartalmazó XML dokumentum esetén, egy szerzőnek két könyve van. Ha a szerzőkre vagyunk kíváncsiak és Stevens-t csak egyszer akarjuk megadni, akkor a következő kifejezést használhatjuk:

```
distinct-values(doc("bibl.xml")//szerző/vnév)
```

A lekérdezés eredménye:

```
Stevens Abiteboul Buneman Suciu
```

Amint látjuk nem elemeket, csak azok értékeit adja meg a distinct-values.

példa: szerző elemeket a következőképpen kell megadni:

```
for $n in
  distinct-values(doc("bibl.xml")//szerző/vnév)
return <vnév>{ $n }</vnév>
```

A lekérdezés eredménye:

```
<vnév>Stevens</vnév>
<vnév>Abiteboul</vnév>
<vnév>Buneman</vnév>
<vnév>Suciu</vnév>
```

Az XQuery lehetőséget ad, hogy az IDREF típusú tulajdonságok által adott hivatkozást kövessük.

- *x* IDREF típusú tulajdonságok halmaza

- *x* => *y* kifejezés megadja azon objektumokat, melyek tag neve *y* és az ID tulajdonsága egyezik valamelyikkel az IDREF-ek közül.

példa: Tekintsük az első zenés CD-ket leíró DTD és XML adatokat és a következő lekérdezést: Keressük a rock stílusú albumok zeneszámait és előadójuk nevét!

```
for $a in doc("zene.xml")//Album
    $z in $a/@Zeneszámai=>Zeneszám
let $e := $a/@Előadója=>Előadó
where $a/Stílus = "rock"
return <rockZene>
{  $z/SzCíme, $e/ENév }
</rockZene>
```

- `$a` változó az Album elemeken fut végig, a szűrőfeltétel a rock zenét válogatja ki.
- Album elemnek Zeneszámai nevű IDREFS típusú tulajdonsága
- `$z` változó értéke rendre Albumon belül az összes Zeneszám elem.

- ezt úgy tudjuk elérni, ha a => operátorral a hivatkozást követjük. Mivel a Zeneszámai egy tulajdonság, a @ jel kell megelőzze, majd a hivatkozott elem típusát, vagyis a Zeneszám-ot kell megadjuk.
- Hasonlóan az Előadója is IDREF típusú tulajdonsága az Album elemnek, a hivatkozást (@Előadója=>) követve és a hivatkozott elem típusát megadva: Előadó, az Album Előadó elemét kapjuk meg.

A lekérdezés eredménye:

```
<rockZene>
  <SzCíme>In the Shadows</SzCíme>
  <ENév>The Rasmus</ENév>
</rockZene>
<rockZene>
  <SzCíme>Guilty</SzCíme>
  <ENév>The Rasmus</ENév>
</rockZene>
```

példa: Adjuk meg zenestíluson belül a rendelkezésünkre álló zene időtartamát, a következő formában:

```
<StílusösszIdő>
  <StílusCsop>
    <Stílus>rock</Stílus>
    <Időtartam>1233</Időtartam>
  </StílusCsop>
  <StílusCsop>
    <Stílus>rap</Stílus>
    <Időtartam>560</Időtartam>
  </StílusCsop>
     ⋮
</StílusösszIdő>
```

Megoldás:

```
<StílusösszIdő>
for $s in
 distinct-values(doc("zene.xml")//Album/Stílus)
let $a in doc("zene.xml")//Album[Stílus=$s]
let $z=$a/@Zeneszámai=>Zeneszám
return
<StílusCsop>
 <Stílus>{$s}</Stílus>
  <Időtartam>{ sum($z/Időtartam) } </Időtartam>
</StílusCsop>
</StílusösszIdő>
```

- $s változó a különböző stílus értékeket veszi fel.
- egy adott stílus érték esetén az $a változó azon albumok sorozatát tartalmazza, melyeknek a stílusa az $s változó értékével egyenlő.
- ezen albumok összes zeneszáma a $z változóba kerül, melynek Időtartam nevű elemének értékét összeadjuk a sum($z/Időtartam) függvény segítségével, mely egy értéket és nem elemet ad vissza, ezért szükség volt megadni az elemet bevezető és elemet záró tagot.
- a distinct-values függvény is értéket és nem elemet ad vissza.

Ha a zenés albumok XML adatai hierarhikus formában vannak megadva és a zene2.xml állományban vannak tárolva, a lekérdezéseket is másképpen kell megadjuk.

példa: Keressük a rock stílusú albumok zeneszámait és előadójuk nevét!

```
for $a in doc("zene2.xml")//Album
    $z in $a/Zeneszám
let $e := $a/parent::Előadó
where $a/Stílus = "rock"
return <rockZene>
{  $z/SzCíme, $e/ENév }
</rockZene>
```

- az `$a` változó az Album elemeken fut végig, a szűrőfeltétel a rock zenét válogatja ki.
- a Zeneszám ebben az esetben gyerekeleme az albumnak, az előadó pedig szülő eleme az albumnak.

példa: Adjuk meg zenestíluson belül a rendelkezésünkre álló zene időtartamát:

```
<StílusösszIdő>
for $s in
distinct-values(doc("zene2.xml")//Album/Stílus)
let $a in doc("zene.xml")//Album[Stílus=$s]
let $z=$a/Zeneszám
return
 <StílusCsop>
   <Stílus>{$s}</Stílus>
   <Időtartam>{ sum($z/Időtartam) }
   </Időtartam>
 </StílusCsop>
</StílusösszIdő>
```

Mennyiség meghatározás az XQuery-ben

Vannak olyan kifejezések, amelyek eredményükben azt mondják, hogy „mindegyik” és „létezik”.

```
every változó in kifejezés1 satisfies
                kifejezés2
```

```
some változó in kifejezés1 satisfies
                kifejezés2
```

– *kifejezés1* adatok szekvenciáját állítja elő, és a változó ciklusban megkapja értékül az egyes adatokat.
– ezeknek az értékeknek mindegyikére a *kifejezés2* (amely normál esetben tartalmazza a változót) kiértékelésre kerül és egy logikai értéket állít elő
– „every” változatban az egész kifejezés eredménye hamis, ha van olyan, a *kifejezés1* által előállított elem, amely hamissá teszi a *kifejezés2*-t; ellenkező esetben igaz lesz az eredmény.
– „some” változatban az egész kifejezés eredménye igaz, ha van olyan, a *kifejezés1* által előállított adat, amely igazzá teszi a *kifejezés2*-t; ellenkező esetben hamis lesz az eredmény.

Példa:

```
let $színészek := doc(”színészek.xml”)
for $s in $színészek/Színészek/Színész
where every $c in $s/Lakcím/Város satisfies
   $c = ”Hollywood”
return $s/Név
```

some nem szükséges:

```
let $színészek := doc(”színészek.xml”)
for $s in $színészek/Színészek/Színész
where $s/Lakcím/Város = ”Hollywood”
return $s/Név
```

Elágazás az XQuery kifejezésekben

Egy XQuery if-then-else kifejezés formája

`if (`*`kifejezés1`*`) then `*`kifejezés2`*` else `*`kifejezés3`*

- először kiértékeljük a *kifejezés1*-et.
- amennyiben ez igaz, akkor kiértékeljük a *kifejezés2*-t, amely a teljes kifejezés eredménye lesz.
- ha a *kifejezés1* hamis, akkor az egész kifejezés eredménye a *kifejezés3* lesz.

Példa:

```
let $kk :=
      doc(”filmek.xml”)/Filmek/Film[@FilmCím =
                                ”King Kong”]
for $v in $kk/Verzió
return
   if ($v/@év = max($kk/Verzió/@év))
   then <Legfrissebb>{$v}</Legfrissebb>
   else <Régebbi>{$v}</Régebbi>
```

Összekapcsolások az XQuery-ben

- két xml állomány adatait úgy kapcsolhatjuk össze, hogy mindegyikhez egy-egy változót rendelünk.
- ha csak ennyit teszünk, akkor az xml adatok Descartes szorzatát kapjuk
- a joint úgy valósíthatjuk meg, hogy a Descartes szorzatból a where feltétel segítségével kiválogatjuk a megfelelő párokat.

példa: Legyen a könyvészetet leíró adat a bibl.xml állományban és feltételezzük, hogy ezen könyvekről bírálatokat tartalmazó adatok egy review.xml állományban vannak tárolva. Legyen a reviews.xml állomány tartalma:

```
<bírálatok>
  <bírálat>
   <cím>TCP/IP Illustrated</cím>
   <osztályzás>5</osztályzás>
   <megjegyzés>Excellent technical content. Not
                  much plot.
   </megjegyzés>
  </bírálat>
  <bírálat>
   <cím>Data on the Web</cím>
   <osztályzás>4</osztályzás>
     <megjegyzés>A well founded introduction to
            semistructured data.
     </megjegyzés>
```

```
  </bírálat>
</bírálatok>
```

A következő lekérdezés a bibl.xml állományban található bibliográfiai elemekhez a róluk szóló bírálatokat társítja, ha van róluk bírálat a reviews.xml állományban. A könyv vagy cikk *címe* az, ami alapján összetársítjuk a könyvet, illetve a cikket, a róla szóló bírálattal. Az eredmény cím és bírálat párosokat fog tartalmazni.

```
for $c in distinct-values(doc("bibl.xml")//cím)
    $b in doc("reviews.xml")//bírálat
where $c = $b/cím
return <bírálat>{ $c, $b/megjegyzés }</bírálat>
```

A lekérdezés eredménye:

```
<bírálat>
   <cím>TCP/IP Illustrated</cím>
   <megjegyzés>Excellent technical content. Not
                  much plot.
   </megjegyzés>
</bírálat>
<bírálat>
   <cím>Data on the Web</cím>
      <megjegyzés>A well founded introduction to
          semistructured data. </megjegyzés>
</bírálat>
```

Amint látjuk, csak azok a könyvek jelennek meg, melyekről van bírálat, tehát ez egy belső (inner) join.

Ha azokat a könyveket is az eredménybe szeretnénk látni, melyekről nem létezik bírálat, a következőképpen tehetjük meg:

példa: Ez a példa egy külső baloldali összekapcsolás (left outer join) a könyvek (bibl.xml) és bírálatok (reviews.xml) között.

```
for $c in doc("bibl.xml")//cím
return
<bírálat>
{ $c }
    {
   for $b in doc("reviews.xml")//bírálat
   where $b/cím = $c
   return $b/megjegyzés
 }
</bírálat>
```

példa: Vegyük ismét a zenés CD-ket tartalmazó példát, melyet még úgy is megtervezhetjük, hogy három különböző állományba tároljuk az előadókat, albumokat és zeneszámokat.
Vegyük az Előadók.xml-t, melyben az előadókat tároljuk és szerkezete a következő:

```
<!DOCTYPE Előadók[
   <!ELEMENT Előadók (Előadó*)
   <!ELEMENT Előadó (ENév, Nemzetiség) >
   <!ELEMENT ENév (#PCDATA)>
   <!ELEMENT Nemzetiség (#PCDATA)>
]>
```

Tekintsük az Albumok.xml-t, melyben az albumokat tároljuk a következő szerkezettel:

```
<!DOCTYPE Albumok[
<!ELEMENT Albumok (Album*)
<!ELEMENT Album (ACíme, ElőadóNév, Stílus,
     MegjelenEve) >
<!ELEMENT ACíme (#PCDATA)>
<!ELEMENT ElőadóNév (#PCDATA)>
<!ELEMENT Stílus (#PCDATA)>
<!ELEMENT MegjelenEve (#PCDATA)>
]>
```

A Zeneszamok.xml, a zeneszámokat tartalmazza és szerkezete a következő:

```
<!DOCTYPE Zeneszámok [
<!ELEMENT Zeneszámok (Zeneszám*)
<!ELEMENT Zeneszám (SzCíme, AlbumCíme,
    Időtartam) >
<!ELEMENT SzCíme (#PCDATA)>
<!ELEMENT AlbumCíme (#PCDATA)>
<!ELEMENT Időtartam (#PCDATA)>
]>
```

Amint látjuk az albumnál hivatkozunk az előadóra a nevével, illetve a zeneszámtól az albumra.

Ebben az esetben, ha kíváncsiak vagyunk a zeneszámok előadóira, illetve stílusára, használhatjuk a join műveletet.

```
for $z in doc("Zeneszamok.xml")//Zeneszám
    $a in doc("Albumok.xml")//Album
where $z/AlbumCíme = $a/ACíme
return <szám>
{ $z/SzCíme, $z/AlbumCíme, $a/ElőadóNév,
    $a/Stílus}
       </szám>
```