

Adatbázisok 1.

XML lekérdezőnyelvek

– 3. rész

XPath

XQuery

Hasznos linkek:


https://www.w3schools.com/xml/xquery_intro.asp

<https://www.tutorialspoint.com/xquery/index.htm>

XQuery

- Az XQuery egy SQL-hez hasonló lekérdező nyelv (*query language*), ami XPath kifejezéseket használ.
- Ugyanúgy a tételek listája adatmodellt használja.
- Az XQuery egy funkcionális nyelv, illetve kifejezés nyelv (*expression language*).
 - Ez magába foglalja, hogy ahol kifejezés szerepelhet, ott tetszőleges XQuery kifejezés szerepelhet. Ez eltérés az SQL-től. Az SQL-ben nem mindenhol szerepelhetett SQL alkérdés például.

Tételek listája (részletesebben)

- Az XQuery-ben előfordulhat, hogy listák listája generálódik.
- Az ilyen listákat a rendszer „sima” listává alakítja át.
- **Példa:** (1 2  (3 4)) = (1 2 3 4).

↑
Üres lista.

FLWR kifejezések (flower)

1. Egy vagy több **for** és/vagy **let** záradék.
2. Ezek után opcionálisan egy **where** záradék.
3. Végül egy **return** záradék.

Az FLWR kifejezések szemantikája

- Mindegyik **for** egy ciklust (*loop*) generál.
 - a **let** a cikluson belüli hozzárendeléseket adja meg.
- Minden iterációjánál a beágyazott ciklusnak (*nested loop*), ha van ilyen, ki kell értékelni a **where** záradékot.
- Ha a **where** záradék IGAZ, a **return** záradéknak megfelelő értéket a végeredményhez kapcsoljuk.

FOR záradék

for <változó> in <kifejezés>, . . .

- A változók \$ jellel kezdődnek.
- A **for** változója egy ciklusban sorra bejárja a kifejezés eredményének összes tételét.
- A **for** után megadott részek tehát minden egyes tételre végrehajtódnak egyszer.

teázók.xml tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<teázók>
  <teázó név = "JoeTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
    <ár melyikTea = "Pyramid">3.00</ár>
  </teázó>
  <teázó név = "SueTeázója">
    <ár melyikTea = "Brisk">3.50</ár>
  </teázó>
  <tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
</teázók>
```

Példa: FOR

Az előző dián
szereplő
dokumentum.

```
for $sor in  
  document("teázók.xml")/teázók/tea/@név  
return
```

```
<teaNév> {$sor} </teaNév>
```

- \$sor a példa dokumentum tea elemeinek név attribútumán fut végig.
- Az eredmény teaNév elemek listája:

```
<teaNév>Brisk</teaNév>
```

```
<teaNév>Pyramid</teaNév> . . .
```

A {} jelek
közötti kifejezést
mindig kiértékeli
a rendszer, ha a
return záradék
végrehajtására
kerül a sor a
ciklusban.

Kapcsos zárójelek (*braces*)

- Ha azt szeretnénk, hogy egy változó nevet, pl. \$x, ne sima szöveggént kezeljen a rendszer kapcsos zárójelek közé kell tennünk.
 - **Példa:** <A>\$x egy A elem lesz "\$x" értékkel ugyanúgy, mint a <A>foo is egy A elem "foo" értékkel.

Kapcsos zárójelek --- (2)

- De `return $x` értéke egyértelmű.
- Tagek vagy idézőjelek nélküli sima sztringet nem adhat vissza a lekérdezés, azaz, ha `$x`-t sima sztringként szeretnénk visszaadni, nem pedig a `$x` változó értékére vagyunk kíváncsiak, akkor az eredménynek `return <a>$x` vagy `return "$x"` alakúnak kell lennie.

LET záradék

let <változó> := <kifejezés>, . . .

- A változó értéke *tételek listája* lesz, ez a lista a kifejezés eredménye.
- A **let** záradék hatására nem indul el egy ciklus; a **for** záradék hatására igen.

Példa: LET

```
let $d := document("teázók.xml")
```

```
let $sor := $d/teázók/tea/@név
```

```
return
```

```
<teaNév> {$sor} </teaNév>
```

- Az eredmény egyetlen elemből áll az összes teanévvel:

```
<teaNév>Brisk Pyramid ...</teaNév>
```

teázók.xml tartalma:

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

```
<teázók>
```

```
  <teázó név = "JoeTeázója">
```

```
    <ár melyikTea = "Brisk">2.50</ár>
```

```
    <ár melyikTea = "Pyramid">3.00</ár>
```

```
  </teázó>
```

```
  <teázó név = "SueTeázója">
```

```
    <ár melyikTea = "Brisk">3.50</ár>
```

```
  </teázó>
```

```
<tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
```

```
</teázók>
```

Order-By záradék

- Az FLWR valójában FLWOR, ahol egy **order-by** záradék előzheti meg a **return** záradékot.
- Alakja: order by <kifejezés>
 - Opcionális **ascending** vagy **descending** (a <kifejezés> után írandó).
- A kifejezés a változók minden hozzárendelésére kiértékelődik.
- A végeredmény listájának sorrendjén változtat.

Példa: Order-By

- A Brisk összes árát kilistázzuk, a legalacsonyabb legelőször.

```
let $d := document("teázók.xml")
```

```
for $p in $d/teázók/teázó/ár[@melyikTea="Brisk"]
```

```
order by $p
```

```
return $p
```

← Rendezi a
hozzárendelés
értékeit.

← \$p-hez a megfelelő
ár elemeket rendeli.

↑
Az eredmény ár
elemeknek egy listája.

Összehasonlítás: SQL ORDER BY

- Az SQL ugyanezen az elven működik; a FROM és WHERE záradékok eredménye rendeződik, nem a végeredmény.
- **Példa:** R(a,b) relációra:

```
SELECT b FROM R  
WHERE b > 10
```

Aztán, a már rendezett sorokból vesszük a b értékeket.

```
ORDER BY a;
```

R sorai, ahol $b > 10$
az a értékek szerint
rendeződnek.

Példa: WHERE

teázók.xml tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<teázók>
  <teázó név = "JoeTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
    <ár melyikTea = "Pyramid">3.00</ár>
  </teázó>
  <teázó név = "SueTeázója">
    <ár melyikTea = "Brisk">3.50</ár>
  </teázó>
  <tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
</teázók>
```

```
for $x in document("teázók.xml")/teázók/teázó
where $x/@név = "JoeTeázója"
return $x/ár
```

- A fenti lekérdezés kimenete:

```
<ár melyikTea="Brisk">2.50</ár>
```

```
<ár melyikTea="Pyramid">3.00</ár>
```


Predikátumok

teázók.xml tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<teázók>
  <teázó név = "JoeTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
    <ár melyikTea = "Pyramid">3.00</ár>
  </teázó>
  <teázó név = "SueTeázója">
    <ár melyikTea = "Brisk">3.50</ár>
  </teázó>
  <tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
</teázók>
```

- A feltételekben a „**létezést**” követeljük meg.
- **Példa:** /teázók/teázó[@név] jelentése “az összes olyan teázó, aminek létezik neve.”
- **Példa:** /teázók/tea[@árulja = "JoeTeázója"] azon teák listáját adja vissza, melyeket Joe teázójában megkaphatunk.

Példa: összehasonlítások

teázók.xml tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<teázók>
  <teázó név = "JoeTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
    <ár melyikTea = "Pyramid">3.00</ár>
  </teázó>
  <teázó név = "SueTeázója">
    <ár melyikTea = "Brisk">3.50</ár>
  </teázó>
  <tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
</teázók>
```

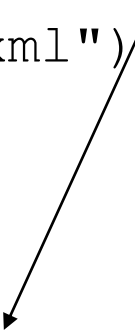
- Az összes teára, amit Joe teázójában árulnak, adjuk vissza az ár elemeket az összes teázót figyelembe véve.
- Az eredmény TTÁ („teázó-tea-ár”) elemekből áll majd, melynek attribútumai a teázó és a tea nevét adják majd meg, egy aleleme pedig az árat.

Stratégia

1. Készítsünk egy tripla for ciklust, ahol vesszük az összes **tea** elemet, aztán az összes **teázó** elemet, majd minden egyes teázóra a hozzá tartozó **ár** elemeket.
2. Ellenőrizzük, hogy a teát árulják-e Joe teázójában, és hogy a tea neve és az aktuális **ár** elemben a **melyikTea** attribútum értéke egyezik-e. `(contains($arg1, $arg2))`
3. A kívánt alakú eredmény megadása.

A lekérdezés

Ez igaz, ha 'JoeTeázója'
részsztringje az aktuális
tea tag árulja
attribútum értékének



teázók.xml tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<teázók>
  <teázó név = "JoeTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
    <ár melyikTea = "Pyramid">3.00</ár>
  </teázó>
  <teázó név = "SueTeázója">
    <ár melyikTea = "Brisk">3.50</ár>
  </teázó>
  <tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
</teázók>
```

```
let $tzk := document("teázók.xml")//teázók
for $tea in $tzk/tea
for $tz in $tzk/teázó
for $ar in $tz/ár
where contains($tea/@árulja, 'JoeTeázója') and $ar/@melyikTea = $tea/@név
return <TTÁ tea='{ $tea/@név}' teázó='{ $tz/@név}'>{ $ar}</TTÁ>
```

- A fenti lekérdezés kimenete:

```
<TTÁ tea="Brisk" teázó="JoeTeázója">
  <ár melyikTea="Brisk">2.50</ár>
</TTÁ>
<TTÁ tea="Brisk" teázó="SueTeázója">
  <ár melyikTea="Brisk">3.50</ár>
</TTÁ>
```

Példa: összehasonlítások

- Adott az alábbi **vendégek.xml** XML dokumentum:

```
<?xml version="1.0" encoding="UTF-8"?>
<Vendégek>
  <Vendég>
    <Név>Sue</Név>
    <Mobil>
      <Szolgáltató>70</Szolgáltató>
      <Szám>1234567</Szám>
    </Mobil>
  <Mobil>
    <Szolgáltató>20</Szolgáltató>
    <Szám>7654321</Szám>
  </Mobil>
</Vendég>
</Vendégek>
```

Példa: összehasonlítások

- Szeretnénk lekérni azokat az vendégeket, akiknek a mobilszolgáltatójuk 70-es és a számuk 7654321

- Első kísérletünk:

```
for $a in document("vendégek.xml")/Vendégek/Vendég  
where $a/Mobil/Szolgáltató = '70' and
```

```
    $a/Mobil/Szám = '7654321'
```

```
return $a/Név
```

- Meglepő eredmény:

```
<Név>Sue</Név>
```

Szigorú összehasonlítások (*strict comparisons*)

- Ha meg szeretnénk követelni, hogy az összehasonlított listák egyetlen elemet tartsanak, az alábbi összehasonlításokat kell alkalmaznunk:
 - eq, ne, lt, le, gt, ge.
- **Példa:** `$a/Mobil/Szolgáltató eq 70` igaz, ha a `$a/Mobil/Szolgáltató` kifejezés eredményeként kapott tételek listája egyetlen elemet tartalmaz, melynek az 70 értéke.

Példa: összehasonlítások

- Tegyük fel, hogy az **vendégek.xml** XML dokumentum vége az alábbiban különbözik:

...

```
<Vendég>
```

```
<Név>Joe</Név>
```

```
<Mobil>
```

```
<Szolgáltató>70</Szolgáltató>
```

```
<Szám>7654321</Szám>
```

```
</Mobil>
```

```
<Mobil>
```

```
<Szolgáltató>30</Szolgáltató>
```

```
<Szám>1111111</Szám>
```

```
</Mobil>
```

```
</Vendég>
```

```
</Vendégek>
```


Példa: összehasonlítások

- Második kísérletünk:

```
for $a in document("vendégek.xml")/Vendégek/Vendég
where $a/Mobil/Szolgáltató eq '70' and
      $a/Mobil/Szám eq '7654321'
return $a/Név
```

- Az összehasonlítás elbukik!

Elemek és értékek összehasonlítása

teázók.xml tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<teázók>
  <teázó név = "JoeTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
    <ár melyikTea = "Pyramid">3.00</ár>
  </teázó>
  <teázó név = "SueTeázója">
    <ár melyikTea = "Brisk">3.50</ár>
  </teázó>
  <tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
</teázók>
```

- Ha egy elemet hasonlítunk össze egy értékkel,
- akkor az elemnek a hozzátartozó értékét vesszük, ha az az érték atomi.
- **Példa:**

```
/teázók/teázó[@név="JoeTeázója"]/ár[@melyikTea="Brisk"] eq "2.50"
```

- **Másképpen:**

```
for $i in document("teázók.xml")/teázók/teázó[@név='JoeTeázója']/ár
where $i[@melyikTea='Brisk'] eq '2.50'
return $i
```

Az elemek adatainak kinyerése

- Tegyük fel, hogy elemek értékeit szeretnénk összehasonlítani.
- Az E elem értékét a *data* függvény használatával kaphatjuk meg: *data(E)*.

Példa: data()

```
let $tzk := document("teázók.xml")/teázók
for $tea in $tzk/tea
for $tz in $tzk/teázó
for $ar in $tz/ár
where contains($tea/@árulja, 'JoeTeázója')
and $ar/@melyikTea = $tea/@név
```

← Régebbi lekérdezés a vége nélkül

teázók.xml tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<teázók>
  <teázó név = "JoeTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
    <ár melyikTea = "Pyramid">3.00</ár>
  </teázó>
  <teázó név = "SueTeázója">
    <ár melyikTea = "Brisk">3.50</ár>
  </teázó>
  <tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
</teázók>
```

- Tegyük fel, hogy az előbbi lekérdezésünkben módosítjuk a return záradékot:

```
return data($tz/@név)
```

- A fenti lekérdezés kimenete:

JoeTeázója SueTeázója

Példa: összehasonlítások

- Harmadik kísérletünk:

```
for $a in document("vendégek.xml")/Vendégek/Vendég
```

where `$a/Mobil[data (Szolgáltató) eq '70']` and

```
data (Szám) eq '7654321']
```

```
return data ($a/Név)
```

- Visszatér „Joe”-val

```
<?xml version="1.0" encoding="UTF-8"?>
<Vendégek>
  <Vendég>
    <Név>Sue</Név>
    <Mobil>
      <Szolgáltató>70</Szolgáltató>
      <Szám>1234567</Szám>
    </Mobil>
    <Mobil>
      <Szolgáltató>20</Szolgáltató>
      <Szám>7654321</Szám>
    </Mobil>
  </Vendég>
  <Vendég>
    <Név>Joe</Név>
    <Mobil>
      <Szolgáltató>70</Szolgáltató>
      <Szám>7654321</Szám>
    </Mobil>
    <Mobil>
      <Szolgáltató>30</Szolgáltató>
      <Szám>1111111</Szám>
    </Mobil>
  </Vendég>
</Vendégek>
```

Ismétlődések kiszűrése

- Használjuk a `distinct-values` függvényt, aminek a paramétere: elemek listája.
- **Finomság:** a függvény a tagek nélkül veszi az értékeket és ezek szöveges értékét hasonlítja össze.
 - Az eredményben nem írja vissza a tageket.

Példa: az összes különböző ár

teázók_V.xml tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<teázók>
  <teázó név = "JoeTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
    <ár melyikTea = "Pyramid">3.00</ár>
  </teázó>
  <teázó név = "SueTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
  </teázó>
  <tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
</teázók>
```

```
return distinct-values (
```

```
  let $tzk := doc("teázók.xml")
  return $tzk/teázók/teázó/ár
```

```
)
```

Emlékezzünk vissza: az XQuery funkcionális nyelv, azaz ahol érték jelenhet meg, oda tetszőleges XQuery kifejezést is írhatunk.

Logikai érték (boolean)

- Egy-egy kifejezés *logikai értéke* a következő:
 1. ha a kifejezés logikai típusú, akkor az aktuális érték.
 2. FALSE, ha a kifejezés kiértékelésének eredménye: 0, "" [az üres sztring] vagy () [az üres lista].
 3. TRUE, különben.

Példa: logikai értékek

1. `@név="JoeTeázója"` TRUE, ha név attribútum értéke "JoeTeázója".
2. `/teázók/teázó[@név="Lórúgás"]` TRUE, ha létezik olyan teázó, amely név attribútumának Lórúgás az értéke.


Logikai műveletek

- E_1 and E_2 , E_1 or E_2 , not(E) tehát minden kifejezésre alkalmazható.
- **Példa:** not(3 eq 5 or 0) az értéke TRUE.
- A true() és false() függvények (paraméterek nélkül) TRUE és FALSE értéket adnak vissza. A konstansok kiírása helyett ezt használják.

Elágazó kifejezések (*branching expressions*)

- if (E_1) then E_2 else E_3 is értelmezhető.

- Példa:

```
if ($tz/@név eq "JoeTeázója")  
then $tz/ár else 
```

Az üres lista.

Kvantorok (*quantifiers*)

some $\$x$ in E_1 satisfies E_2

1. E_1 -t ki kell értékelni.
 2. $\$x$ vegye fel sorban E_1 eredményének értékeit, és értékeljük ki mindegyik értékkel E_2 -t.
 3. Az eredmény IGAZ, ha $\$x$ legalább egy értékére E_2 igaz.
- Hasonlóan:

every $\$x$ in E_1 satisfies E_2

Példa: Some

```
for $tz in
  doc("teázók.xml")/teázók/teázó
where some $p in $tz/ár
  satisfies $p < 2.00
return data($tz/@név)
```

teázók.xml tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<teázók>
  <teázó név = "JoeTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
    <ár melyikTea = "Pyramid">3.00</ár>
  </teázó>
  <teázó név = "SueTeázója">
    <ár melyikTea = "Brisk">3.50</ár>
  </teázó>
  <tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
</teázók>
```

Vegyük észre: a where \$tz/ár < 2.00
ugyanazt a hatást éri el.

Példa: Every

```
for $tz in
  doc("teázók.xml")/teázók/teázó
where every $p in $tz/ár
  satisfies $p <= 5.00
return data($tz/@név)
```

teázók.xml tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<teázók>
  <teázó név = "JoeTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
    <ár melyikTea = "Pyramid">3.00</ár>
  </teázó>
  <teázó név = "SueTeázója">
    <ár melyikTea = "Brisk">3.50</ár>
  </teázó>
  <tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
</teázók>
```

Dokumentum sorrend

- A dokumentum sorrend szerinti összehasonlítás műveletei: << és >>.
- **Példa:** \$d/teázók/tea[@név="Brisk"] << \$d/teázók/tea[@név="Pyramid"] igaz, ha a bal oldali tea elem megelőzi a jobb oldalit.

Halmazműveletek

- A **union**, **intersect**, **except** itt is alkalmazhatóak pontok listájára.
 - A jelentés hasonló SQL-beli jelentéshez.
 - Az ismétlődések itt is törlődnek.
 - Az eredmény elemei dokumentum sorrendben jelennek meg.

Összesítések

- Összesítő függvények: `count`, `sum`, `max`, `min`, `avg` itt is megvannak
- Ezek bemenete tételek listája
- Ez lehet üres lista is, viszont ennek a kezelésében már eltérő viselkedést mutatnak

Összesítések

- Például az alábbira:

```
let $elemek := ()
```

```
let $aggr := count($elemek)
```

```
return <result>{$aggr}</result>
```

- az eredmény:

```
<result>0</result>
```

- ugyanezt kapjuk, ha `sum` szerepel a `count` helyén, viszont ha `min`, `max`, `avg`, akkor:

```
<result></result>
```

Összekapcsolások

- Az XQueryben össze lehet kapcsolni két dokumentumot
- Változók kellenek a megfelelő dokumentumokra
- A for záradék kell egy dokumentum elemeinek előállítására

Példa: összekapcsolások

- Adott az alábbi **vendégek2.xml** XML dokumentum:

```
<?xml version="1.0" encoding="UTF-8"?>
<Vendégek>
<Vendég>
<Név>Sue</Név>
<KedvencTea>Pyramid</KedvencTea>
</Vendég>
<Vendég>
<Név>Joe</Név>
<KedvencTea>Brisk</KedvencTea>
<Cím>Fő utca 1.</Cím>
</Vendég>
</Vendégek>
```

teázók2.xml tartalma:

```
<?xml version="1.0" encoding="UTF-8"?>
<teázók>
  <teázó név = "JoeTeázója">
    <ár melyikTea = "Brisk">2.50</ár>
    <ár melyikTea = "Pyramid">3.00</ár>
  </teázó>
  <teázó név = "SueTeázója">
    <ár melyikTea = "Brisk">3.50</ár>
  </teázó>
  <tea név = "Brisk" árulja = "JoeTeázója SueTeázója"/>
  <tea név = "Pyramid" árulja = "JoeTeázója"/>
</teázók>
```

Példa: összekapcsolások

- Szeretnénk lekérni az vendég-„kedvenc tea”-„hol árulják” hármassokat:

```
for $a in document("vendégek2.xml")/Vendégek/Vendég
for $s in document("teázók2.xml")/teázók/tea
where data($a/KedvencTea) = data($s/@név)
return <result név = "{data($a/Név)}"
        kedvencTea = "{data($a/KedvencTea)}"
        hol="{data($s/@árulja)}"/>
```

- Az eredmény:

```
<result név="Joe" kedvencTea="Brisk" hol="JoeTeázója SueTeázója"/>
<result név="Sue" kedvencTea="Pyramid" hol="JoeTeázója"/>
```