Adatbázisok II.

6

Jánosi-Rancz Katalin Tünde
tsuto@ms.sapientia.ro
327A

XQUERY

XQUERY jellemzői

- XML dokumentumok lekérdezésére szolgáló lekérdezőnyelv
- W3C szabvány 2007 óta; a böngészők és ABKR értik és végre is tudják hajtani az XQuery utasításokat.
- Halmazorientált
- XML bemeneti adat és XML kimeneti adat
- a parancsok nem XML formátumban adottak
- XPath alapú, az elérési-út kifejezéseit használja fel
- SQL-hez hasonló (de laza a kapcsolat)
- XSLT vetélytársa
- procedurális elemeket is tartalmaz
- az XQuery csak lekérdezésre ad lehetőséget, adatmódosításra nem (XUPDATE)
- teljes specifikációja a http://www.w3.org/TR/xquery/

XPATH

- lehetőséget ad egy XML dokumentumban való navigációra (gráfok-fa szerkezet)
- Példa:
- doc() - egy URI (Universal Resource Identifier) által megadott dokumentum csomópontot ad vissza
  - doc("bibl.xml")/bibliográfia/könyv
  - a doc függvény megnyitja a bibl.xml állományt,
  - /bibliográfia szelektálja a dokumentum gyökér elemét
  - /könyv pedig a könyv elemeket.
- Egy XPath lépés általános formája a következő:
  - irány::csomópont_típus
    [feltételes_kifejezés]

XPATH példák - Feltételes kifejezések

- Példa 1:
  - doc("bibl.xml")/bibliográfia/könyv/attribute::év
  - rövidebben:
  - doc("bibl.xml")/bibliográfia/könyv/@év

-az elérési-út kifejezés által szolgáltatott csomópontok egy részhalmazát választjuk ki.
- a feltételes kifejezést szögletes zárójelben adjuk meg

- Példa 2: A következő elérési-út kifejezés azon szerző elemeket adja meg, melyeknek vezeték neve Stevens.
  - doc("bibl.xml")/bibliográfia/könyv/szerző[vnév="Stevens"]

- Példa 3: A köv. kifejezés minden könyv esetén az első szerzőt adja meg.
  - doc("bibl.xml")/bibliográfia/könyv/szerző[1]

- Példa 4: az egész dokumentum legelső szerzője:
  - (doc("bibl.xml")/bibliográfia/könyv/szerző)[1]

XPATH példák - Feltételes kifejezések

- tulajdonság értékére is vonatkozhat feltétel.
- a tulajdonság nevét a @ kell megelőzze.

példa:

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

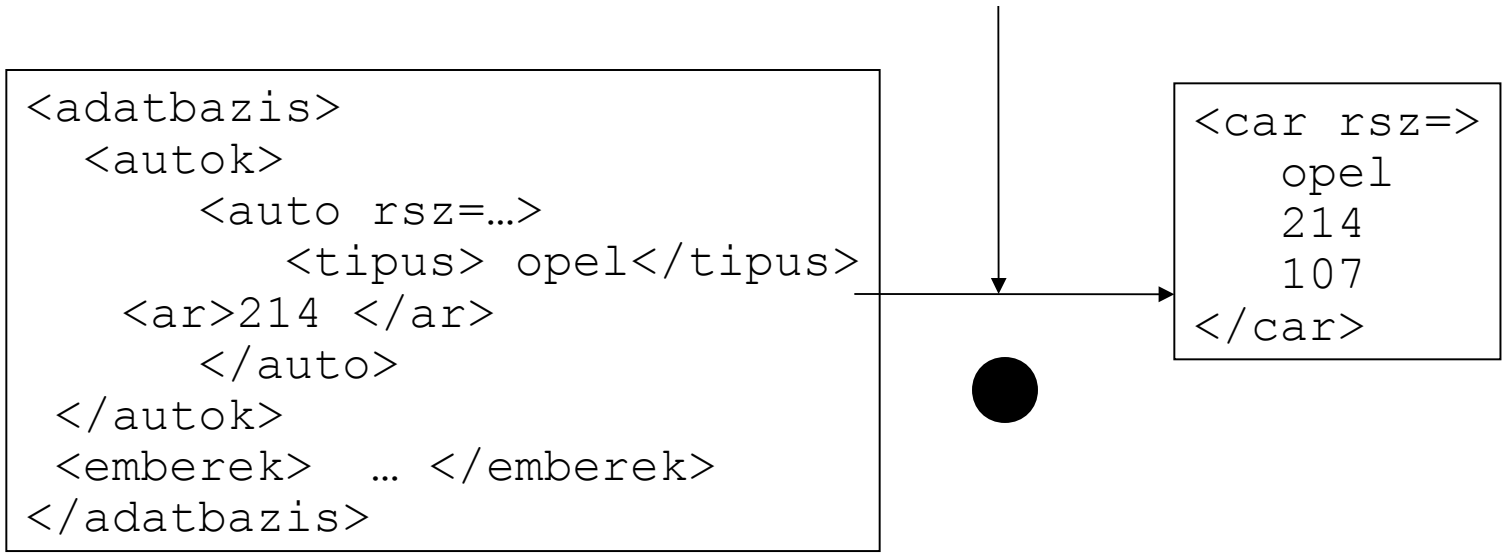
Vagy:

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

a * bármely elem nevét helyettesíti.

XQuery minta

```
for $x in doc('xx9.xml')/adatbazis/autok/auto
where $x/ar < 222
order by $x/tipus descending
return
   <car> {$x/@rsz} {$x/tipus/text()} {$x/ar} {ll:felez($x/ar)}</car>
```



XQuery nyelvi alapok

XPath-ra épülő kifejezések:

- element() : bármely csomópont
- element(A,B) : A nevű, B típusú csomópont
- attribute(A,B) : A nevű, B típusú elemjellemző
- text() : szövegcsomópont
- node() : bármely csomópont
- node() * : bármely csomópont akárhányszor
- attribute() + : egy vagy több elemjellemző
- element(*,B)? : opcionális B típusú elem
- derives-from(A,B) : A típus a B-ből származik-e
- item() : csomópont vagy érték
- comment() : megjegyzés

XQuery nyelvi alapok

XPath-ra épülő kifejezések:

szekvencia : (1,2,5,…,8)
tartomány : 1 to 6
érték összehasonlítás : eq ne lt le gt ge
szekvencia, tartomány összehasonlítás: =,<, >, !=
csomópont összehasonlítás : is << >>

kifejezés megadása kiértékelésre: { kif }

statikus eredmény XML struktúra felépítése: <nev>

<auto> Fiat, melynek ara {2+4} </auto>

XQuery lekérdezés struktúrája - FLWOR kifejezések:

- A for záradék használata opcionális. A változókhoz szekvenciákat rendelünk, majd ezeket iteratívan feldolgozzuk.
- A let záradék (opcionális) segítségével szintén szekvenciákat rendelkhetünk a változókhoz, ám itt nem történik semmiféle iteráció.
- A where záradék az SQL WHERE záradékához hasonlóan "szűrő" feltételeket tartalmazhat. Ez is opcionális záradék.
- Az order by záradék (opcionális) használatával az eredmény rendezetten jelenik meg.
- A return segítségével adhatjuk meg a lekérdezés végeredményét. Ez az egyetlen kötelező záradék.

FLWOR II.

- A lekérdezés elején tetszőleges számú for és let záradék szerepelhet tetszőleges sorrendben.
- Egy let vagy for záradéknak szerepelnie kell.
- A lekérdezést pontosan egy return záradéknak kell zárnia.
- Érzékeny a kis-, nagybetű különbségekre. (Ezért például az iménti záradékokat is kisbetűvel kell írni.)
- A legegyszerűbb XQuery program:

```
let $x := 1
return <üdvözlet>Szeretlek világ!</üdvözlet>
```

Egyértékű változó – a LET záradék

```
LET $v := ertek
RETURN
    kifejezes
```

```
let $x := (‘a’,’b’,’s’)
return <a> {$x} </a>
```

```
let $x := (‘a’,’b’,’s’)
return <a> $x </a>
```

Többértékű változó - a FOR záradék

```
FOR $v IN lista
RETURN
    kifejezes
```

```
for $x in (‘a’,’b’,’s’)
return <a> {$x} </a>
```

- Példa:

```
for $i in (2, 3, 4), $j in ($i+5, 2)
return ($i, $j)
```

- Eredmény:

```
2 7 2 2 3 8 3 2 4 9 4 2
```

FOR vs LET

FOR használata esetén 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.

```
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>
```

LET használata esetén: 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 záradék -Példa- Descartes szorzat

```
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>
```

XML dokumentum kijelölés

```
doc(file-specifikáció)
```

Részfa kijelölés (XPath)

```
doc(file-specifikáció)/p1/p2/…
```

```
for $x in doc("xx9.xml")/adatbazis/autok/auto
return <a> {$x}  </a>
```

Példák I.

```
for $b in doc("bibl.xml")//könyv
where $b/ár < 50
return $b/cím
Eredmény
<cím>Data on the Web</cím>
```

```
for $x in doc("kolcsonzes.xml")//kolcsonzo
where $x/nev = 'Kis Virag'
return $x//cd
```

```
for $x in doc("kolcsonzes.xml")//kolcsonzo
where string-length($x/nev) > 9
return $x/nev
```

```
for $x in doc("kolcsonzes.xml")//kolcsonzo[string-
   length(nev)>9]
return $x/nev
```

```
for $x in doc("kolcsonzes.xml")//kolcsonzo
where $x//konyv[@ar=2500 and cim='Momo']
return $x/nev
```

Példák II.

```
for $x in doc("kolcsonzes.xml")//kolcsonzo
where count($x//konyv) >= 2
return ($x/nev, <azon>{string($x/@azon)}</azon>)

for $x in doc("kolcsonzes.xml")//kolcsonzo
return ($x/nev, <konyvek>{for $y in $x//konyv
          return $y/cim}</konyvek>)

for $x in doc("kolcsonzes.xml")/kolcsonzesek,
      $y in $x/kolcsonzo
where $y//ar = max($x//ar)
return $y/nev

let $x := distinct-values(
for $y in doc("kolcsonzes.xml")//cd
return string($y/@eloado))
return $x
```

Csomópontok létrehozása

```
FOR $v IN lista
…
RETURN
    {ELEMENT {nev} {ertek}}
    {ATTRIBUTE {nev}{ertek}}
    {TEXT{ertek}}
```

```
for $x in doc('xx9.xml')/adatbazis/autok/auto
where $x/ar > 222
return
  <a>  {element  car {text{$x/@rsz}}}   </a>
```

Elemek rendezése

```
FOR $v IN lista
LET $w := kifejezes
WHERE feltetel
ORDER BY kifejezes mod
RETURN
    kifejezes
```

```
for $x in doc('xx9.xml')/adatbazis/autok/auto
where $x/ar < 222
order by $x/tipus descending
return
  <a>  <car> {$x/@rsz} {$x/tipus/text()} </car>  </a>
```

Példa

a 200-nál drágább autók rendszáma

```
for $v in fn:doc('xx9.xml')//auto
where $v/ar>200
return
 element eredmeny {$v/@rsz}
```

Autó rendszáma és a tulaj neve

```
for $a in fn:doc('xx9.xml')//auto
for $e in fn:doc('xx9.xml')//ember
where $a/@tulaj eq $e/@kod
return
 element eredmeny {
     element auto {$a},
     element tulaj  {$e}
 }
```

Minta

Autók rendszám és ár , ár szerint rendezve

```
for $a in fn:doc('xx9.xml')//auto
order by $a/ar
return
   element auto {$a/tipus,  $a/ar}
```

A 200-nál drágább autók rendszámai rsz sorrendben

```
for $a in fn:doc('xx9.xml')//auto
where $a/ar > 200
order by $a/@rsz
return
   <aa>  {$a/@rsz} </aa>
```

Autó elemek rendszám tartalommal

```
for $a in fn:doc('xx9.xml')//auto
return
  element auto {text{$a/@rsz}}
```

Autokhoz olyan elemek, melynek neve a rendszám értéke, tartalma: tipus, jellemzője: ár

```
for $a in fn:doc('xx9.xml')//auto
return
  element {$a/@rsz} {attribute a {$a/ar} , text{$a/tipus}}
```

Feltételes végrehajtás

```
FOR $v IN lista
…
RETURN
    IF (kifejezes)
    THEN kifejezes
    ELSE kifejezes
```

- Az else ág megadása kötelező. Viszont, ha utána az üres szekvencia: () szerepel, akkor olyan, mintha nem lenne else ág.

```
for $x in doc('xx9.xml')/adatbazis/autok/auto
return
  <a>  <car>
    {$x/@rsz} {$x/tipus/text()} {$x/ar}
     {if ($x/ar>151) then 'sok' else 'keves'}
</car>  </a>
```

```
<a>  {
for $x in doc('xx9.xml')/adatbazis/autok/auto
return
   <car> {$x/@rsz} {if ($x/ar>151) then 'sok' else 'keves'}</car>
 }   </a>
```

```
<adatok>
{
for $a in fn:doc('xx9.xml')//auto
return
  element car { text {$a/tipus},
  element ar  {if  ($a/ar/@valuta eq 'USD')
  then  text {250*$a/ar} else  text {$a/ar}  }}
 }
</adatok>
```

Gyári függvények

```
doc()
uppercase()
substring()

max()
min()
avg()
sum()
count()

distinct-values()

every $x in kif1 satisfies kif2
some $x in kif1 satisfies kif2
```

Autotipusok es autoik

```
<adatok>
{
for $t in fn:distinct-values( fn:doc('xx9.xml')//auto/tipus )
return
     element tipus {attribute tip {$t},
     element autok {
        for $a in fn:doc('xx9.xml')//auto
        where $a/tipus eq $t
        return
           element auto  {$a/@rsz}
     }
     }
}
</adatok>
```

Tipusok és darabszámuk

```
<adatok>
{
for $t in fn:distinct-values( fn:doc('xx9.xml')//auto/tipus )
return
     element tipus {attribute tip {$t},
     attribute db {count(
        for $a in fn:doc('xx9.xml')//auto
        where $a/tipus eq $t
        return
           element auto  {$a/@rsz}
       )
     }
     }
}
</adatok>
```

Gyári függvények

```
for $x in doc('xx9.xml')/adatbazis/autok
return
  <atlagar> {fn:avg($x//ar)} </atlagar>
```

```
<db>
  {
     fn:count(
        for $x in doc("xx9.xml")/adatbazis/autok/auto
        where $x/ar > 211
        return $x
     )
  }
</db>
```

Gyári függvények

Akiknek nincs autoja

```
for $e in fn:doc('xx9.xml')//ember
let $a := fn:doc('xx9.xml')//auto[@tulaj = $e/@kod]
where fn:count($a) eq 0
return
   element ember {$e/nev}
```

```
for $e in fn:doc('xx9.xml')//ember
where  not (some $x in fn:doc('xx9.xml')//auto
    satisfies $e/@kod eq $x/@tulaj)
return
   element ember {text {$e/nev}}
```

Saját függvények létrehozása

```
DECLARE NAMESPACE prefix=kifejezes;

DECLARE FUNCTION prefix:fnev ($p1 AS t1,..) AS rtip
{
    utasitasok…
    RETURN kifejezes
}
```

Saját függvények

```
declare namespace ll="http:me.kl";
declare function ll:felez($x as xs:decimal) as xs:decimal
{
   let $c:=2
   return $x div $c
};

for $x in doc('xx9.xml')/adatbazis/autok/auto
where $x/ar < 222
order by $x/tipus descending
return
   <car> {$x/@rsz} {$x/tipus/text()}
         {$x/ar} {ll:felez($x/ar)}</car>
```

Első n szám összege

```
declare namespace ll="http:me.kl";
declare function ll:ossz($r as xs:integer) as xs:integer
{
   let $x := 1
   return  (
    if ($r > 0) then
      $r +  ll:ossz($r -1)
    else
      0
    )
};

let $x := 5
return
   <ered> {ll:ossz($x)}</ered>
```

XQUERY -IDREF

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: 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.

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.

XQUERY - Join

A join műveletetre is ad lehetőséget az Xquery.

- 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 megfelel párokat.

Példa:

```
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>
```

XQUERY - példák

- Adott a következő movies.xml es genres.xml

```
<?xml version="1.0"?>
<movies
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation="movies.xsd">

     <movie>
          <genre>horror</genre>
          <title>The Excorsit</title>
          <year>1973</year>
          <rating>8.0</rating>
          <votes>41872</votes>
     </movie>

     <movie>
          <genre>sci-fi</genre>
          <title>Star Wars</title>
          <year>1977</year>
          <rating>8.8<7rating>
          <votes>161724</votes>
     </movie>

     <movie>
          <genre>horror</genre>
          <title>Psycho</title>
          <year>1960</year>
          <rating>8.6</rating>
          <votes>63723</votes>
     </movie>
     <movie>
          <genre>horror</genre>
          <title>Bram Stoker's Dracula</title>
          <year>1992</year>
          <rating>7.1<7rating>
          <votes>24380</votes>
     </movie>
     <movie>
          <genre>horror</genre>
          <title>The Shining</title>
          <year>1980</year>
          <rating>8.3</rating>
          <votes>62984</votes>
     </movie>

     <movie>
          <genre>horror</genre>
          <title>Rosemary's Baby</title>
          <year>1968</year>
          <rating>7.9<7rating>
          <votes>15676</votes>
     </movie>

     <movie>
          <genre>sci-fi</genre>
          <title>Serenity</title>
          <year>2005</year>
          <rating>8</rating>
          <votes>37184</votes>
     </movie>

     <movie>
          <genre>comedy</genre>
          <title>Life of Brian</title>
          <year>1979</year>
          <rating>8.1<7rating>
          <votes>40270</votes>
     </movie>

     <movie>
          <genre>horror</genre>
          <title>The movie that is as good
as Psycho</title>
          <year>2006</year>
          <rating>8.6</rating>
          <votes>63723</votes>
     </movie>

</movies>
```

XQUERY - példák

```
<?xml version="1.0" encoding="UTF-8"?>
<genres>
        <genre>horror</genre>
        <genre>fantasy</genre>
        <genre>sci-fi</genre>
        <genre>comedy</genre>
        <genre>drama</genre>
        <genre>romance</genre>
</genres>
```

1. Irjuk ki XQUERY segitsegevel, hogy mindenik mufajhoz hany film tartozik.

Genre	Movies
comedy	1
horror	6
sci-fi	2

2. Irjuk ki XQUERY segitsegevel a horror filmek sorszamat, cimet es ratingjet, rating szerinti csokkeno sorrendben.

No.	Title	Rating
2	Psycho	8.6
6	The movie that is as good as Psycho	8.6
4	The Shining	8.3
1	The Excorsit	8.0
5	Rosemary's Baby	7.9
3	Bram Stoker's Dracula	7.1

3. Irjuk ki XQUERY segitsegevel a horror filmeket sorszamozva, cimuket es ratingjet, rating szerinti csokkeno sorrendben.

No.	Title	Rating
1	Psycho	8.6
2	The movie that is as good as Psycho	8.6
3	The Shining	8.3
4	The Excorsit	8.0
5	Rosemary's Baby	7.9
6	Bram Stoker's Dracula	7.1

XQUERY - példák

4. Irjuk ki XQUERY segitsegevel a mufajonkenti legjobb filmeket; mufaj, cim es rating sorrendben.

Genre	Title	Rating
comedy	Life of Brian	8.1
horror	Psycho The movie that is as good as Psycho	8.6
sci-fi	Star Wars	8.8

5. Irjuk ki XQUERY segitsegevel, hogy mindenik mufajhoz hany film tartozik, ha egy sincs irjunk ki 0-t.

Genre	Movies
comedy	1
drama	0
fantasy	0
horror	6
romance	0
sci-fi	2

6. Irjuk ki XQUERY segitsegevel mufajonkent a filmeket; mufaj, rating es cim sorrendben, ha egy mufajhoz nem tartozik film irjuk ki, hogy <NO MOVIES>.

Genre	Rating	Movies
comedy	8.1	Life of Brian
drama	<NO MOVIES>	<NO MOVIES>
fantasy	<NO MOVIES>	<NO MOVIES>
horror	8.6	Psycho
		The movie that is as good as Psycho
romance	<NO MOVIES>	<NO MOVIES>
sci-fi	8.8	Star Wars