

XML lekérdezőnyelvek

XPath

XQuery

Az XPath/XQuery adatmodell

- ◆ A relációk megfelelője ebben a környezetben a *tételek (item) listája (sequence)*.
- ◆ Ez azt jelenti, hogy a bemenetet, a köztes lépések eredményeit és a végeredményt is tételek listájaként kezeljük.
- ◆ Egy *tétel* lehet:
 1. egyszerű érték, pl.: egész vagy sztring.
 2. Csomópont.

A csomópontok fő típusai

1. A *dokumentum csomópontok* a teljes dokumentumot reprezentálják.
2. *Elem csomópontok*: a tagek (jelölők) és a közöttük lévő dokumentumrészlet.
3. *Attribútumok*: a nyitó tagekben szerepelnek és ott kapnak értéket.

Öt tétel szekvenciája

```
10
"tíz"
10.0
<Számrendszer bázis = "8">
    <Digit>1</Digit>
    <Digit>2</Digit>
</Számrendszer>
@val="10"
```

- ◆ Az első három tétel egyszerű típusú (egész, szöveg, valós sz.)
- ◆ A negyedik elem csomópont típus
- ◆ Az utolsó egy attribútum csomópont típus

Dokumentum csomópont

- ◆ A doc(URL) vagy document(URL) parancs hatására jön létre.
- ◆ Példa: doc(<http://abc.com/sales.xml>)
- ◆ Minden XPath (és XQuery) lekérdezés hivatkozik egy dokumentum csomópontra.
 - ◆ Példa: az XML Sémában szereplő XPath kifejezések arra a dokumentumra hivatkoznak, amelyre a séma éppen vonatkozik.

A használt példa DTD-je

```
<!DOCTYPE kocsmák [  
  <!ELEMENT kocsmák (kocsma*, sör*)>  
  <!ELEMENT kocsma (ár+)>  
    <!ATTLIST kocsma név ID #REQUIRED>  
  <!ELEMENT ár (#PCDATA)>  
    <!ATTLIST ár melyikSör IDREF #REQUIRED>  
  <!ELEMENT sör EMPTY>  
    <!ATTLIST sör név ID #REQUIRED>  
    <!ATTLIST sör árulja IDREFS #IMPLIED>  
>
```

Példa dokumentum

<kocsmák>

Elem csomópont.

```
<kocsmma név = "Joe bárja">  
  <ár melyikSör = "Bud">2.50</ár>  
  <ár melyikSör = "Miller">3.00</ár>  
</kocsmma> ...
```

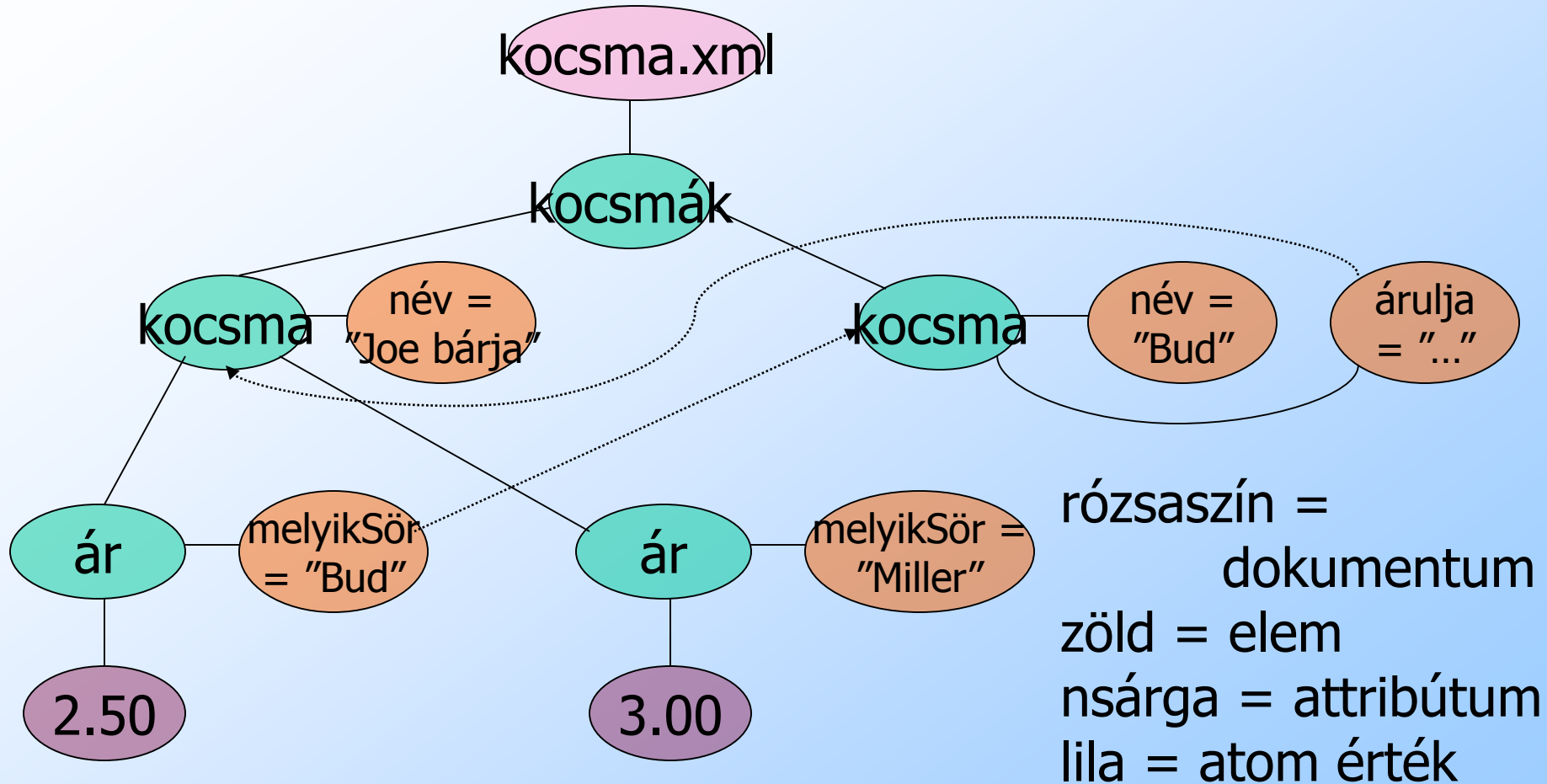
<sör név = "Bud" árulja = "Joe bárja
Sue bárja ... "> ...

Attribútum csomópont

</kocsmák>

A dokumentum csomópont mindez, plusz az <? xml version... rész.

A csomópontok típusa



Utak az XML dokumentumban

- ◆ Az XPath segítségével az XML dokumentumokat járhatjuk be. Más szóval utakat adhatunk meg.
- ◆ Az utak mindig tételek egy listáját választják ki.

Út kifejezések

- ◆ Egyszerű formájában az utak perjel és jelölők (tagek) sorozatából állnak. A kifejezés perjellel kezdődik.
 - ◆ Példa: /kocsmák/kocsma/ár
- ◆ **Informális jelentés:** a dokumentum csomópontból kiindulva balról-jobbra haladva kövessük a jelölőket sorra.

Útkifejezések kiértékelése

- ◆ Tegyük fel, hogy az első jelölő a gyökér.
 - ◆ Ennek a jelölőnek a feldolgozása a gyökér pontot tartalmazó, azaz egyelemű, sorozatot eredményezi.
- ◆ Tegyük fel, hogy rendelkezésünkre áll tételek egy sorozata és a következő jelölő X .
 - ◆ Minden elem pontot helyettesítsünk az X jelölőjű alelemeivel a listában.

Példa: /kocsmák

<kocsmák>

<kocsma név = "Joe bárja">

<ár melyikSör = "Bud">2.50</ár>

<ár melyikSör = "Miller">3.00</ár>

</kocsma> ...

<sör név = "Bud" árulja = "Joe bárja

Sue bárja ... "> ...

</kocsmák>

A kocsmák elem

Példa: /kocsmák/kocsma

<kocsmák>

<kocsma név = "Joe bárja">

<ár melyikSör = "Bud">2.50</ ár>

<ár melyikSör = "Miller">3.00</ ár>

</kocsma> ...

<sör név = "Bud" árulja = "Joe bárja

Sue bárja ..."/> ...

</kocsmák>

A kocsma elem, amit más kocsma elemek is követhetnek.

Példa: /kocsmák/kocsma/ár

<kocsmák>

<kocsma név = "Joe bárja">

<ár melyikSör = "Bud">2.50</ ár>

<ár melyikSör = "Miller">3.00</ár>

</kocsma> ...

<sör név = "Bud" árulja = "Joe bárja

Sue bárja ..."/> ...

</kocsmák>

ár elemek, melyeket más
kocsmák ár elemei követhetnek.

Attribútumok az utakban

- ◆ Az attribútumokat a @ jel jelöli, ez után következik az attribútum neve.

Példa:

/kocsmák/kocsmá/ár/@melyikSör

<kocsmák>

<kocsmá név = "Joe bárja">

<ár melyikSör = "Bud">2.50</ ár>

<ár melyikSör = "Miller">3.00</ár>

</kocsmá> ...

<sör név = "Bud" árulja = "Joe bárja

Sue bárja ..."/> ...

</kocsmák>

A "Bud" "Miller" értékek
belekerülnek az eredménybe.

Ne felejtsük: tételek listája

- ◆ Eddig a tételek mindig elemek voltak.
- ◆ Ha egy útkifejezés attribútummal végződik, akkor a lista atomi típusú elemekből áll (az előző példában sztringekből).

Utak, melyek akárhol kezdődhetnek

- ◆ Ha az út a dokumentum pontból indul, akkor a $//X$ útkifejezés megtalálja az összes X gyerekelemet a beágyazottság bármely szintjén, és visszaadja abban a sorrendben, ahogy az a dokumentumban megjelenik,
- ◆ illetve ha ez a útkifejezés folytatódik: $//X/...$, akkor vagy a gyökérből fog indulni vagy bármely olyan részelemből, aminek a jelölője X .

Példa: //ár

<kocsmák>

<kocsma név = "Joe bárja">

<ár melyikSör = "Bud">2.50</ ár>

<ár melyikSör = "Miller">3.00</ár>

</kocsma> ...

<sör név = "Bud" árulja = "Joe bárja

Sue bárja ..."/> ...

</kocsmák>

Ezek az ár elemek és bármely
másik ár eleme a
dokumentumnak.

Jolly-joker: *

- ◆ A csillag (*) tetszőleges jelölő nevet helyettesít.
- ◆ **Példa:** /*/*/ár az összes „dédunoka” (harmadik szinten lévő) ár elemet adja vissza.

Példa: /kocsmák/*

Ez a kocsmá elem és esetleg más kocsmá elemek, plusz ez a sör elem és esetleg más sör elemek.

<kocsmák>

```
<kocsmá név = "Joe bárja">  
  <ár melyikSör = "Bud">2.50</ ár>  
  <ár melyikSör = "Miller">3.00</ár>  
</kocsmá> ...
```

```
<sör név = "Bud" árulja = "Joe bárja  
  Sue bárja ..."/> ...
```

</kocsmák>

Szűrési feltétel

- ◆ A jelölőket feltételek [...] követhetik.
- ◆ Ilyenkor csak azok az útkifejezésre illeszkedő utak kerülnek be az eredménybe, melyek a feltételt is kielégítik.

Példa: szűrési feltétel

◆ /kocsmák/kocsmas/ár[. < 2.75]

<kocsmák>

<kocsmas név = "Joe bárja">

<ár melyikSör = "Bud">2.50</ ár>

<ár melyikSör = "Miller">3.00</ár>

Az aktuális
elem.

Csak ez az ár elem kerül be az
eredménybe a láthatók közül.

Példa: szűrés attribútummal

◆ /kocsmák/kocsmá/ár[@melyikSör = "Miller"]

<kocsmák>

<kocsmá név = "Joe bárja">

<ár melyikSör = "Bud">2.50</ ár>

<ár melyikSör = "Miller">3.00</ár>

Tengelyek

- ◆ Általánosan: az útkifejezésekben minden egyes lépésnél *tengelyekkel (axes)* adhatjuk meg a következő lépésnél feldolgozandó pontok listáját.
- ◆ A default tengely a **child::** --- ami az összes gyermeket veszi az aktuális pontoknak.

Példa: tengelyek

- ◆ /kocsmák/sör valójában a
/child::kocsmák/child::sör rövidítése.
- ◆ @ pedig az **attribute::** tengely
rövidítése.
 - ◆ Így a, /kocsmák/sör[@név = "Bud"]
jelentése
/kocsmák/sör[attribute::név = "Bud"]

További tengelyek

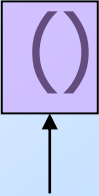
◆ Néhány további hasznos tengely:

1. **parent::** = az aktuális pont(ok) szülője (szülei).
2. **descendant-or-self::** = az aktuális pont(ok) és az összes leszármazott.
 - ◆ // valójában ennek a rövidítése (ez majdnem igaz).
3. **ancestor::**, **ancestor-or-self** éít.
4. **self** (ennek rövidítése: .)

XQuery

- ◆ Az XQuery egy SQL-hez hasonló lekérdezőnyelv, ami XPath kifejezéseket használ.
- ◆ Ugyanúgy a tételek listája adatmodellt használja.
- ◆ Az XQuery egy funkcionális nyelv.
 - ◆ Ez azt jelenti, hogy ahol kifejezés szerepelhet, ott tetszőleges XQuery kifejezés szerepelhet. Ez eltérés az SQL-től. Az SQL-ben a SELECT-nél nem 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 generál.
 - ◆ a **let** a cikluson belüli hozzárendeléseket adja meg.
- ◆ Minden iterációjánál a beágyazott ciklusnak, 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 végrehajtnak egyszer.

A korábbi példákban
is használt
dokumentum.

Példa: FOR

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

```
for $sor in  
  document("kocsmak.xml")/kocsmák/sör/@név  
return  
  <sörNév> { $sor } </sörNév>
```

- ◆ \$sor a példa dokumentum sör elemeinek név attribútumán fut végig.
- ◆ Az eredmény sörNév elemek listája:
 <sörNév>Bud</sörNév>
 <sörNév>Miller</sörNév> . . .

Kapcsos zárójelek

- ◆ 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 $x` értéke egyértelmű.
- ◆ Jelölők vagy idézőjelek nélküli sima sztringet nem adhat vissza a lekérdezés, azaz, ha `$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("kocsmák.xml")
```

```
let $sor := $d/kocsmák/sör/@név
```

```
return
```

```
<sörNév> {$sor} </sörNév>
```

◆ Az eredmény egyetlen elemből áll az összes sörnévvel:

```
<sörNév>Bud Miller ...</sörNév>
```

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 a változók minden hozzárendelésére kiértékelődik.
- ◆ A végeredmény listájának sorrendjén változtat.

Példa: Order-By

- ◆ A Bud összes árát kilistázzuk, a legalacsonyabb legelőször.

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

```
for $p in
```

```
$d/kocsmák/kocσμα/ár[@melyikSör="Bud"]
```

```
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.

Feltételek (predicates)

- ◆ A feltételekben a „**létezés**” követeljük meg.
- ◆ **Példa:** /kocsmák/kocsma[@név] jelentése “az összes olyan kocsma, aminek létezik neve.”
- ◆ **Példa:** /kocsmák/sör[@árulja = “Joe bárja”] azon sörök listáját adja vissza, melyeket Joe bárjában megkaphatunk.

Példa: összehasonlítások

- ◆ Az összes sörre, amit Joe bárjában árulnak, adjuk vissza az ár elemeket az összes kocsmát figyelembe véve.
- ◆ Az eredmény BBP elemekből áll majd, melynek attribútuma a kocsmá és a sör nevét adják majd meg, egy aleleme pedig az árat.


Stratégia

1. Készítsünk egy tripla for ciklust, ahol vesszük az összes **sör** elemet, aztán az összes **kocsmá** elemet, majd minden egyes kocsmára a hozzá tartozó **ár** elemeket.
2. Ellenőrizzük, hogy a sör kapható-e Joe bárjában, és hogy a sör neve és az aktuális **ár** elemben a **melyikSör** attribútum értéke egyezik-e.
3. A kívánt alakú eredmény megadása.

A lekérdezés

```
let $bars = doc("kocsmak.xml") / kocsmák
for $beer in $bars/sör
for $bar in $bars/kocisma
for $price in $bar/ár
where $beer/@árulja = "Joe bárja" and
    $price/@melyikSör = $beer/@név
return <BBP kocisma = {$bar/@név} sör =
    {$beer/@név}>{$price}</BBP>
```

Ez nem igazán hatékony.



„Szigorú” összehasonlítások

- ◆ Ha meg szeretnénk követelni, hogy az összehasonlított listák egyetlen elemet tartalmazzanak az alábbi összehasonlításokat kell alkalmaznunk:
 - ◆ eq, ne, lt, le, gt, ge.
- ◆ **Példa:** \$beer/@árulja eq "Joe bárja"
igaz, ha a beer változóhoz egyetlen elem lett hozzárendelve, melynek árulja attribútuma Joe bárja.

Elemek és értékek összehasonlítása

◆ Ha egy elemet hasonlítunk össze egy értékkel, akkor az elemnek a hozzátartozó értékét vesszük, ha az az érték atomi.

◆ **Példa:** `/kocsmák/kocsma[@név="Joe bárja"] /ár[@melyikSör="Bud"] eq "2.50"`

Elemek összehasonlítása

- ◆ Nem elegendő, ha két elem „ugyanolyannak tűnik”.
- ◆ Példa:

```
/kocsmák/kocsma[@név="Joe bárja"]/  
ár[@melyikSör="Bud"] eq  
/kocsmák/kocsma[@név="Sue bárja"]/  
ár[@melyikSör="Bud"]
```

hamis, még akkor is ha Joe és Sue ugyanannyit kérnek a Bud sörért.

Elemek összehasonlítása – (2)

- ◆ Egy elem csak önmagával tud egyenlő lenni.
- ◆ **Tudniillik**: az elemek valójában mutatók, amelyek a dokumentum megfelelő helyére mutatnak, azaz nem azonosak a szöveges tartalmukkal.

Az elemek adatainak kinyerése

- ◆ Tegyük fel, hogy elemek értékeit szeretnénk összehasonlítani nem pedig az elhelyezkedésüket az adott dokumentumban.
- ◆ Az E elem értékét a *data* függvény használatával kaphatjuk meg: $data(E)$.

Példa: data()

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

```
return <BBP kocasma = { $bar/@name }  
      sör = { $beer/@name }  
      ár = { data($price) } />
```

Ismétlődések kiszűrése

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

Példa: az összes különböző ár

```
return distinct-values (
```

```
let $bars = doc("kocsmák.xml")  
return $bars/kocsmák/kocσμα/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="Joe bárja"` TRUE, ha név attribútum értéke "Joe bárja".
2. `/kocsmák/kocsma[@név="Lórúgás"]` TRUE, ha létezik olyan koccsma, amely név attribútumának Lórúgás az értéke.

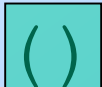
Logikai műveletek

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

Elágazó kifejezések

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

◆ Példa:

```
if ($koccsma/@név eq "Joe bárja")  
then $koccsma/ár else 
```

Az üres lista.

Kvantifikálás


some x in E_1 satisfies E_2

1. E_1 -t ki kell értékelni.
 2. x vegye fel sor E_1 eredményének értékeit, és értékeljük ki ezzel az 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 $bar in  
  doc("kocsmak.xml") / kocsmák / kocsmák  
where some $p in $bar / ár  
  satisfies $p < 2.00  
return $bar / @név
```



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

Példa: Every

```
for $bar in
  doc("kocsmak.xml") / kocsmák / kocσμα
where every $p in $bar / ár
  satisfies $p <= 5.00
return $bar / @név
```

Dokumentum sorrend

- ◆ A dokumentum sorrend szerinti összehasonlítás műveletei: << és >>.
- ◆ **Példa:** \$d/kocsmák/sör[@név="Bud"] << \$d/kocsmák/sör[@név="Miller"] igaz, ha a bal oldali sör 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.