XML

Document Type Definitions (DTD)

XML séma

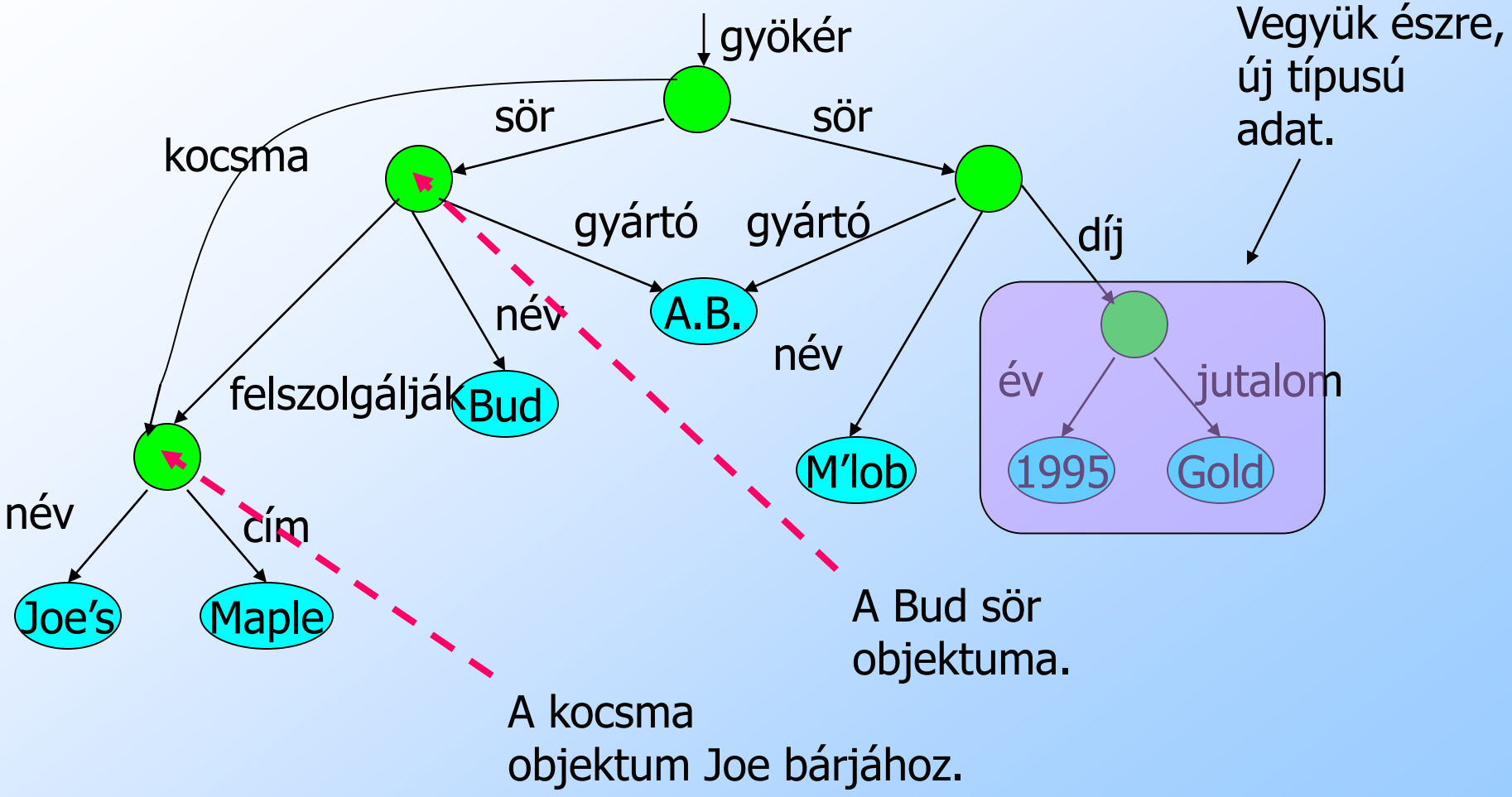
Féligstrukturált adat

- Egy másik, fákon alapuló adatmodell.
- Motiváció: az adatok rugalmas megjelenítése.
- Motiváció: *dokumentumok* megosztása rendszerek és adatbázisok között.

A féligstrukturált adatok gráfja

- Pontok = objektumok.
- Az élek címkéi tulajdonképpen az attribútumnevek, kapcsolatok.
- Az atomi értékek a levelekben tárolódnak.

Példa: adatgráf



XML

- XML = *Extensible Markup Language*.
- A HTML a dokumentumok kinézetét írja le (pl. rózsa “italic”), az XML-ben a jelentés, szemantika leírására használják a tageket (pl. “ez egy cím”).

XML dokumentumok

- A dokumentum *deklarációval* kezdődik, amit egy speciális <?xml ... ?> tag határol.
- Tipikusan:

```
<?xml version = “1.0” encoding
  = “utf-8” ?>
<?xml version = ”1.0”
  standalone = ”yes” ?>
```

  - “standalone” = “DTD-t csak validációra használjuk (yes) vagy nem (no, ez a default)”

Tagek

- A tagek, mint a HTML esetében is nyitó és záró elemből állnak <FOO> … </FOO>.
  - Opcionálisan <FOO/>.
- Tageket tetszőlegesen egymásba ágyazhatjuk.
- Az XML tagek érzékenyek a kis- és nagybetű különbségre.

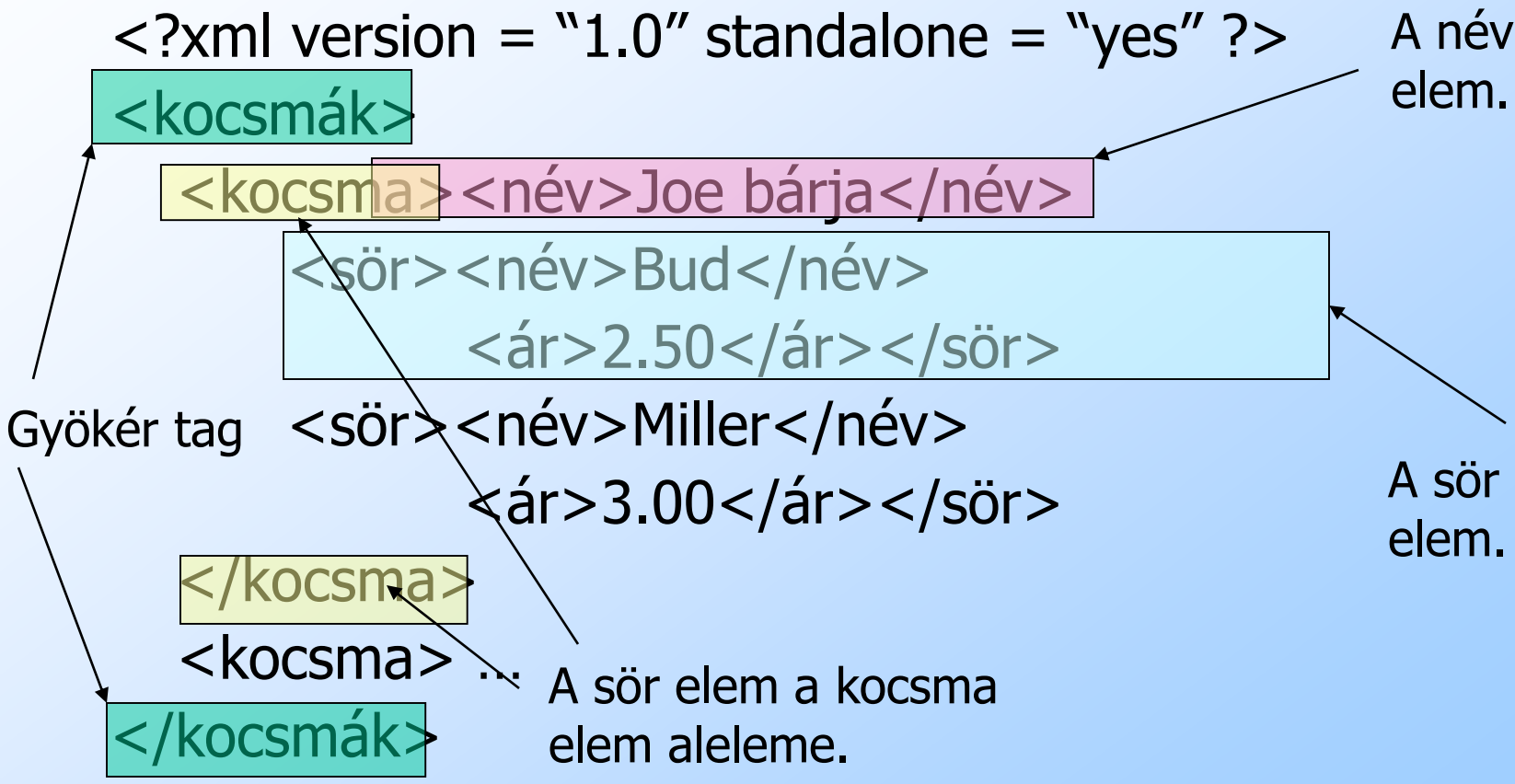
Jól formált és valid XML

- *Jól formált XML* megengedi, hogy önálló tageket vezessünk be.
- *Valid XML* illeszkedik egy előre megadott sémára: DTD vagy XML séma.

Jól formált XML

- Nem hiányzik a *deklaráció*: <?xml … ?>.
- Minden nyitó tagnek megvan a záró párja.

Példa: jól formázott XML



DTD felépítése

<!DOCTYPE *gyökér elem* [

  <!ELEMENT elem *név* (*összetevők*) >

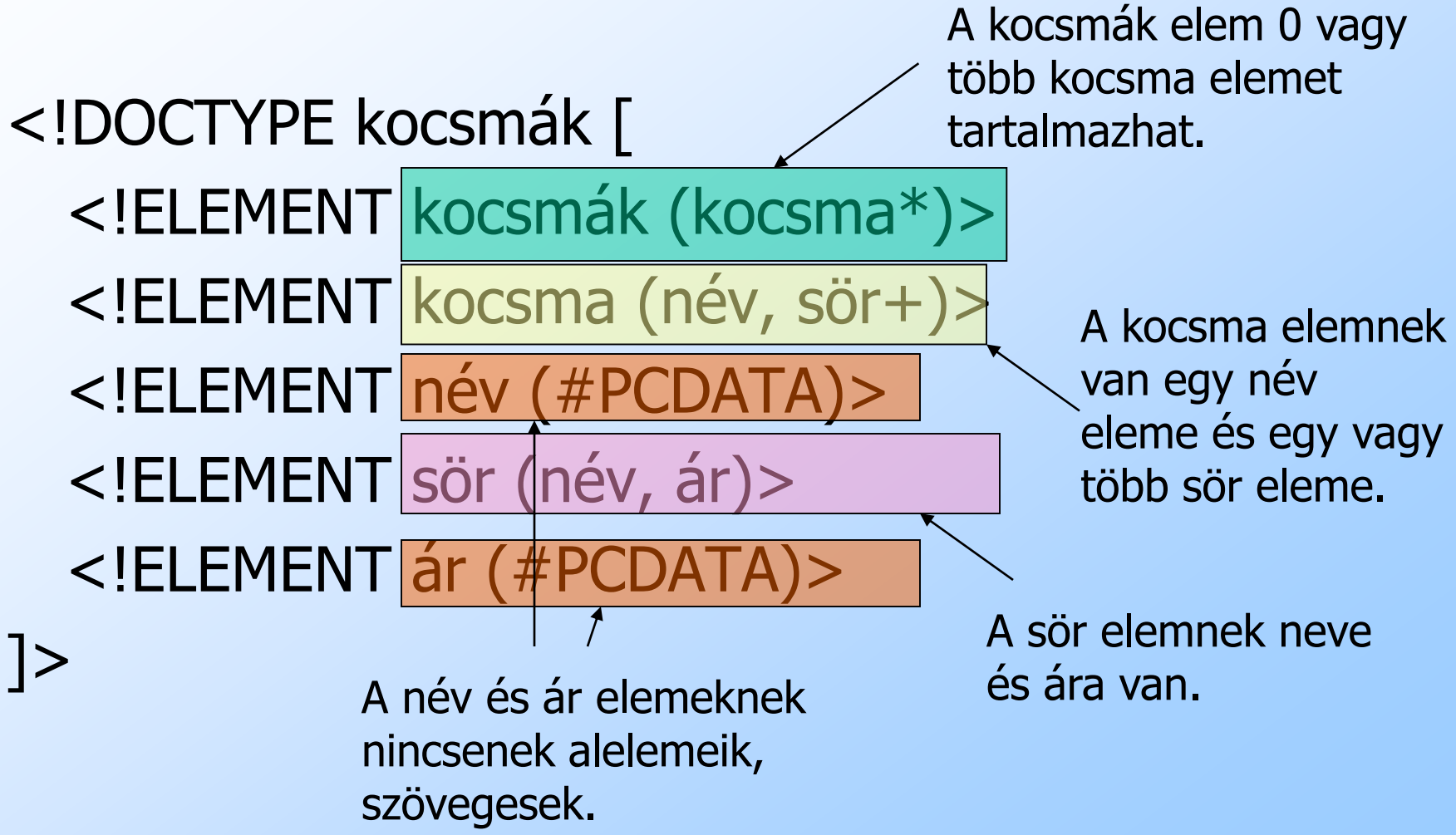
  . . . *további elemek* . . .

]>

DTD ELEMENT

- Egy-egy elem leírása az elem nevét és zárójelek között az alelemek megadását jelenti.
  - Ez magában foglalja a alelemek sorrendjét és multiplicitását.
- A levelek (szöveges elemek) típusa #PCDATA (*Parsed Character DATA*).

Példa: DTD


<!DOCTYPE kocsmák [
<!ELEMENT kocsmák (kocsma*)>
<!ELEMENT kocsma (név, sör+)>
<!ELEMENT név (#PCDATA)>
<!ELEMENT sör (név, ár)>
<!ELEMENT ár (#PCDATA)>
]>
A kocsmák elem 0 vagy több kocsma elemet tartalmazhat.
A kocsma elemnek van egy név eleme és egy vagy több sör eleme.
A sör elemnek neve és ára van.
A név és ár elemeknek nincsenek alelemeik, szövegesek.

Elem leírások

- Az alelemek a felsorolás sorrendjében kell, hogy kövessék egymást.
- Egy elemet a felsorolásban egy további szimbólum követhet megadva a multiplicitását.
  - * = 0 vagy több.
  - + = 1 vagy több.
  - ? = 0 vagy 1.
- A | szimbólum jelentése: vagy.

Példa: elem leírás

- A névhez opcionálisan hozzátartozik egy titulus, egy kereszt- és vezetéknév (ebben a sorrendben), vagy egy IP cím:

```
<!ELEMENT név (
  (titulus?, kereszt, vezeték) |
  IP cím
)>
```

Példa: DTD-k használata (a)

```
<?xml version = "1.0" standalone = "no" ?>
<!DOCTYPE kocsmák [
    <!ELEMENT kocsmák (kocsma*)>
    <!ELEMENT kocsma (név, sör+)>
    <!ELEMENT név (#PCDATA)>
    <!ELEMENT sör (név, ár)>
    <!ELEMENT ár (#PCDATA)>
]>
```

a DTD

a dokumentum

```
<kocsmák>
    <kocsma><név>Joe bárja</kocsma>
        <sör><név>Bud</név> <ár>2.50</ár></sör>
        <sör><név>Miller</név> <ár>3.00</ár></sör>
    </kocsma>
    <kocsma> …
</kocsmák>
```

Példa: DTD-k használata (b)

- Feltesszük, hogy a kocsmák DTD a bar.dtd fájlban van.

```
<?xml version = "1.0" standalone = "no" ?>
<!DOCTYPE kocsmák SYSTEM "bar.dtd">
<kocsmák>
    <kocsma><név>Joe bárja</név>
        <sör><név>Bud</név>
            <ár>2.50</ár></sör>
        <sör><név>Miller</név>
            <ár>3.00</ár></sör>
    </kocsma>
    <kocsma> …
</kocsmák>
```

A DTD-t bar.dtd fájlból nyeri.

Attribútumok

- Az XML-ben a nyitó tag-ek mellett szerepelhetnek *attribútumok*.
- Egy DTD-ben,

`<!ATTLIST` *E* `...>`

Az *E* taghez tartozó attribútumokat adja meg.

Példa: attribútumok

- A kocsma elemhez hozzátartozik egy `típus` attribútum.

```
<!ELEMENT kocsma (név sör*)>
  <!ATTLIST kocsma típus CDATA
     #IMPLIED>
```

Sztring.

Az attribútum opcionális,
ennek ellentéte: #REQUIRED

Példa: attribútum használatára

```
<kocsma típus = ”skót”>
  <név>McGuiver sörözője</név>
  <sör><név>Scottish ale</név>
          <ár>5.00</ár></sör>
  ...
</ kocsma>
```

ID és IDREF attribútumok

- Az attribútumok segítségével egyik elemről a másikra mutathatunk.
- Emiatt az XML dokumentum reprezentációja inkább gráf, mintsem egyszerű fa (fenntartásokkal kell fogadni ezt az információt).

ID-k létrehozatala

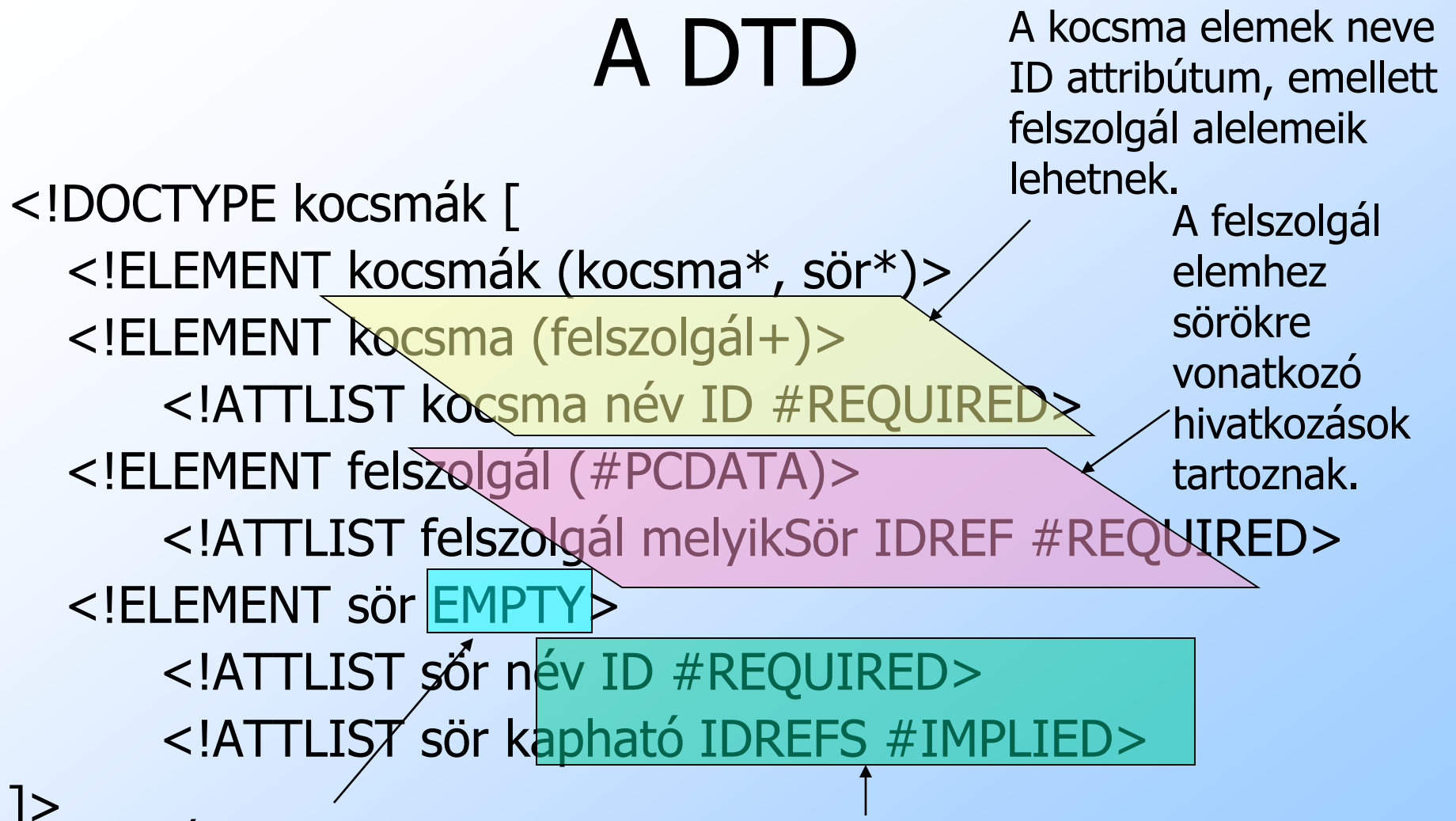
- Adjuk meg egy *E* elemet és egy *A* attribútumát, aminek típusa: ID.
- Amikor az E elemet (<*E*>) egy XML dokumentumba használjuk, az *A* attribútumnak egy máshol nem szereplő értéket kell adnunk.
- Példa:

```
<E   A = ”xyz”>
```

IDREF-ek létrehozatala

- Egy *F* elem az IDREF attribútum segítségével hivatkozhat egy másik elemre annak ID attribútumán keresztül.
- Vagy: az IDREFS típusú attribútummal több másik elemre is hivatkozhat.

A DTD


A kocsma elemek neve ID attribútum, emellett felszolgál alelemeik lehetnek.
A felszolgál elemhez sörökre vonatkozó hivatkozások tartoznak.
<!DOCTYPE kocsmák [
<!ELEMENT kocsmák (kocsma*, sör*)>
<!ELEMENT kocsma (felszolgál+)>
<!ATTLIST kocsma név ID #REQUIRED>
<!ELEMENT felszolgál (#PCDATA)>
<!ATTLIST felszolgál melyikSör IDREF #REQUIRED>
<!ELEMENT sör EMPTY>
<!ATTLIST sör név ID #REQUIRED>
<!ATTLIST sör kapható IDREFS #IMPLIED>
]>
lásd később
A sör elemeknek is van egy ID típusú attribútuma (név), a kapható attribútum pedig kocsmák nevét tartalmazza.

Példa: a DTD-re illeszkedő dokumentum

```
<kocsmák>
   <kocsma név = ”Joe bárja”>
        <felszolgál melyikSör = ”Bud”>2.50</felszolgál>
        <felszolgál melyikSör = ”Miller”>3.00
        </felszolgál>
   </kocsma> …
   <sör név = ”Bud” kapható = ”Joe bárja
        Suzy bárja …” /> …
   </kocsmák>
```

Példa: üres elem

- A DTD:

<!ELEMENT felszolgál EMPTY>

<!ATTLIST felszolgál melyikSör IDREF #REQUIRED>

<!ATTLIST felszolgál ár CDATA #REQUIRED>

- Példa a használatra:

<felszolgál melyikSör = ”Bud” ár = ”2.50” />

a záró tag itt lerövidül

XML séma

- Az XML sémák segítségével szintén XML dokumentumok szerkezetét adhatjuk meg. Itt több megszorítást lehet előírni, mint a DTD-k esetén.
- Az XML séma maga is egy XML dokumentum.

Az XML séma dokumentum szerkezete

```
<? xml version = … ?>
<xs:schema xmlns:xs =
  ”http://www.w3.org/2001/XMLSchema”>
                    . . .
</xs:schema>
```

A schema tehát az xs névtérhez tartozó elem.

Az ”xs” *névteret* adja meg, amit a megadott URL azonosít. Itt a névtérhez tartozó elemek leírása is megtalálható.

Az xs:element elem

- Az attribútumai:
  1. name = a definiált elem neve (tagben miként szerepel).
  2. type = az elem típusa.
     - Lehet XML séma típus, pl. xs:string.
     - Vagy egy olyan típus, amit az adott XML sémában deklarálunk.

Példa: xs:element

```
<xs:element name = ”név”
     type = ”xs:string” />
```

EMPTY

- A következő elemet írja le:

  <név>Joe bárja</név>

Séma definíció szinten üres
Dokumentumban <tag> között

Összetett típusok

- alelemeket tartalmazó elemek leírásához xs:complexType használáljuk.
  - A name attribútummal nevet adhatunk ennek a típusnak.
- Az xs:complexType egy tipikus aleleme az xs:sequence, amihez xs:element elemek egy sorozata tartozik.
  - A minOccurs és maxOccurs attribútumok használatával az adott xs:element előfordulásainak számát korlátozhatjuk.

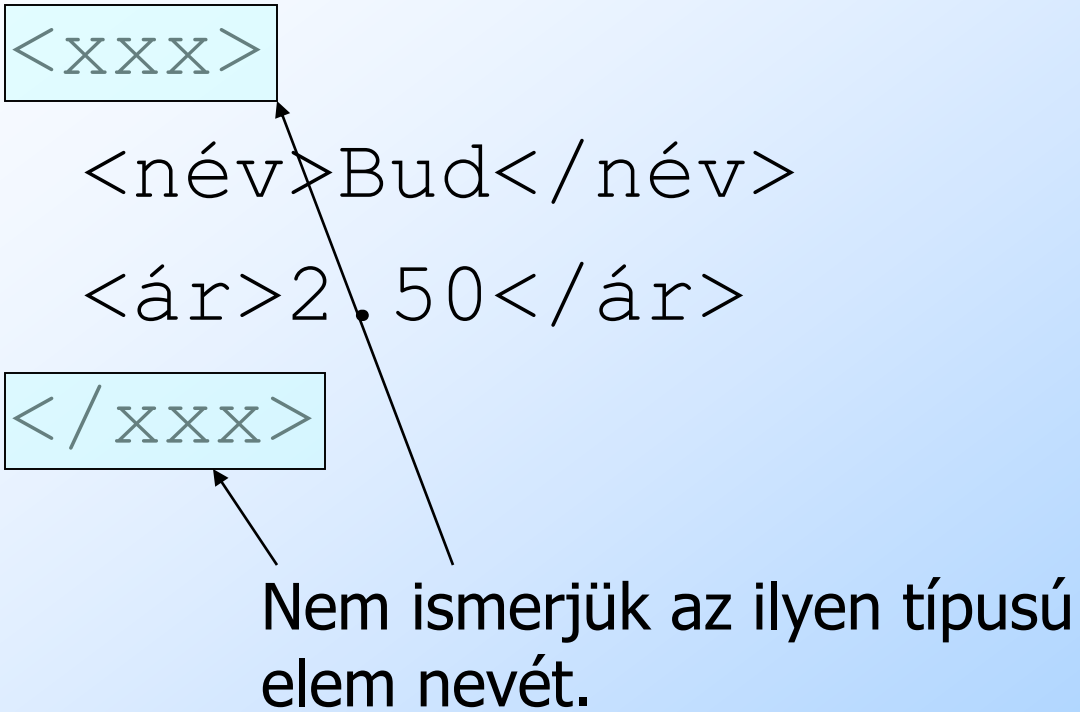
Példa: típus a sörökhöz

```
<xs:complexType name = ”sörTípus”>
  <xs:sequence>
    <xs:element name = ”név”
      type = ”xs:string”
      minOccurs = ”1” maxOccurs = ”1” />
    <xs:element name = ”ár”
      type = ”xs:float”
      minOccurs = ”0” maxOccurs = ”1” />
  </xs:sequence>
</xs:complexType>
```

Pontosan egy előfordulás.

Mint a ? a DTD-ben.

Egy sör típusú elem



Példa: típus a kocsmákhoz

```
<xs:complexType name = ”kocsmaTípus”>
  <xs:sequence>
    <xs:element name = ”név”
      type = ”xs:string”
      minOccurs = ”1” maxOccurs = ”1” />
    <xs:element name = ”sör”
      type = ”sörTípus”
      minOccurs = ”0” maxOccurs =
          ”unbounded” />
  </xs:sequence>
</xs:complexType>
```

Mint a * a DTD-ben.

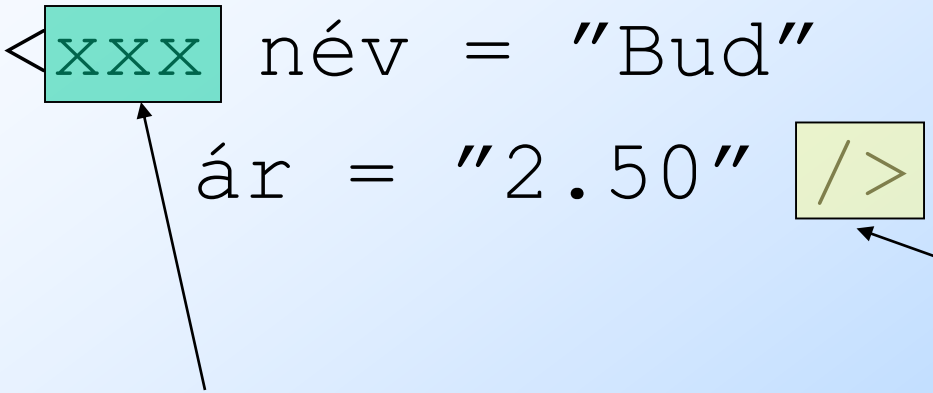
xs:attribute

- xs:attribute elemek használatával az összetett típuson belül a típushoz tartozó elemek attribútumait adhatjuk meg.
- Az xs:attribute elem attribútumai:
  - name és type mint az xs:element esetén.
  - use = ”required” vagy ”optional”.

Példa: xs:attribute

```
<xs:complexType név = ”sörTípus”>
  <xs:attribute name = ”név”
     type = ”xs:string”
     use = ”required” />
  <xs:attribute name = ”price”
     type = ”xs:float”
     use = ”optional” />
</xs:complexType>
```

Az új sörTípusnak megfelelő elem



Továbbra sem tudjuk az elem nevét.

Üres elemmel van dolgunk, mert nincsenek alelemei.

Egyszerű típus megszorítások

- Az xs:simpleType segítségével felsorolásokat adhatunk meg és az alaptípusra vonatkozó megszorításokat.
- name attribútuma van és
- xs:restriction aleleme.

Megszorítások: xs:restriction

- A base attribútum adja meg, hogy melyik egyszerű típusra (simple type) vonatkozik a megszorítás: pl. xs:integer.
- xs:{min, max}{Inclusive, Exclusive} a négy attribútum használatával alsó és felső korlátokat adhatunk meg.
- xs:enumeration alelem, a value attribútuma után megadhatjuk a felsorolás elemeit.

Példa: engedély attribútum a kocsmákhoz

```
<xs:simpleType name = ”engedély”>
  <xs:restriction base = ”xs:string”>
    <xs:enumeration value = ”minden” />
    <xs:enumeration value = ”csak sör” />
    <xs:enumeration value = ”csak bor” />
  </xs:restriction>
</xs:simpleType>
```

Példa: az árak [1,5) intervallumba eshetnek

```
<xs:simpleType name = ”ár”>
  <xs:restriction
     base = ”xs:float”
     minInclusive = ”1.00”
     maxExclusive = ”5.00” />
</xs:simpleType>
```

Kulcsok az XML sémában

- Az xs:element elemhez tartozhat xs:key alelem.
- Jelentése: ezen az elemen belül, minden elem, ami egy adott *szelektor* ösvényen keresztül elérhető egyedi értékekkel kell, hogy rendelkezzen a megadott mezőinek (*field*) kombinációin (alelem, attribútum).
- Példa: egy kocsmák elemen belül a kocsma elemek név attribútumának egyedinek kell lennie.

Példa: kulcs

```
<xs:element name = ”kocsmák” … >
    . . .
  <xs:key name = ”kocsmaKulcs”>
     <xs:selector xpath = ”kocsma”/>
     <xs:field xpath = ”@név” />
  </xs:key>
    . . .
</xs:element>
```

A @ attribútumot jelöl.

Az XPath segítségével az XML fákat járhatjuk be (következő óra).

Idegen kulcsok

- A xs:keyref alelem az xs:element elemen belül előírja, hogy ezen az elemen belül bizonyos értékek, melyeket ugyanúgy a szelektor és mezők használatával adhatunk meg, egy kulcs értékei között kell, hogy szerepeljenek.

Példa: idegen kulcs

- Tegyük fel, hogy a név alelem kulcs a kocsma elemekre vonatkozóan.
  - A kulcs neve legyen *kocsmaKulcs*.
- Az alkeszek elemhez *látogat* alelemeket szeretnénk hozzáadni. Ezen elemek *kocsma* attribútuma *idegen kulcs* lesz, a *kocsma* elem *név* alelemére hivatkozik.

Példa: idegen kulcs az XML sémában

```
<xs:element name = ”alkeszek”
      . . .
  <xs:keyref name = ”kocsmaRef”
     refers = ”kocsmaKulcs”>
     <xs:selector xpath =
          ”alkeszek/látogat” />
     <xs:field xpath = ”@kocsma” />
  </xs:keyref>
</xs:element>
```

Framework - Keretrendszer

1. *Information Integration* : Making databases from various places work as one.
2. *Semistructured Data* : A new data model designed to cope with problems of information integration.
3. *XML* : A standard language for describing semistructured data schemas and representing data.

1. *Információ integráció*: A különböző helyekről származó adatbázisokat úgy üzemeltetni, mintha egységes egészet alkotnának.
2. *Félig-strukturált adat*: Viszonylag új adatmodell, amely segít megbirkózni az adatintegráció problémájával.
3. *XML*: Szabványos nyelv a félig-strukturált adatok leírására.

The Information-Integration Problem

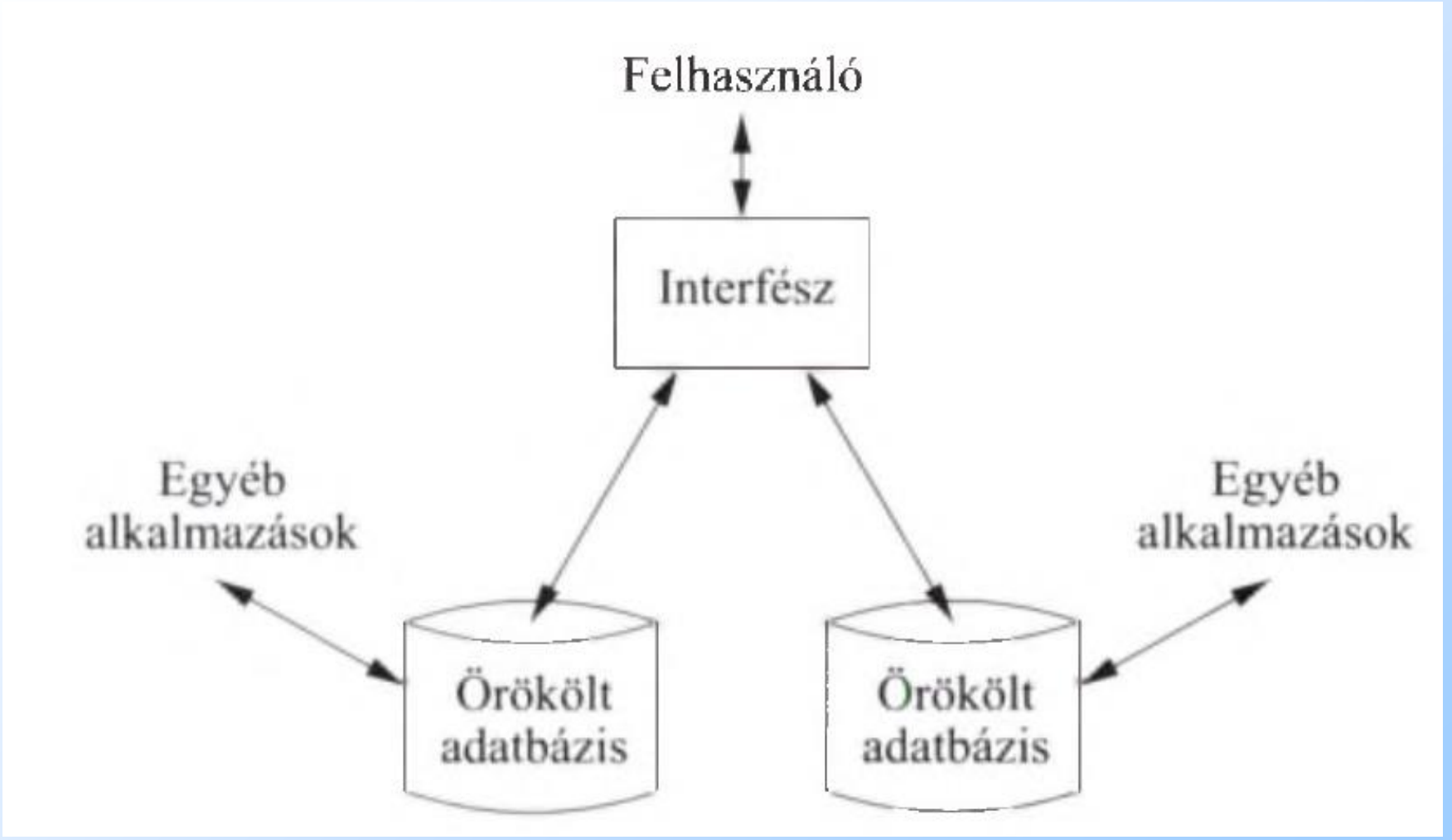
- Related data exists in many places and could, in principle, work together.
- But different databases differ in:
  1. Model (relational, object-oriented?).
  2. Schema (normalized/unnormalized?).
  3. Terminology: are consultants employees? Retirees? Subcontractors?
  4. Conventions (meters versus feet?).

- Egymáshoz kapcsolható adat elemek sok helyen léteznek és elvileg együtt működésre alkalmasak volnának:
- De a különböző adatbázisok több tekintetben különböznek:
  1. Adatbázis modellek (relációs, objektum-orientált, NoSQL, dokumentum stb.)
  2. Séma (normalizált, nem normalizált)
  3. Szakkifejezések: tanácsadó alkalmazott-e? Visszavonult nyugdíjas-e? Alvállalkozó?
  4. Konvenciók (méter kontra láb [metrikus (SI, CGI), birodalmi].

Example

- Every bar has a database.
  - One may use a relational DBMS; another keeps the menu in an MS-Word document.
  - One stores the phones of distributors, another does not.
  - One distinguishes ales from other beers, another doesn't.
  - One counts beer inventory by bottles, another by cases.

- Mindegyik kocsmának van adatbázisa:
  - Az egyik relációs adatbáziskezelőt használ; másik MS-Word-ben tartja nyilván a menüt.
  - Az egyik nyilvántartja a mobil telefonok forgalmazóit, a másik nem.
  - Az egyik megkülönbözteti az angol barna sört a többi sörtől, a másik nem.
  - Az egyik a leltárban a söröket palackokként tartja nyilván, másik a göngyöleg egységei alapján.

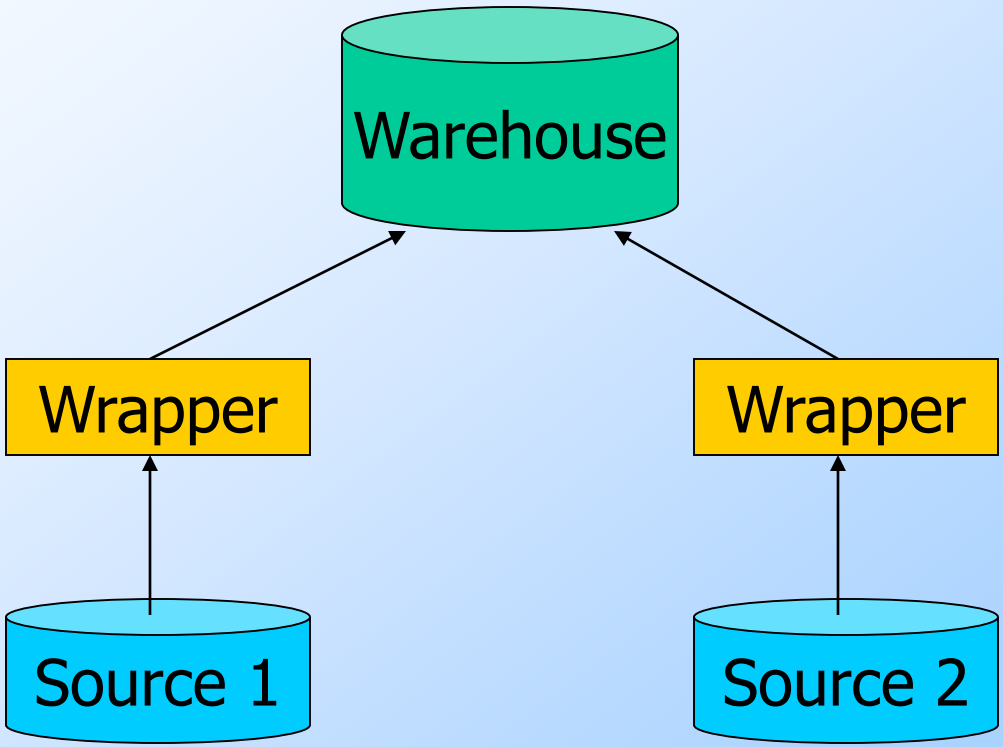

Felhasználó
Interfész
Egyéb alkalmazások
Egyéb alkalmazások
Örökölt adatbázis
Örökölt adatbázis

Two Approaches to Integration

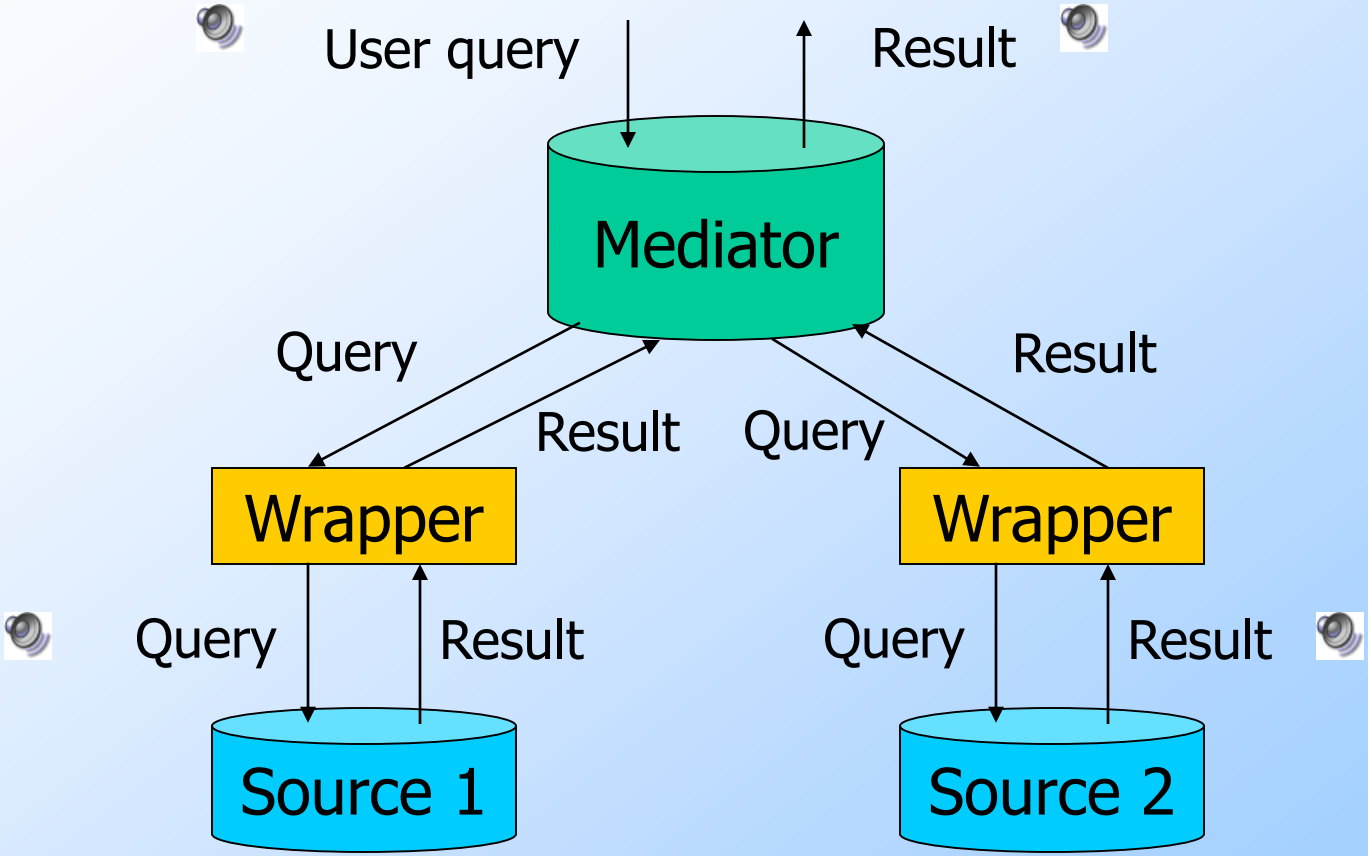
1. *Warehousing* : Make copies of the data sources at a central site and transform it to a common schema.
   - Reconstruct data daily/weekly, but do not try to keep it more up-to-date than that.
2. *Mediation* : Create a view of all sources, as if they were integrated.
   - Answer a view query by translating it to terminology of the sources and querying them.

1. Adattárház: Az adatforrásokról egy központi másolatot készít, és egy közös adat sémává transzformálja.
   - Az adatokat naponta, hetente frissítik, de ennél nem szabad nagyobb pontosságot megcélozni.
2. Mediáció, közvetítés: Az összes adatforrásra egy nézetet kell létrehozni, mintha egy integrált rendszer részei volnának:
   - A nézetre vonatkozó lekérdezést úgy lehet megválaszolni, hogy a lekérdezést az egyes adatforrások szakkifejezéseire fordítják le és azután kérdezik le az eredeti adatforrásokat

Warehouse Diagram



A Mediator


User query
Result
Mediator
Query
Result
Result
Query
Wrapper
Wrapper
Query
Result
Query
Result
Source 1
Source 2