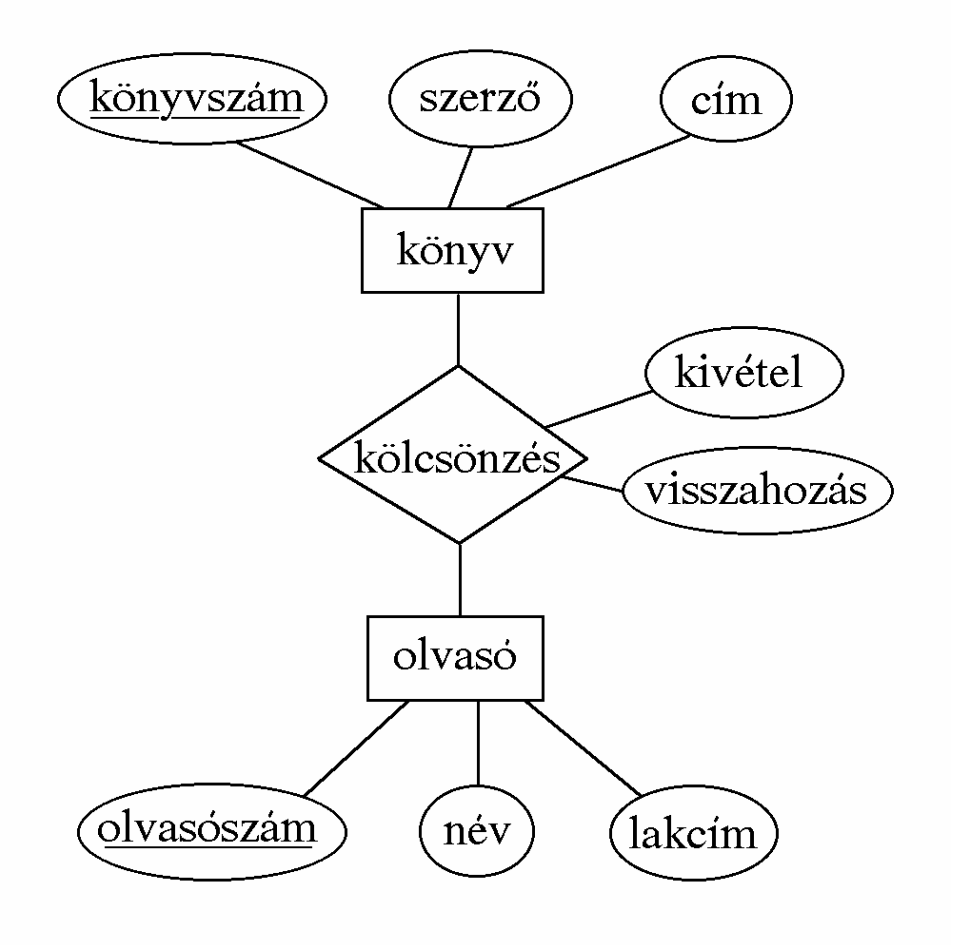
Adatmodellezés, adatbázis-tervezés

Az adatbázis-tervezés lépései:

1. a feldolgozandó információ elemzése,
2. az információk közti kapcsolatok meghatározása,
3. az eredmény ábrázolása (E/K diagram),
4. adatbázisterv készítése (transzformációs lépés),
5. adatbázisterv finomítása (összevonások),
6. megszorítások modellezése, függőségek meghatározása,
7. optimális adatbázisterv készítése (dekomponálás, normalizálás),
8. az adatbázisterv megvalósítása SQL-ben
   (create table…, create view …, stb.).

Egy könyvtár adatmodellje



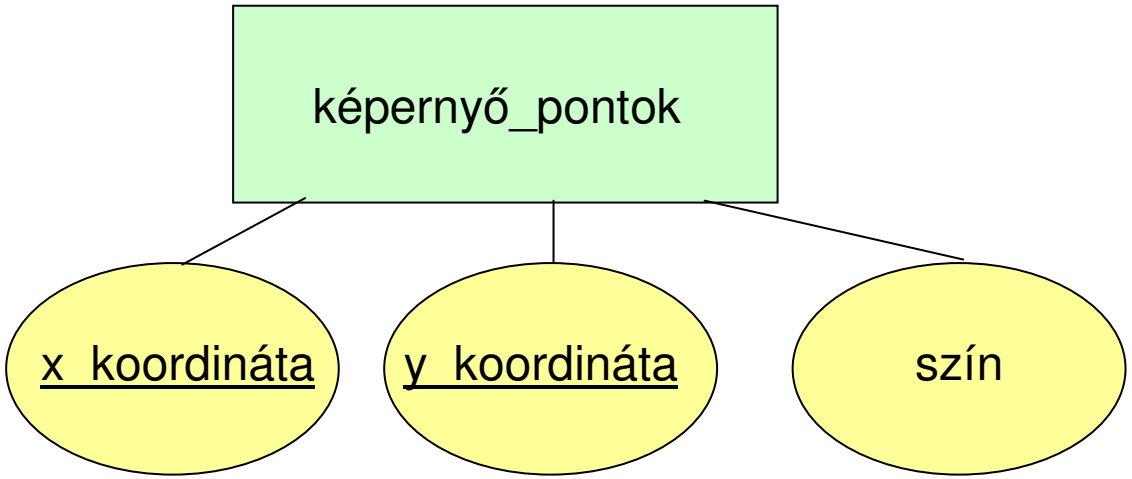
Egyed-kapcsolat formális modell

- **E(A1,…,An) egyedosztály séma,**
  - **E az egyedosztály neve,**
  - **A1,…,An tulajdonságok,**
  - **DOM(Ai) – lehetséges értékek halmaza.**
  - **például: tanár(név, tanszék).**

- **E(A1,…,An) sémájú egyedosztály előfordulása:**
  - **E = {e1,…,em} egyedek (entitások) halmaza, ahol**
    - **ei(k)∈ DOM(Ak),**
    - **semelyik két egyed nem egyezik meg minden attribútumban (az összes tulajdonság szuperkulcs),**
    - **minimális szuperkulcs = kulcs.**

Egyed-kapcsolat (E/K) diagram

- **Egyedosztályok, kapcsolatok, típusok, egyéb feltételezések ábrázolása.**
- **egyedosztály**
  - **az elsődleges (szuper)kulcshoz tartozó tulajdonságokat aláhúzzuk.**



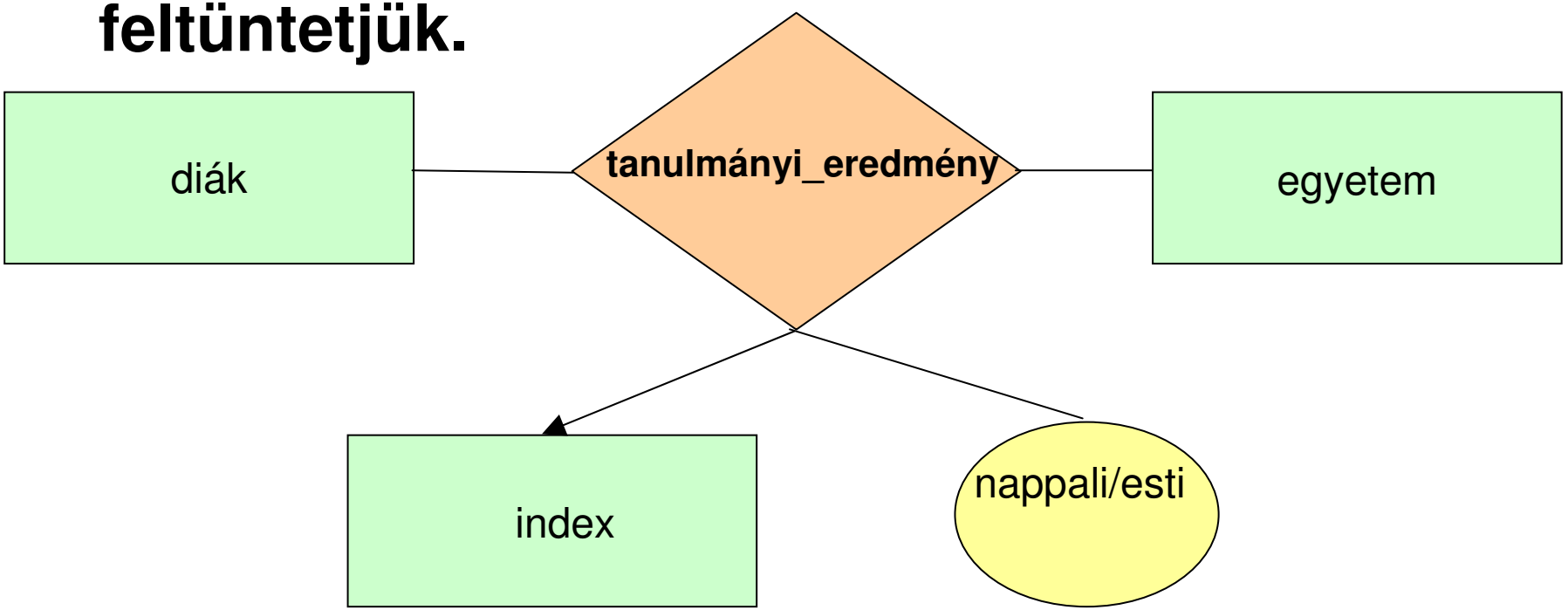
Egyed-kapcsolat formális modell

- **K(E1,…,Ep) a kapcsolat sémája,**
  - **K a kapcsolat neve,**
  - **E1,…,Ep egyedosztályok sémái,**
  - **p=2 bináris kapcsolat, p>2 többágú kapcsolat,**
  - **például: tanít(tanár,tárgy).**

- **K(E1,…,Ep) sémájú kapcsolat előfordulása:**
  - **K = {(e1,…,ep)} egyed p-esek halmaza, ahol**
    - **ei∈ Ei,**
    - **a kapcsolat előfordulásaira tett megszorítások határozzák meg a kapcsolat típusát.**

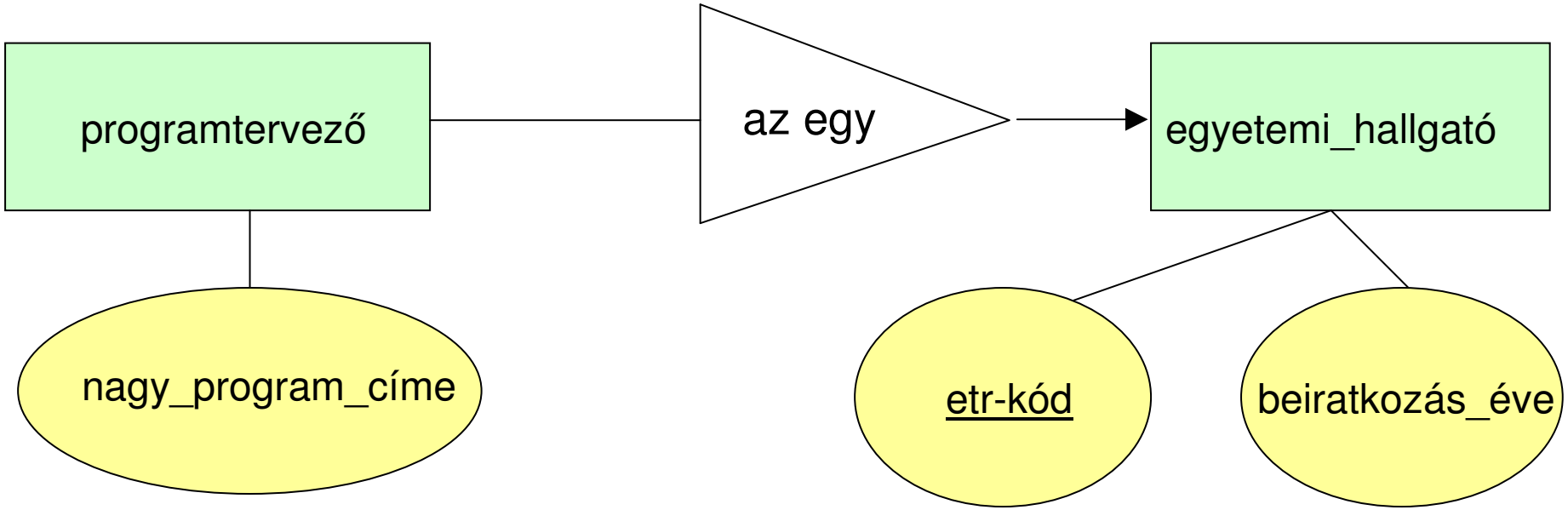
Egyed-kapcsolat (E/K) diagram

- **kapcsolat**
  - **a típusban szereplő "egy" végponthoz nyilat húzunk, a kapcsolat saját tulajdonságait is feltüntetjük.**



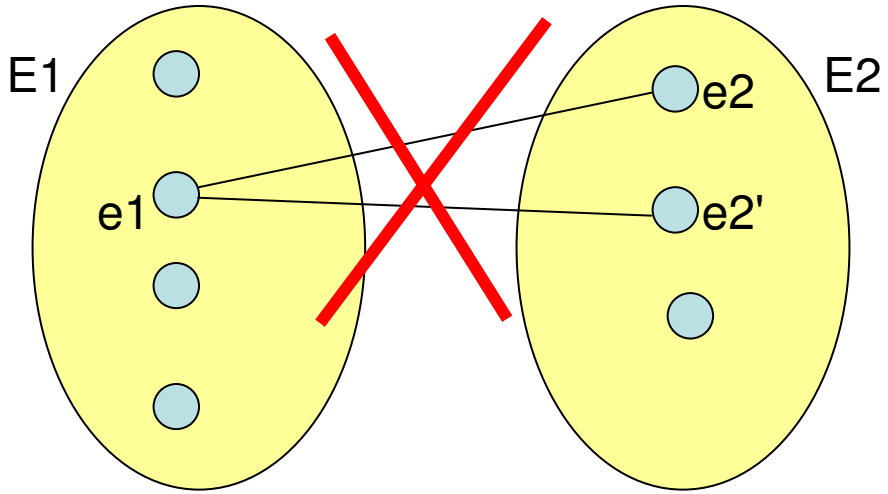
Egyed-kapcsolat (E/K) diagram

- **kapcsolat**
  - **öröklődési típus esetén az általánosabb egyedosztály felé húzunk nyilat, és a speciális osztálynak (alosztálynak) csak az általánostól különböző tulajdonságait adjuk meg.**



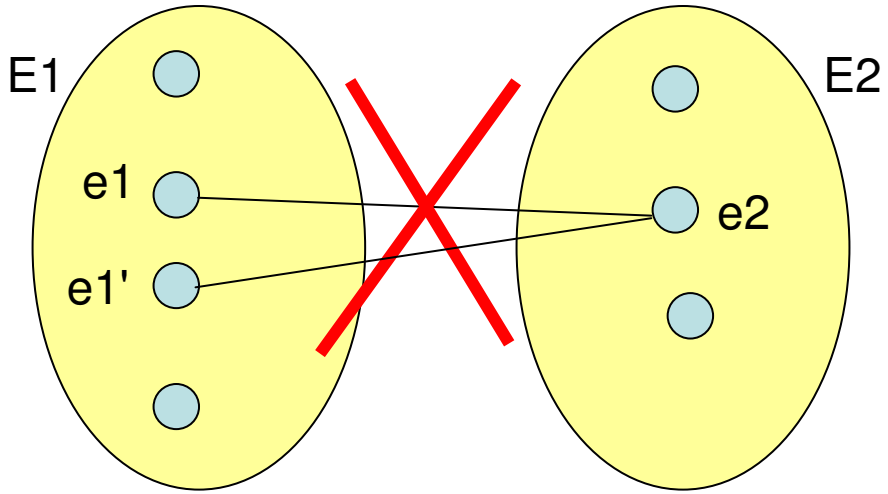
Kapcsolatok típusai

- **K(E1,E2) bináris kapcsolat,**
  - **sok-egy (n:1)**
    - **K {(ei,ej)} alakú előfordulásaiban nem szerepelhet egyszerre (e1,e2) és (e1,e2'), ha e2 és e2' különböznek,**
    - **másképpen: K előfordulásaiban minden E1-beli egyedhez legfeljebb 1 E2-beli egyed tartozhat,**
    - **például: született(név,ország).**



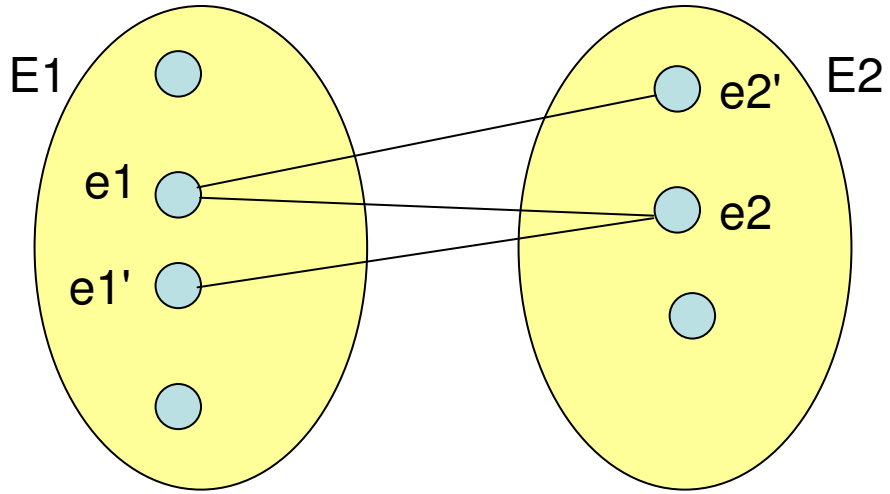
Kapcsolatok típusai

- **K(E1,E2) bináris kapcsolat,**
  - **egy-sok (1:n) (vagy (1:m)),**
    - **K {(ei,ej)} alakú előfordulásaiban nem szerepelhet egyszerre (e1,e2) és (e1',e2), ha e1 és e1' különböznek,**
    - **másképpen: K előfordulásaiban minden E2-beli egyedhez legfeljebb 1 E1-beli egyed tartozhat,**
    - **például: vb_győztes(ország,rendező_ország).**



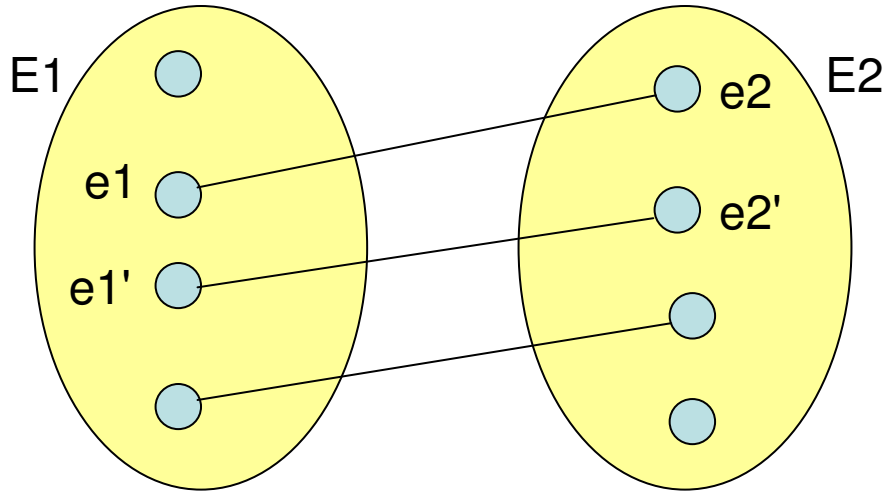
Kapcsolatok típusai

- **K(E1,E2) bináris kapcsolat,**
  - **sok-sok (n:m),**
    - **K {(ei,ej)} alakú előfordulásai nincsenek korlátozva,**
    - **előfordulhat (de nem kötelező, hogy előforduljon) az ábrán látható helyzet, vagyis minden E1-beli egyedhez több E2-beli egyed tartozhat, és fordítva, minden E2-beli egyedhez több E1-beli egyed tartozhat,**
    - **például: tanul(diák,nyelv).**



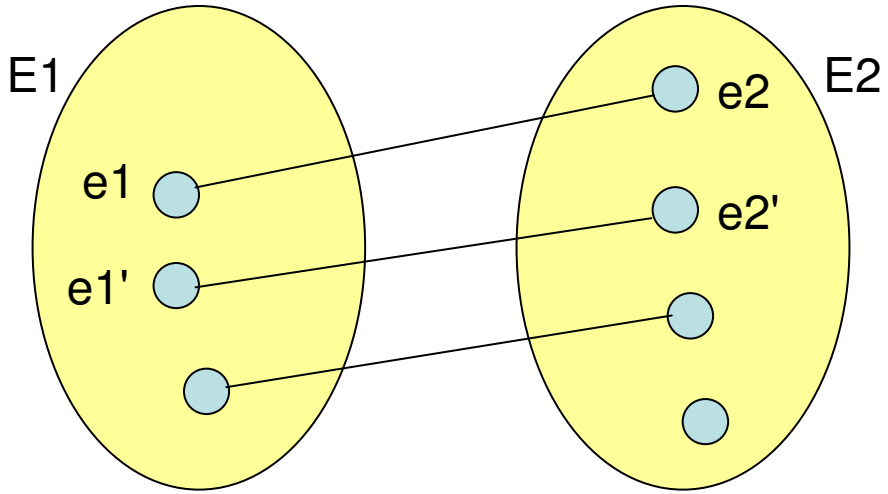
Kapcsolatok típusai

- **K(E1,E2) bináris kapcsolat,**
  - **egy-egy (1:1),**
    - **K {(ei,ej)} alakú előfordulásai egyszerre sok-egy és egy sok típusúak, vagyis minden E1-beli egyedhez legfeljebb egy E2-beli egyed tartozhat, és fordítva, minden E2-beli egyedhez legfeljebb egy E1-beli egyed tartozhat,**
    - **nem kötelezően szerepel minden egyed a kapcsolatban,**
    - **például: házaspár(férfi,nő).**

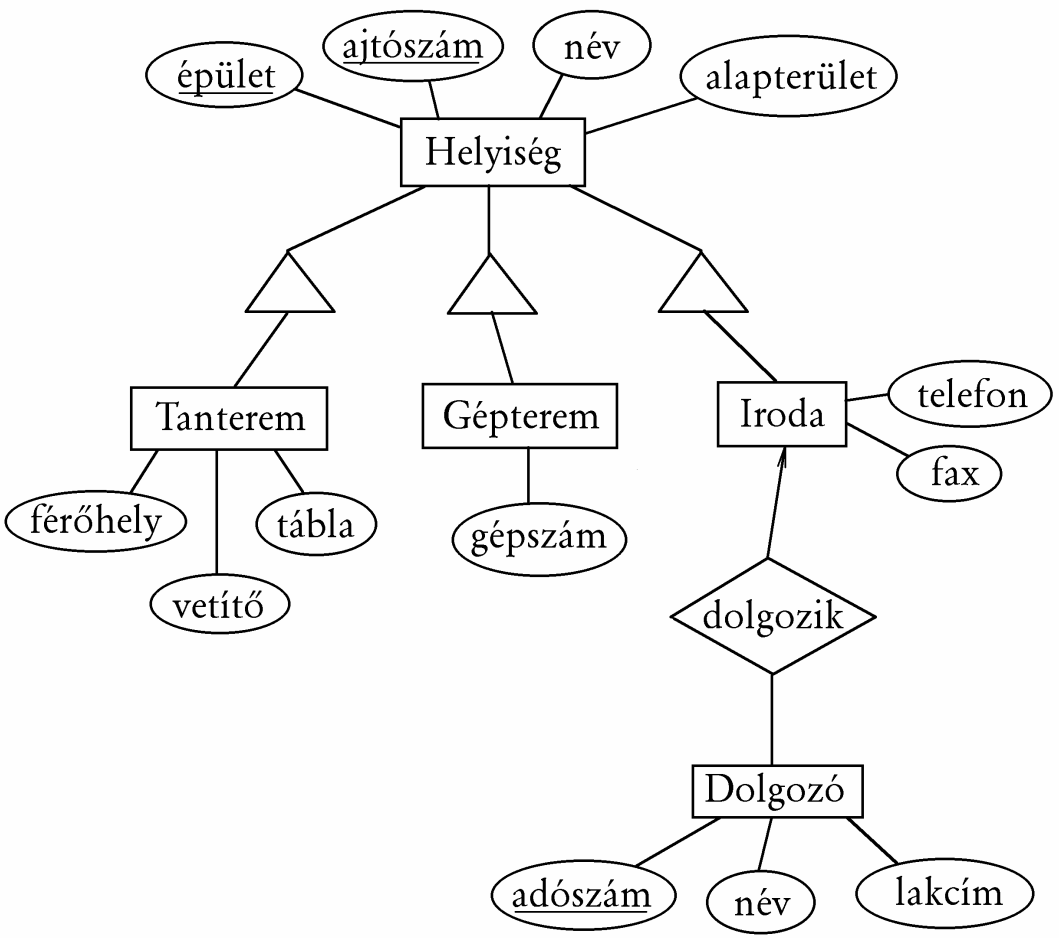


Kapcsolatok típusai

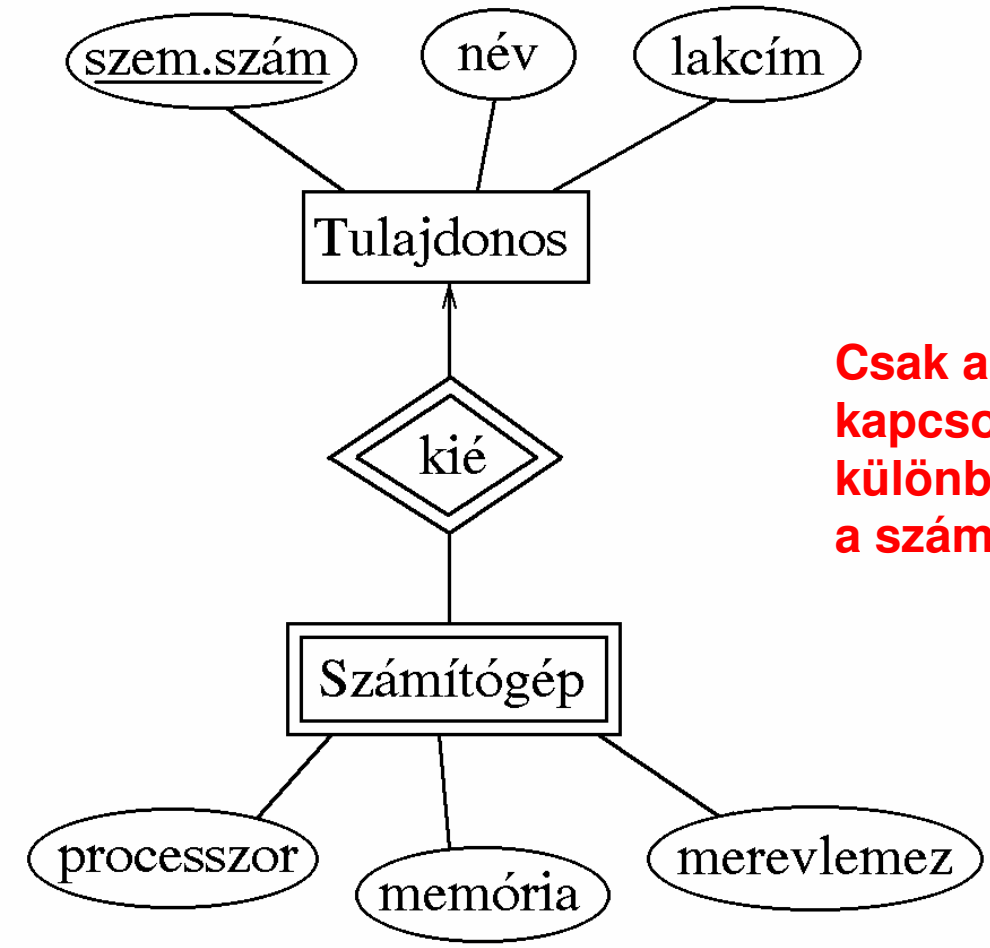
- **K(E1,E2) bináris kapcsolat,**
  - **öröklődési kapcsolat ("az egy", ISA),**
  - **"a PC is a computer" = "a PC az egy számítógép",**
  - **speciális egy-egy kapcsolat,**
  - **K {(ei,ej)} alakú előfordulásaiban az összes E1-beli egyed szerepel,**
  - **például: az_egy(főnök,dolgozó).**



"Az egy" kapcsolat

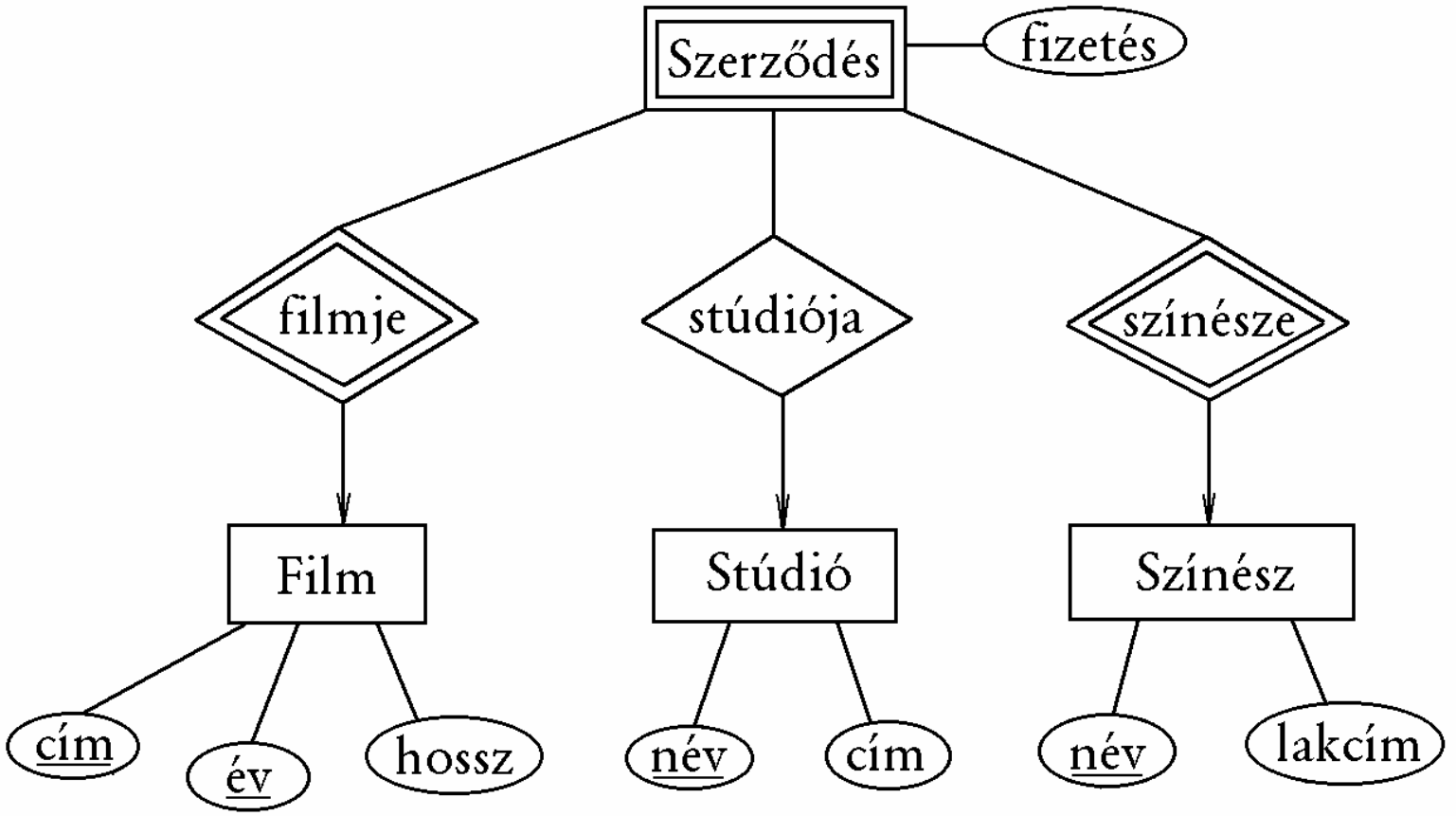


GYENGE EGYEDOSZTÁLY



Csak a "kié" kapcsolaton keresztül különböztethetők meg a számítógépek.

Sokágú kapcsolat helyettesítése gyenge egyedosztállyal és bináris kapcsolatokkal



Általánosítás

- A kapcsolatnak is lehetnek saját tulajdonságai
  - K(E1,…,Ep,A1,…,Aq), ahol A1,…,Aq tulajdonságok,
    - például: index(diák,tárgy,jegy,dátum), ahol a jegy és dátum saját tulajdonság.
- Általánosított egy-sok többágú kapcsolat
  - K(E1,…,Ep) kapcsolat előfordulásaiban nem szerepelhet egyszerre (e1,e2,…,ep) és (e1',e2,…,ep), ha e1 és e1' különböznek, vagyis minden (E2,…,Ep)-beli p-1-eshez csak 1 E1-beli egyed tartozhat,
  - hasonlóan értelmezhető bármelyik p-1 egyedosztályra,
  - például: szállít(szállító,áru,ár), ahol feltesszük, hogy egy szállító egy adott árut csak egyféle áron szállíthat,
    - az ár tekinthető saját tulajdonságnak, vagy Ár(ár) egyedosztálynak.

Szuperkulcsok, kulcsok, azonosítók

- Az egyedosztály szuperkulcsa egy azonosító, vagyis olyan tulajdonság-halmaz, amelyről feltehető, hogy az egyedosztály előfordulásaiban nem szerepel két különböző egyed, amelyek ezeken a tulajdonságokon megegyeznek.
- Az összes tulajdonság mindig szuperkulcs.
- A minimális szuperkulcsot kulcsnak nevezzük.
- Az egyedosztály (szuper)kulcsai azonosításra használhatók.
- Több (szuper)kulcs is lehet. Ezek közül egyet kiválasztunk elsődleges (szuper)kulcsnak, a többi másodlagos (szuper)kulcs.
- Ha E1 egyedosztályban szereplő T1,…,Tk tulajdonságok halmaza az E2 egyedosztály (szuper)kulcsa, akkor azt mondjuk, hogy T1,…,Tk idegen (szuper)kulcsa az E1-nek az E2-re nézve,
  - például: diák(etr_kód,név,anyja_neve,szül_idő,lakcím) egyedosztályban a név,anyja_neve,szül_idő idegen kulcs az ösztöndíjas(név,anyja_neve,szül_idő,év,ösztöndíj) egyedosztályra nézve.

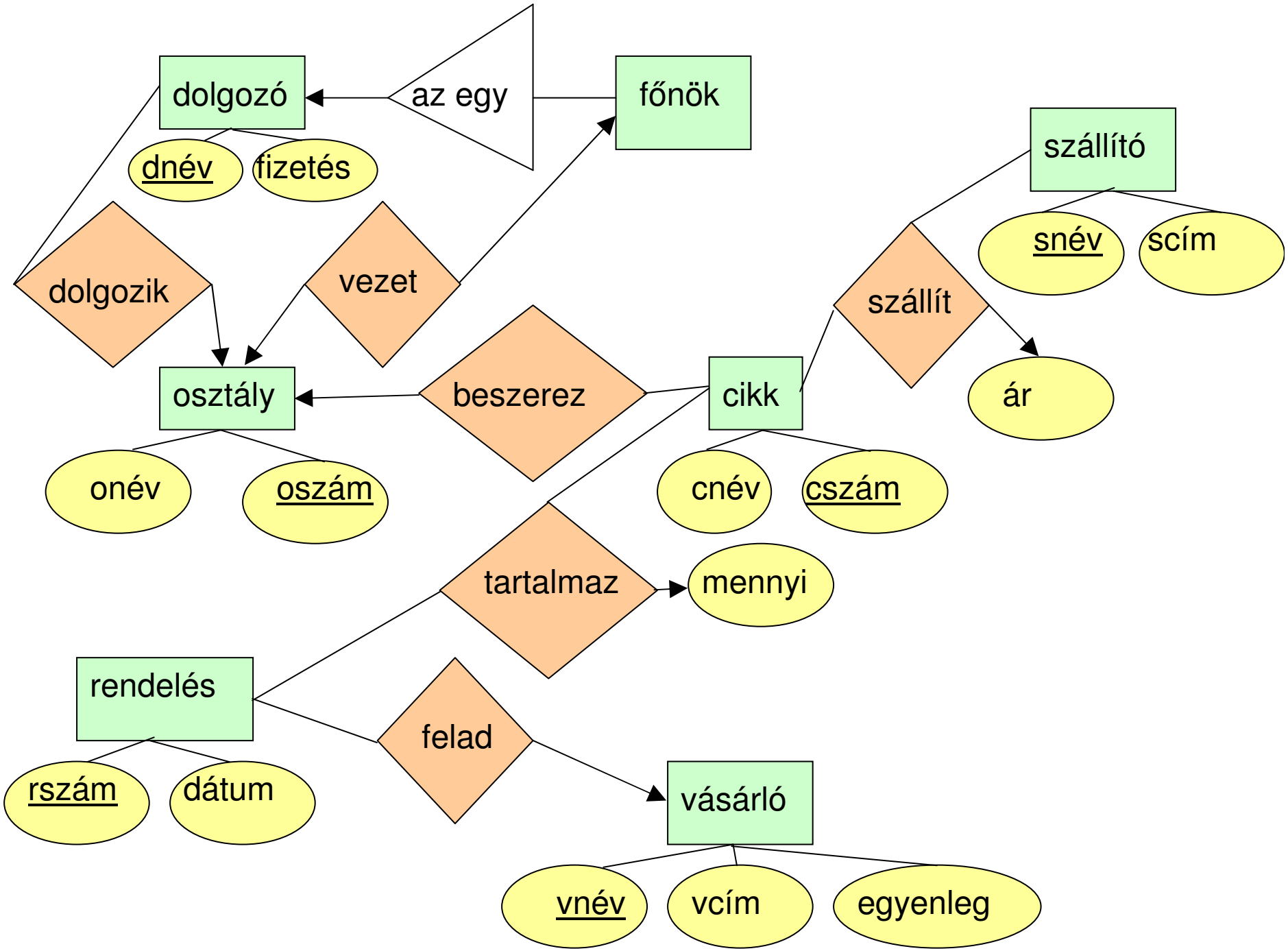
Tervezési alapelvek

- **valósághű modellezés:**
  - megfelelő tulajdonságok tartozzanak az egyedosztályokhoz, például a tanár neve ne a diák tulajdonságai közé tartozzon
- **redundancia elkerülése:**
  - az index(etr_kód,lakcím,tárgy,dátum,jegy) rossz séma, mert a lakcím annyiszor ismétlődik, ahány vizsgajegye van a diáknak, helyette 2 sémát érdemes felvenni:

    hallgató(etr_kód,lakcím), vizsga(etr-kód,tárgy,dátum,jegy).
- **egyszerűség:**
  - fölöslegesen ne vegyünk fel egyedosztályokat
  - például a naptár(év,hónap,nap) helyett a megfelelő helyen inkább dátum tulajdonságot használjunk
- **tulajdonság vagy egyedosztály:**
  - például a vizsgajegy(jegy) osztály helyett jegy tulajdonságot használjunk.

Példa: E/K diagramra

- **Modellezzük egy áruháznak, dolgozóinak, vevőinek és beszállítóinak rendszerét!**
- **Feltételezések:**
  - **az áruház minden osztályát legfeljebb egy ember vezeti,**
  - **minden dolgozó legfeljebb egy osztályon dolgozik,**
  - **az áruházak osztályai felelősek az áruk beszerzéséért,**
  - **minden szállító legfeljebb egyféle áron szállít egy árut,**
  - **egy rendelést legfeljebb egy vevőhöz tartozhat,**
  - **minden rendelésen egy cikkhez legfeljebb egy rendelt mennyiség tartozhat.**


dolgozó
az egy
főnök
dnév
fizetés
dolgozik
vezet
szállító
snév
scím
szállít
ár
osztály
beszerez
cikk
onév
oszám
cnév
cszám
tartalmaz
mennyi
rendelés
felad
rszám
dátum
vásárló
vnév
vcím
egyenleg

E/K diagram átalakítása relációs adatbázistervre

Mi minek felel meg:

- egyedosztály séma ⟺ relációséma
  E(A1,…,An) ⟺ E(A1,…,An)
- tulajdonságok ⟺ attribútumok
- (szuper)kulcs ⟺ (szuper)kulcs
- egyedosztály előfordulása ⟺ reláció
- e egyed ⟺ (e(A1),…,e(An)) sor
- R(E1,…Ep,A1,…,Aq) ⟺ R(K1,…,Kp,A1,…,Aq)
  kapcsolati séma, ahol ⟺ relációséma, ahol
  Ei egyedosztály, ⟺ Ki az Ei (szuper)kulcsa
  Aj saját tulajdonság

E/K modell ⟺ Relációs adatmodell

E/K diagram átalakítása relációs adatbázistervre

- A transzformálás előtt a tulajdonságokat átnevezhetjük, hogy a relációsémában ne szerepeljen kétszer ugyanaz az attribútum.

- Az az_egy kapcsolat esetén a speciális osztály saját attribútumaihoz hozzávesszük az általános osztály (szuper)kulcsát.

- Ha R(E1,E2) sok-egy kapcsolat, akkor R(K1,K2) relációsémának a K1 szuperkulcsa lesz.

- A gyenge entitás relációsémáját bővíteni kell a meghatározó kapcsolat(ok)ban szereplő egyed(ek) kulcsával.

Az Áruház diagram átalakítása adatbázistervvé

Az egyedosztályok átalakítása:

- dolgozó(dnév, fizetés)
- főnök(dnév)
- osztály(onév,oszám)
- szállító(snév,scím)
- cikk(cnév,cszám)
- rendelés(rszám,dátum)
- vásárló(vnév,vcím,egyenleg)

A kapcsolatok átalakítása:

- dolgozik(dnév,oszám)
- vezet(dnév,oszám)
- beszerez(cszám,oszám)
- szállít(cszám,sznév,ár)
- tartalmaz(rszám,cszám,mennyi)
- felad(rszám,vnév)

Összesen 13 relációsémát kaptunk!

Összevonások

- **Két relációsémát összevonhatunk, ha az egyikben van idegen (szuper)kulcs a másikra nézve.**

- **E1(A1,…,An,B1,…,Bm) és E2(B1,…,Bm,C1,…,Cp) helyett**

  E3(A1,…,An,B1,…,Bm,C1,…,Cp) relációsémát vehetjük, ha B1,…,Bm az E2 elsődleges, vagy másodlagos (szuper)kulcsa.

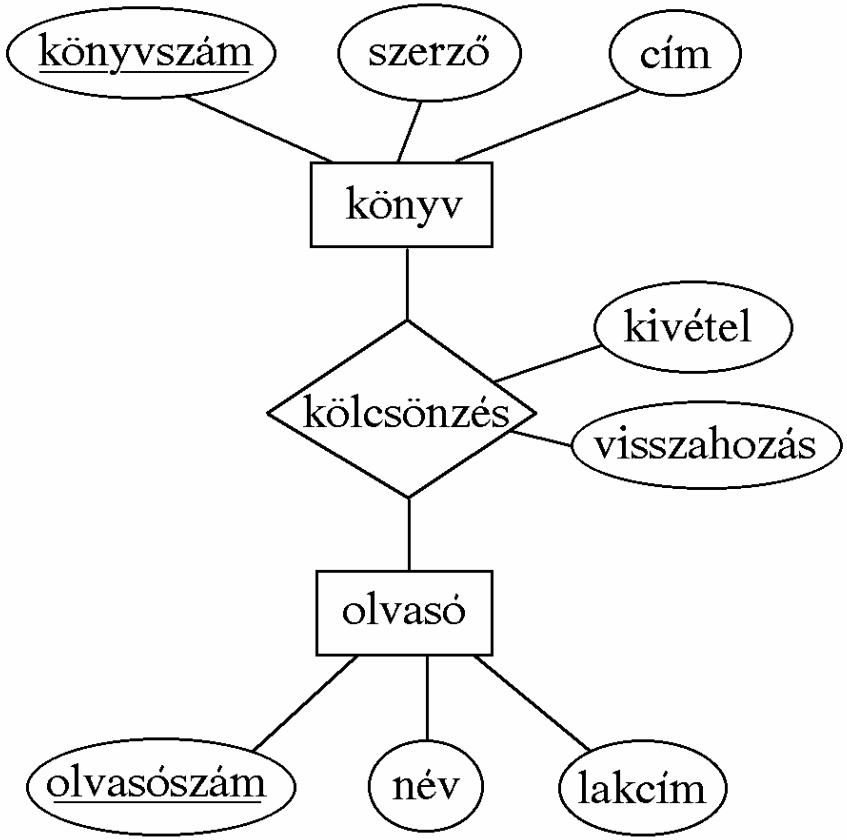
- **Az összevonás eredményét felhasználhatjuk újabb összevonásokban.**

Összevonások eredménye

- dolgozó(<u>dnév</u>,fizetés,oszám)
  dolgozó+dolgozik összevonása
- osztály(onév,<u>oszám</u>,dnév)
  vezet+osztály+főnök összevonása
- szállító(<u>snév</u>,scím)
- cikk(cnév,<u>cszám</u>,oszám)
  cikk+beszerez összevonása
- rendelés(<u>rszám</u>,dátum,vnév)
  rendelés+felad összevonása
- vásárló(<u>vnév</u>,vcím,egyenleg)
- szállít(<u>snév,cszám</u>,ár)
- tartalmaz(<u>rszám,cszám</u>,mennyi)

Összesen 8 relációsémát kaptunk!

Példa



KÖNYV (könyvszám, szerző, cím)
OLVASÓ (olvasószám, név, lakcím)
KÖLCSÖN (könyvszám, olvasószám, kivétel, visszahozás)

Összetett attribútumok leképezése

- **Tegyük fel, hogy az OLVASÓ táblában a *lakcím* attribútumot (helység, utca, házszám) struktúraként szeretnénk kezelni. Relációs adatmodellben erre egyetlen lehetőség van: az**

  OLVASÓ (olvasószám, név, lakcím)

  séma helyett a

  OLVASÓ (olvasószám, név, helység, utca, házszám)

  sémára térünk át.

Többértékű attribútumok leképezése

- **Kérdés, hogy többszerzős könyveket hogyan tartsunk nyilván az adatbázisban.**

  1. Megadás egyértékű attribútumként. **A szerző megadására szolgáló szövegmezőben felsoroljuk a szerzőket.**

- **Hátrányok:**

  – a szerzőket külön-külön nem tudjuk kezelni.

  – sok szerző esetleg nem fér el a megadott mezőben

Többértékű attribútumok leképezése

- *2. Megadás többértékű attribútumként.*
  a) Sorok többszörözése. **A KÖNYV táblában egy könyvhöz annyi sort veszünk fel, ahány szerzője van:**

Könyvszám	*Szerző*	*Cím*
1121	Ullman	Adatbázisok
1121	Widom	Adatbázisok
3655	Radó	Világatlasz
2276	Karinthy	Így írtok ti
1782	Jókai	Aranyember

A megfelelő relációséma:

KÖNYV (könyvszám, szerző, cím)

- **A fenti megoldás hátránya, hogy a többszerzős könyvek címét több példányban kell megadni, ami redundanciát jelent.**

Többértékű attribútumok leképezése

2. Megadás többértékű attribútumként.

b) új tábla felvétele.

A KÖNYV (könyvszám, szerző, cím) sémát az alábbi két sémával helyettesítjük:

KÖNYV (könyvszám, cím)

SZERZŐ (*könyvszám*, szerző)

Többértékű attribútumok leképezése

2. Megadás többértékű attribútumként.

c) Sorszámozás. **Ha a szerzők sorrendje nem közömbös, akkor a SZERZŐ táblát egy sorszám mezővel kell bővíteni (emlékeztetünk rá, hogy a relációs adatmodell nem definiálja a rekordok sorrendjét):**

KÖNYV (könyvszám, cím)

SZERZŐ (könyvszám, sorszám, szerző)

Kapcsolatok leképezése

1. változat: **Ha egy olvasónak egyszerre csak egy könyvet adnak ki, akkor a kölcsönzés 1:1 kapcsolatot jelent. Ilyenkor a KÖLCSÖN sémában a *könyvszám* és az *olvasószám* egyaránt kulcs. Továbbá, a *visszahozás* attribútumra nincs szükségünk, mivel a könyv visszahozásával a könyv-olvasó kapcsolat megszűnik.**

Tehát, a
KÖLCSÖN (könyvszám, olvasószám, kivétel)
vagy a
KÖLCSÖN (könyvszám, olvasószám, kivétel)
sémát vehetjük fel a kapcsolathoz.

A KÖLCSÖN sémát az azonos kulcsú sémába olvasztva a

KÖNYV (könyvszám, szerző, cím, *olvasószám*, kivétel)
OLVASÓ (olvasószám, név, lakcím)

vagy a

KÖNYV (könyvszám, szerző, cím)
OLVASÓ (olvasószám, név, lakcím, *könyvszám*, kivétel)

adatbázissémákat kapjuk.

Kapcsolatok leképezése

2. változat: **Ha egy olvasó több könyvet is kikölcsönözhet, akkor a könyv-olvasó kapcsolat N:1 típusú. Ekkor a KÖLCSÖN sémában csak a *könyvszám* lehet kulcs, ezért a KÖLCSÖN sémát csak a KÖNYV sémába olvaszthatjuk:**

KÖNYV (könyvszám, szerző, cím, *olvasószám*, kivétel)

OLVASÓ (olvasószám, név, lakcím)

Kapcsolatok leképezése

3. változat: **Ha az egyes könyvek korábbi kölcsönzéseit is nyilvántartjuk, akkor nem csak egy olvasóhoz tartozhat több könyv, hanem egy könyvhöz is több olvasó (N:M kapcsolat), sőt adott olvasó adott könyvet egymás után többször is kikölcsönözhet. Ezért a KÖLCSÖN sémában**

{könyvszám, kivétel}

vagy

{könyvszám, visszahozás}

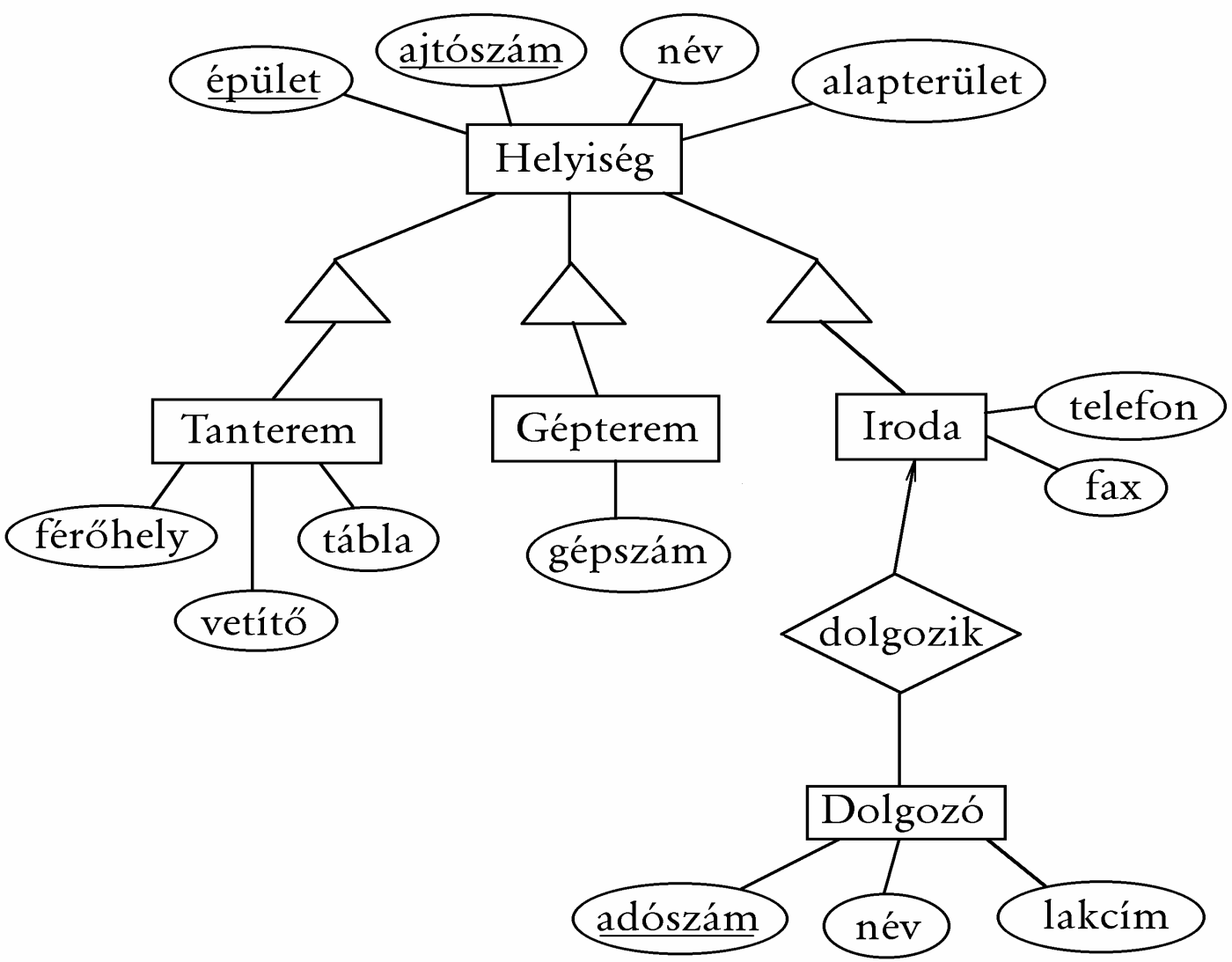
a kulcs, a KÖLCSÖN táblát most sem a KÖNYV, sem az OLVASÓ táblába nem tudjuk beolvasztani. Az adatbázisséma ezért a következő:

KÖNYV (könyvszám, szerző, cím)

OLVASÓ (olvasószám, név, lakcím)

KÖLCSÖN (*könyvszám*, *olvasószám*, kivétel, visszahozás)

Specializáló kapcsolatok leképezése



Specializáló kapcsolatok leképezése

1. Minden altípushoz külön tábla felvétele, egy egyed csak egy táblában szerepel. Az altípusok öröklik a főtípus attribútumait.
(Objektumorientált stílusú reprezentálás)

HELYISÉG (épület, ajtószám, név, alapterület)
TANTEREM (épület, ajtószám, név, alapterület, férőhely, tábla, vetítő)
GÉPTEREM (épület, ajtószám, név, alapterület, gépszám)
IRODA (épület, ajtószám, név, alapterület, telefon, fax)
DOLGOZÓ (adószám, név, lakcím, *épület, ajtószám*)

Hátrányok:
– Kereséskor gyakran több táblát kell vizsgálni (ha például a D épület 803. sz. terem alapterületét keressük).
– Kombinált altípus (például számítógépes tanterem) csak új altípus felvételével kezelhető.

Specializáló kapcsolatok leképezése

2. Minden altípushoz külön tábla felvétele, egy egyed több táblában is szerepelhet. A főtípus táblájában minden egyed szerepel, és annyi altípusában ahánynak megfelel. Az altípusok a főtípustól csak a kulcs-attribútumokat öröklik.
(E/K stílusú reprezentálás.)

HELYISÉG (épület, ajtószám, név, alapterület)
TANTEREM (*épület, ajtószám*, férőhely, tábla, vetítő)
GÉPTEREM (*épület, ajtószám*, gépszám)
IRODA (*épület, ajtószám*, telefon, fax)
DOLGOZÓ (adószám, név, lakcím, *épület, ajtószám*)

Hátrány: Itt is előfordulhat, hogy több táblában kell keresni (például ha a tantermek nevére és férőhelyére vagyunk kíváncsiak).

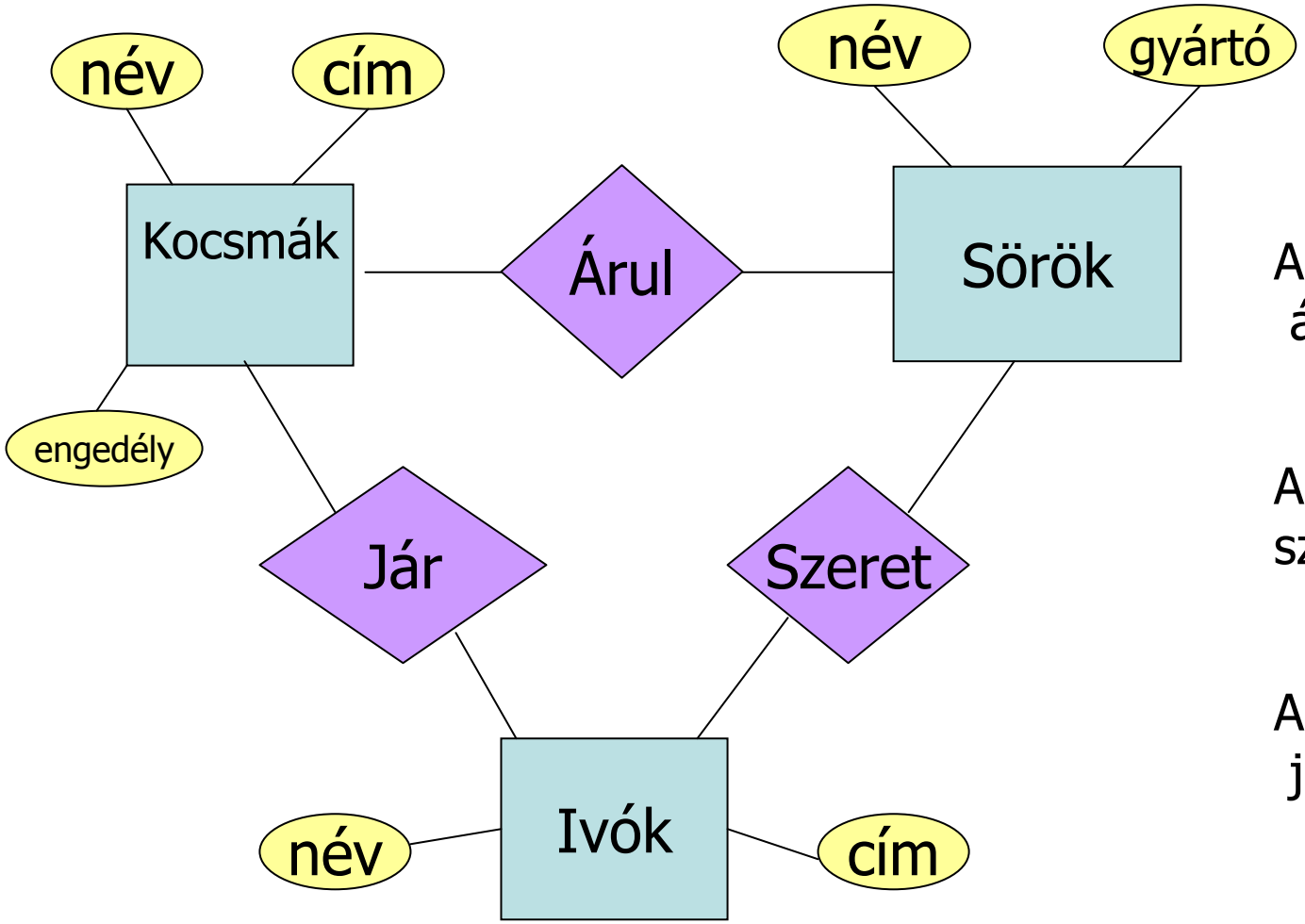
Specializáló kapcsolatok leképezése

3. Egy közös tábla felvétele, az attribútumok únió jával. Az aktuálisan értékkel nem rendelkező attribútumok NULL értékűek.
(Reprezentálás nullértékekkel)

HELYISÉG (épület, ajtószám, név, alapterület, férőhely, tábla, vetítő, gépszám, telefon, fax)
DOLGOZÓ (adószám, név, lakcím, *épület, ajtószám*)

Hátrányok:
– Az ilyen egyesített táblában általában sok NULL attribútumérték szerepel.
– Elveszíthetjük a típusinformációt (például ha a gépteremnél a gépszám nem ismert és ezért NULL, akkor a gépterem lényegében az egyéb helyiségek kategóriájába kerül).

Példa: Kapcsolatok

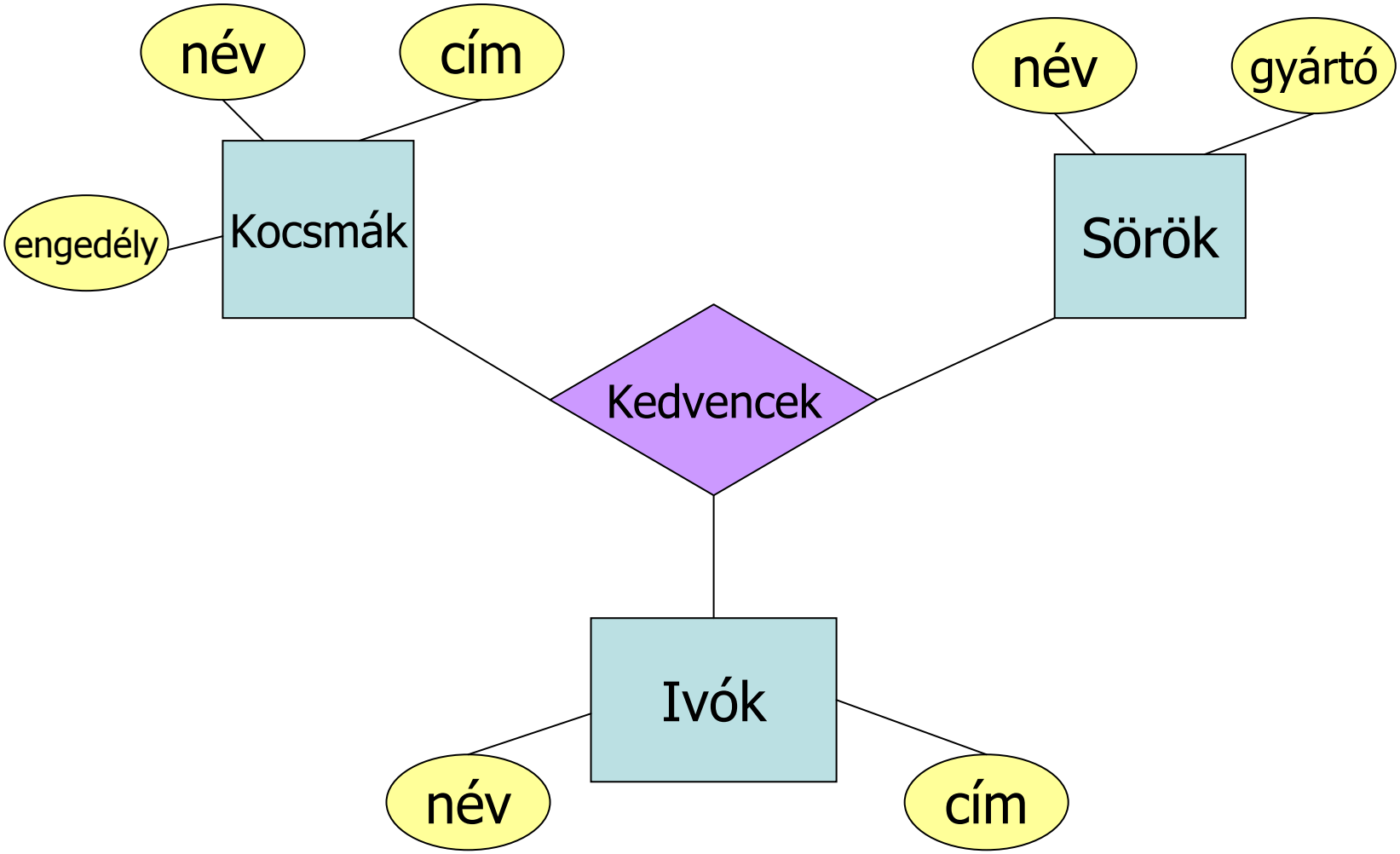


A kocsmák söröket árulnak.

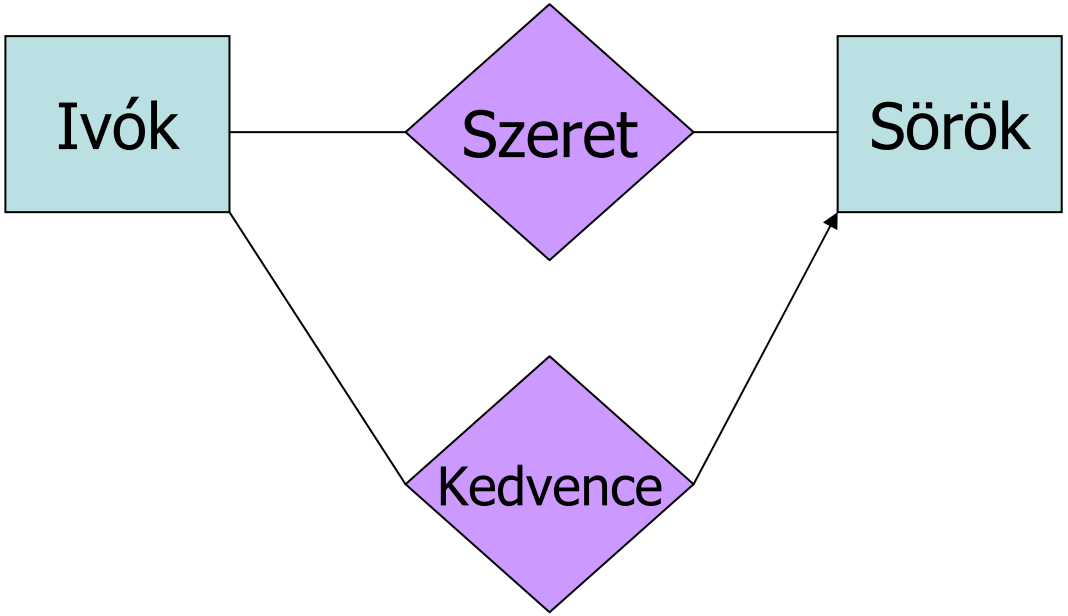
Az ivók söröket szeretnek.

Az ivók kocsmákba járnak.

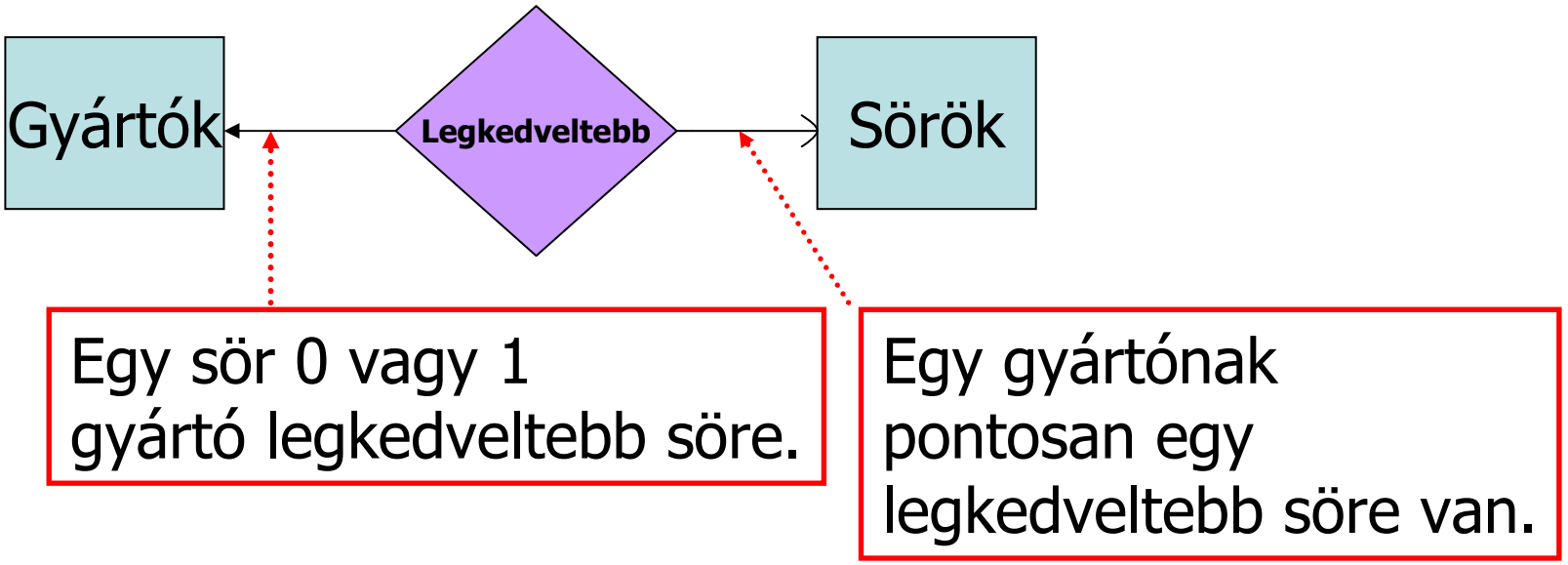
Példa: 3-ágú kapcsolat



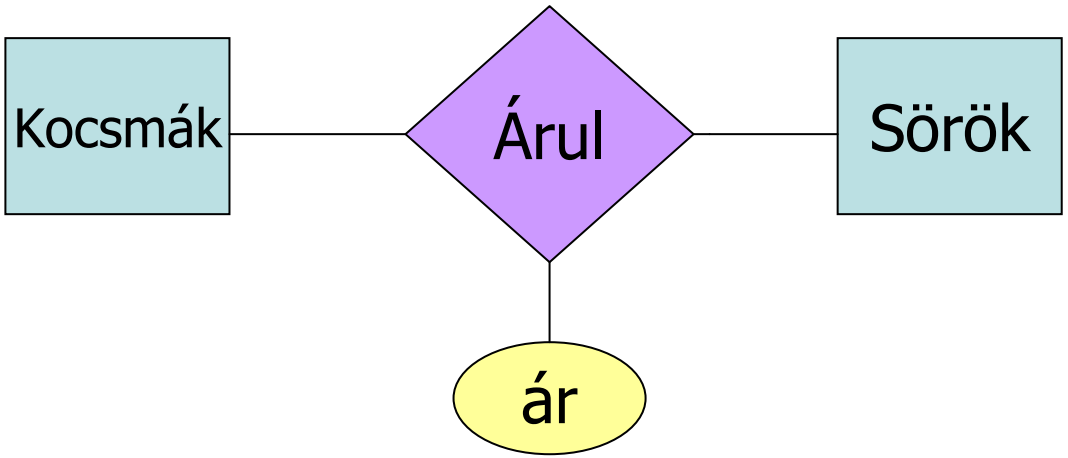
Példa: Több kapcsolat is lehet



E/R diagram

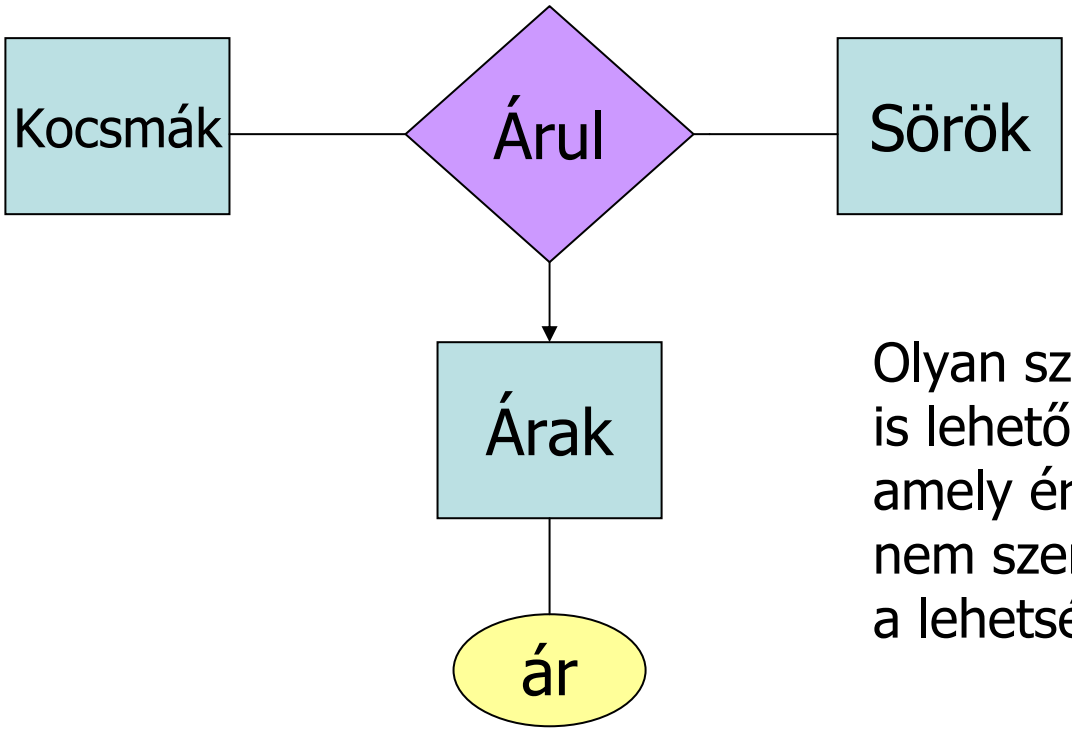

Gyártók
Legkedveltebb
Sörök
Egy sör 0 vagy 1
gyártó legkedveltebb söre.
Egy gyártónak
pontosan egy
legkedveltebb söre van.

Példa: Kapcsolat attribútuma



Az ár a kocsma és sör együttes függvénye, de egyiké sem külön.

Attribútum vagy egyedosztály?

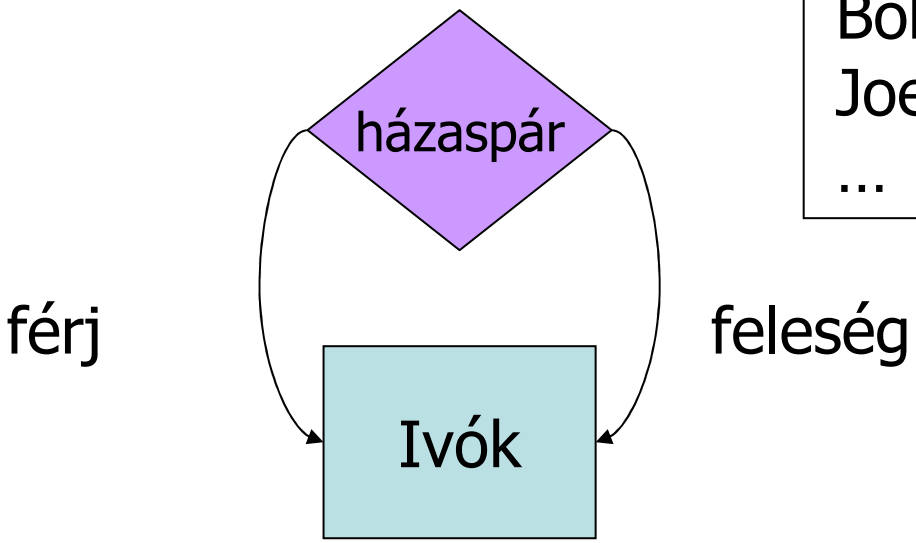


Olyan számokat (árakat)
is lehetőségünk van tárolni,
amely értékek még árként
nem szerepelnek, de csak ezek
a lehetséges árértékek.

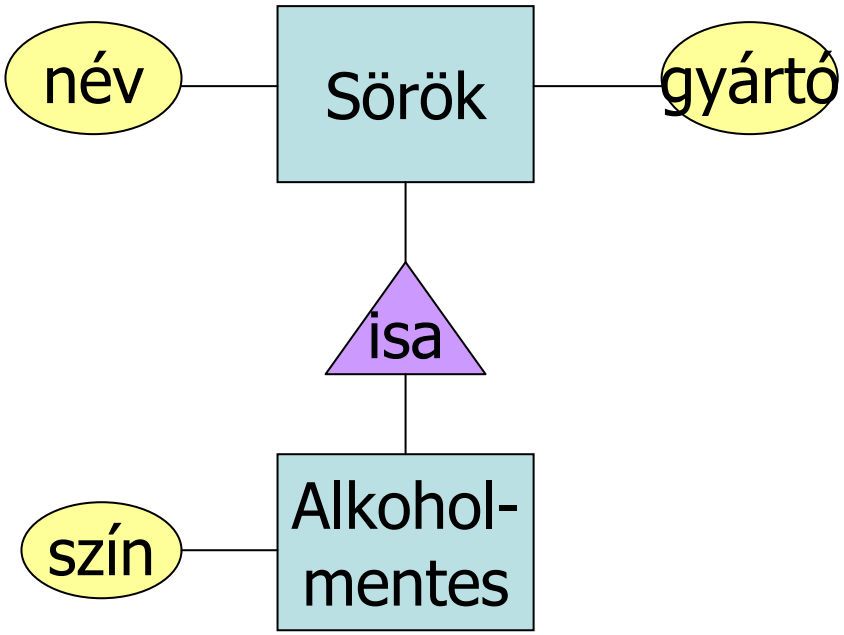
Példa: Szerepek (Roles)

A kapcsolat előfordulása

Férj	Feleség
Bob	Ann
Joe	Sue
...	...

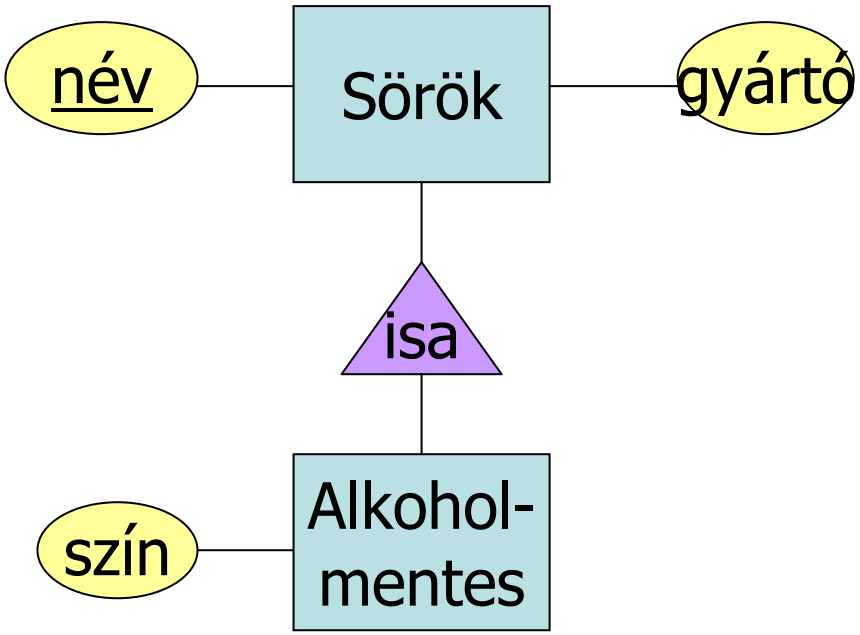


Példa: Öröklődés (Subclasses)

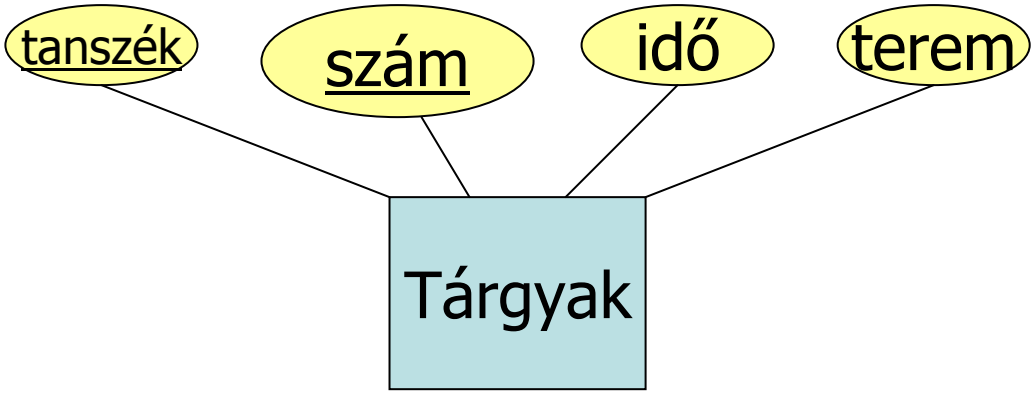


Példa: Kulcsok

A név kulcsa a Sörök-nek

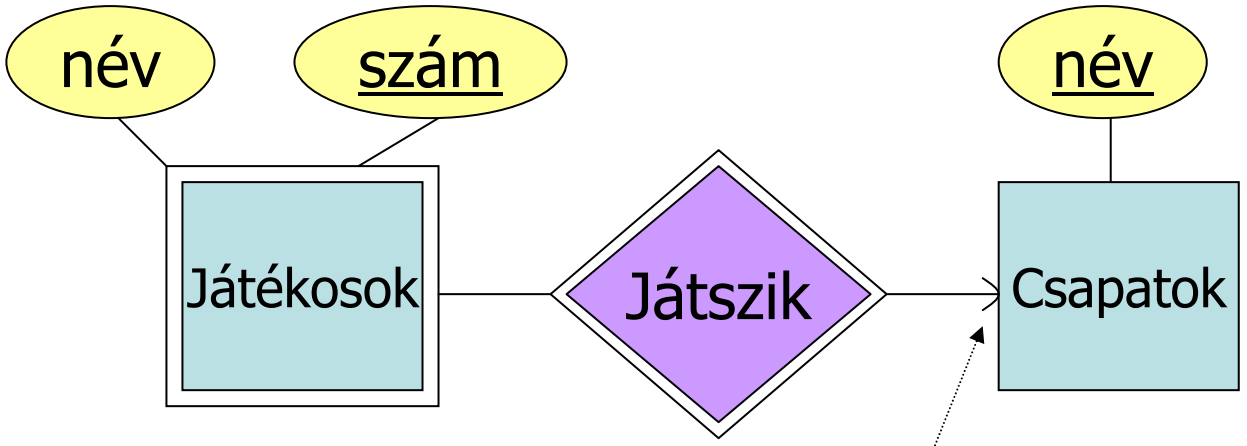


Példa: összetett kulcsok



- **Az elsődlegesen kívül lehet más kulcs is:**
- **(idő, terem) is összetett kulcs.**

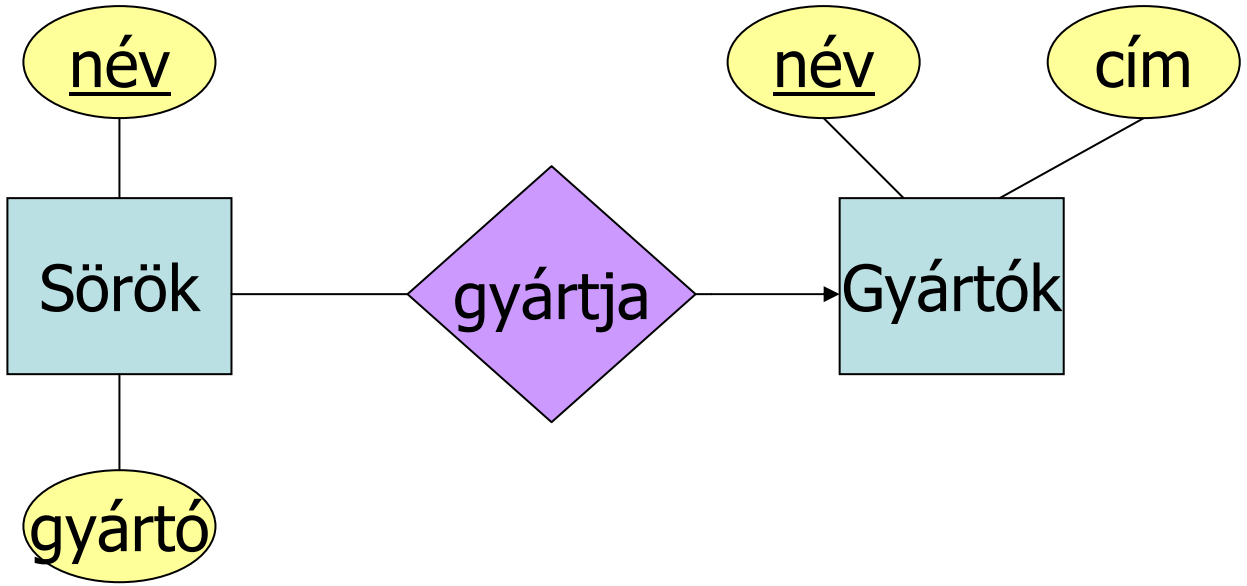
Gyenge egyedosztályok



A kerek végződés jelzi, hogy minden játékoshoz kötelezően tartozik egy csapat, amely az azonosításhoz használható.

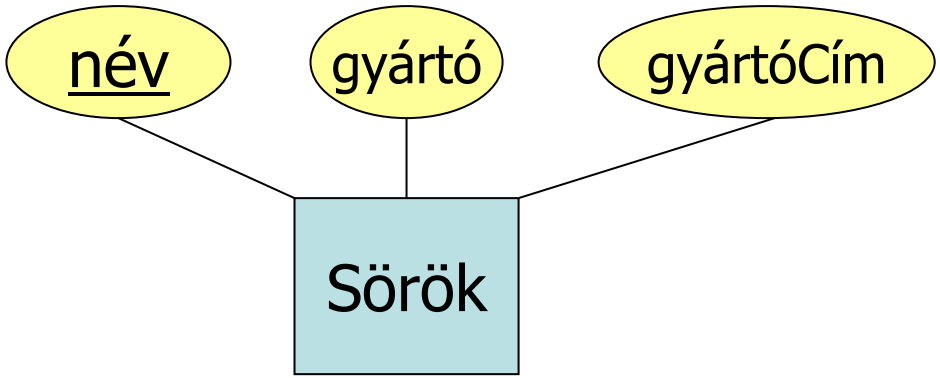
- **Dupla rombusz: sok-egy kapcsolat.**
- **Dupla téglalap: gyenge egyedhalmaz.**

Példa: Rossz modell



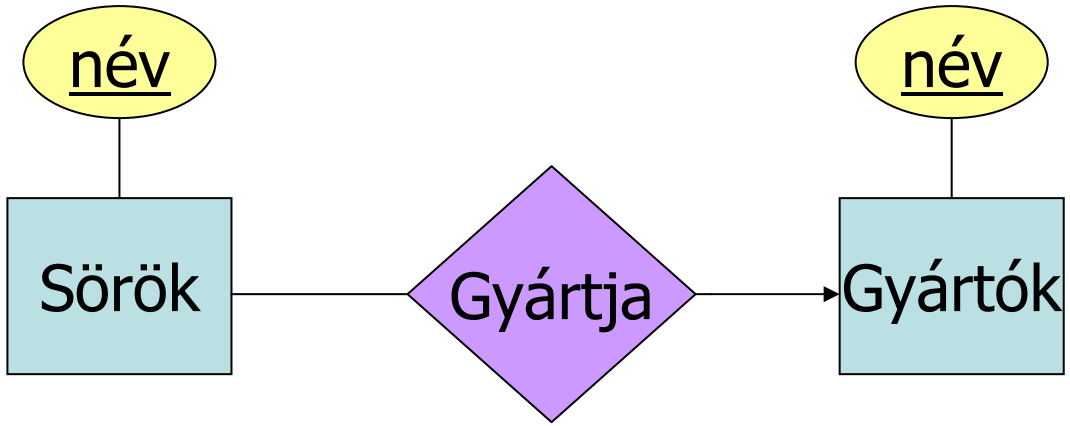
A gyártó egyszerre tulajdonság is meg egyed is!

Példa: Rossz modell



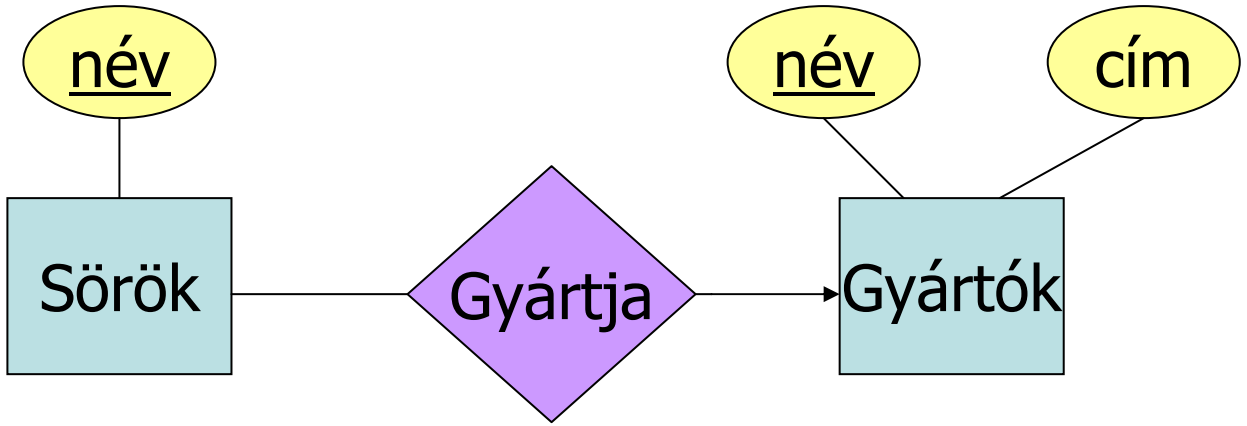
- **Minden sörre megismételjük a gyártó címét is.**
- **Ha egy gyártó éppen most nem gyárt sört, akkor a címét is elveszítjük.**

Példa: Rossz modell



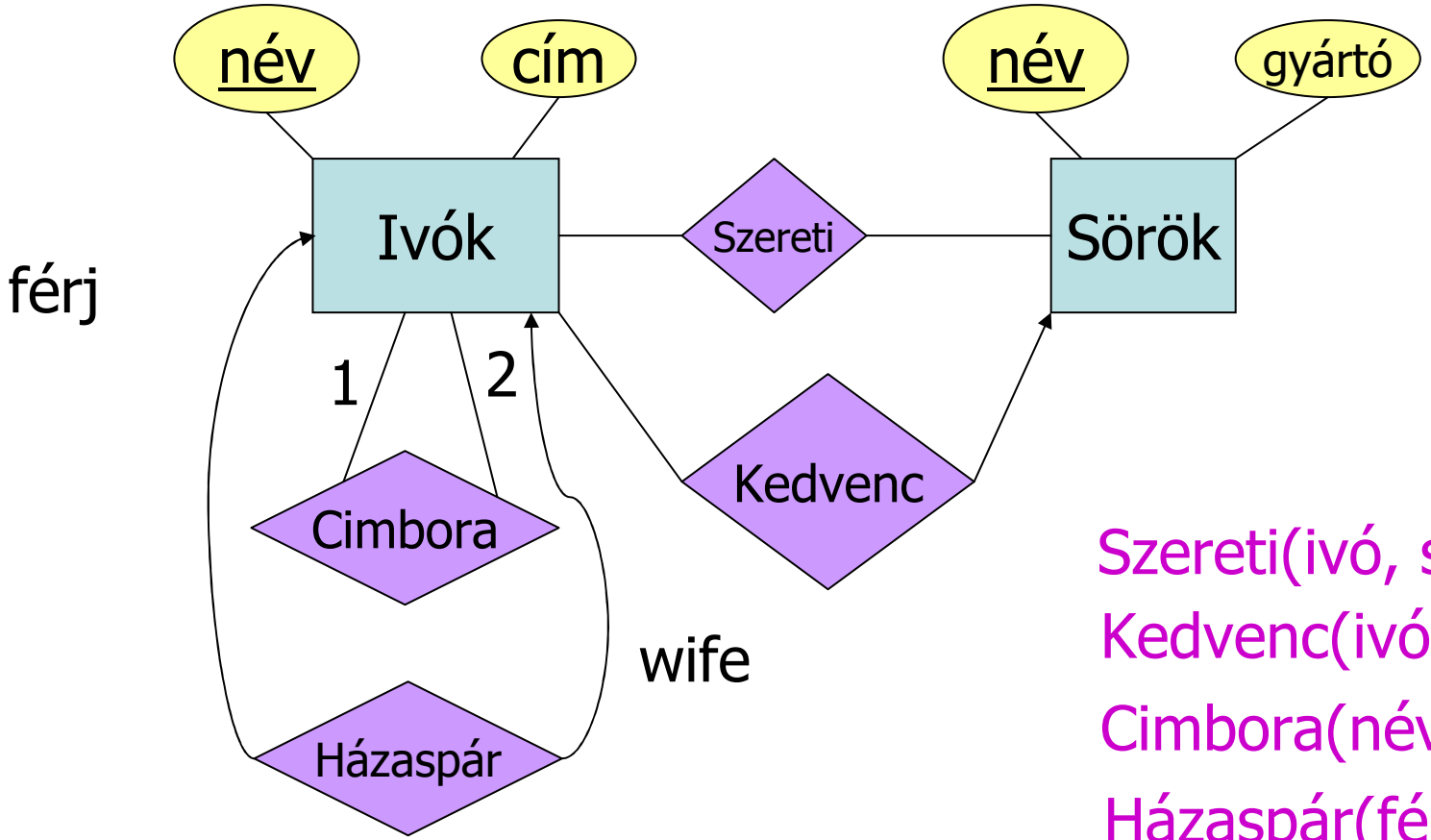
A gyártók egyedosztálynak csak 1 attribútuma van. Az ilyet nem érdemes egyedosztálynak tekinteni, hanem egy másik egyedosztály vagy kapcsolat tulajdonságának.

Példa: Jó modell



Minden gyártó címe csak egyszer fog szerepelni.

Kapcsolat -> Reláció



Szereti(ivó, sör)

Kedvenc(ivó, sör)

Cimbora(név1, név2)

Házaspár(férj, feleség)

A kapcsolatoknak megfelelő sémákban az oszlopokat célszerű átnevezni, például a szerepek alapján. Egyébként is (név,név) séma nem szerepelhetne. Az új nevek: Ivók.név helyett ivó, stb.

Relációk összevonása

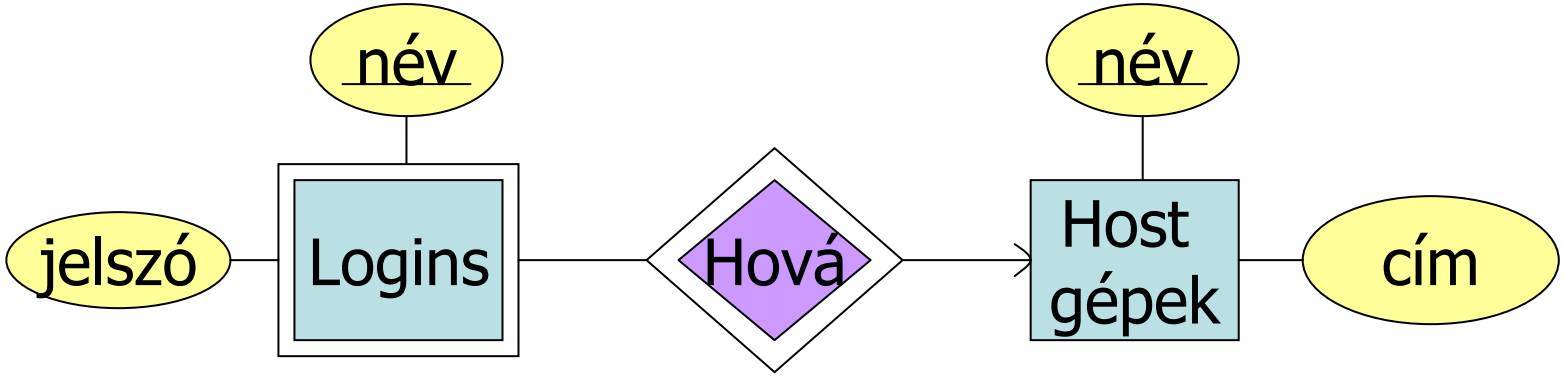
- **Összevonhatunk 2 relációt, ha**

  az egyik egy sok-egy kapcsolatnak megfelelő reláció, a másik pedig a sok oldalon álló egyedhalmaznak megfelelő reláció.

- **Példa:**

  Ivók(név, cím) és Kedvenc(ivó,sör) összevonható, és kapjuk az Ivó1(név,cím,kedvencSöre) sémát.

Példa: Gyenge egyedhalmaz -> Reláció

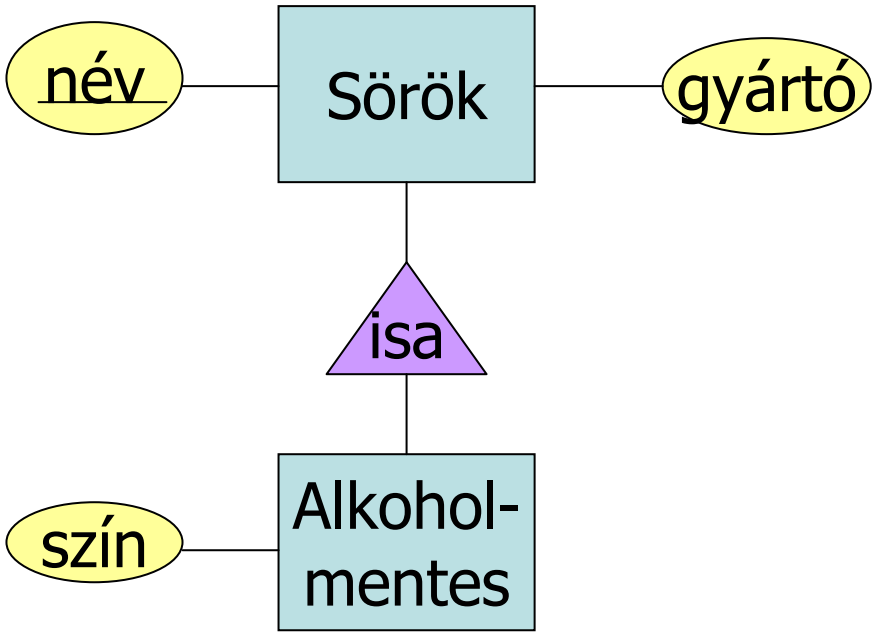


Hostgépek(hostNév, cím)
Logins(loginNév, hostNév, jelszó)
~~Hová(loginNév, hostNév, hostNév2)~~

Beolvasztjuk a Logins relációba

A logins kulcsa összetett:
loginNév,hostNév
Kétszer szerepelne az azonos értékű hostNév a Hová sémában

Példa: Alosztály -> Reláció



Alosztályok átírása: három megközelítés

1. *Objektumorientált stílusban*: Egy reláció minden alosztályra, felsorolva az összes tulajdonságot, beleértve az örökölteket is.

2. *Nullértékek használatával*: Egyetlen reláció az öröklődésben résztvevő összes osztályra. Ha egy egyed nem rendelkezik egy alosztály speciális tulajdonságával, akkor ezt az attribútumot NULL értékkel töltjük majd ki.

3. *E/R stílusban*: Egy reláció minden alosztályra, de az általános osztályból csak a kulcsokat vesszük hozzá a saját attribútumokhoz.

Objektumorientált stílusú reprezentálás

név	gyártó
Bud	Anheuser-Busch

Sörök

név	gyártó	szín
Summerbrew	Pete's	világos

Alkoholmentes

Az olyan lekérdezésekre jó, hogy egy adott gyártó milyen színű alkoholmentes söröket gyárt.

Nullértékek használatával

név	gyártó	szín
Bud	Anheuser-Busch	NULL
Summerbrew	Pete's	világos

Sörök

Általában kevesebb hely elég a tárolásra, kivéve ha nagyon sok attribútum marad nullértékű.

E/R stílusú

név	gyártó
Bud	Anheuser-Busch
Summerbrew	Pete's

Sörök

név	szín
Summerbrew	világos

Alkoholmentes

Az olyan lekérdezésekre jó, hogy egy adott gyártó milyen söröket gyárt, beleértve az alkoholmenteseket is.

Adatmodellező nyelvek

Object Description Language - ODL

Unified Modeling Language - UML

Objektumorientált adatbázisok

- **A szabványosítást végző csoport:**
  ODMG = Object Data Management Group.

- **ODL = Object Description Language, objektumleíró nyelv (olyan utasításokkal, mint a CREATE TABLE utasítás az SQL-ben).**

- **OQL = Object Query Language, az SQL kiterjesztése objektumorientált adatbázisokra.**

ODL

- **ODMG ajánlása szerint az objektumorientált adatbázis-kezelők gyártói egy C++ vagy hasonló objektumorientált nyelvet kiegészítenek OQL objektumorientált lekérdezőnyelvvel, így egyszerűvé válik az objektumorientált adatbázisban tárolt adatok kezelése.**

- **Az ODL állandó (*persistent*) objektumosztályokat definiál, azaz az objektumok egy adatbázisban tárolódnak.**
  - **Az ODL osztályok olyanok, mint az E/K modellben az egyedosztályok, amelyekhez bináris kapcsolatok adhatók meg, és metódusokkal is rendelkezhetnek.**
  - **Az ODL osztálydefiníciói a bővített objektumorientált befogadónyelvnek a részei.**

Osztályok definiálása ODL-ben

- **Az osztály definiálása a következőkből áll:**
  1. **Az osztály neve.**
  2. **Opcionális általános osztályok, ahonnét örökli az osztályelemeket.**
  3. **Opcionális kulcsok definiálása.**
  4. **Elemek definiálása. Az *elem* vagy egy attribútum (attribute), vagy egy bináris kapcsolat (relationship), vagy egy metódus (method).**

```
class <név>
{
<elemek listája, pontosvesszővel elválasztva>
}
```

Példa: Attribútumok és kapcsolatok megadása

```
class Kocsma {
    attribute string név;
    attribute string cím;
    relationship Set<Sör> árul inverse Sör::holÁrulják;
}

class Sör {
    attribute string név;
    attribute string gyártó;
    relationship Set<Kocsma> holÁrulják inverse
    Bar::árul;
}
```

Az árul kapcsolat típusa:
Sör objektumok halmaza

A :: köti össze az osztályt
a hozzátartozó kapcsolattal.

Attribútumok és kapcsolatok megadása

- **Attribútumok megadása:**

  attribute <típus> <név>;

- **Bináris kapcsolatokkal lehet az egyik osztály egy objektumához a másik osztály egy vagy több objektumát hozzárendelni.**

- **A kapcsolat definiálásánál meg kell adni, hogy ugyanez a kapcsolat a másik osztályban milyen néven szerepel. Ez az inverz kapcsolat:**

  relationship <típus> <név> inverse <kapcsolat>;

Az inverz kapcsolat

- Tegyük fel, hogy a *C* osztályban van egy *R* kapcsolat a *D* osztályhoz.
- Ekkor a *D* osztályban kell lennie egy inverz *S* kapcsolatnak a *C* osztályhoz.
- *Az R és S relációs értelemben inverzei egymásnak*, azaz
  - ha *d* objektum egy *c* objektummal *R* kapcsolatban van, akkor és csak akkor *c* objektumnak *d* objektummal *S* kapcsolatban kell lennie.

A kapcsolatok típusai

Egy R kapcsolat típusa a következők valamelyike lehet:

1. **Osztály**

   Például: ha a kapcsolat típusa Kocsma, akkor abban az osztályban, ahol R szerepel, egy objektum egyetlen Kocsma objektummal lehet R kapcsolatban. (1:1)

2. **Halmaz (Set), Multihalmaz (Bag), Lista (List), Tömb (Array)**

   Például: ha a kapcsolat típusa Set<Kocsma>, akkor abban az osztályban, ahol R szerepel, egy objektumhoz Kocsma objektumoknak egy halmaza tartozik az R kapcsolaton keresztül. (1:n)

A kapcsolatok multiplicitása

- **Az ODL-ben minden kapcsolat bináris.**

- **Sok-sok (n:m) típusú kapcsolat:**

  A kapcsolatnak és inverzének típusa: halmaztípus (Set<…>).

- **Sok-egy (n:1) típusú kapcsolat:**

  A "sok"-nak megfelelő osztályban a kapcsolat típusa: osztálytípus.

  Az "egy"-nek megfelelő osztályban az inverz kapcsolat típusa: halmaztípus (Set<…>).

- **Egy-egy (1:1) típusú kapcsolat:**

  A kapcsolatnak és inverzének típusa: osztálytípus.

Példa: A kapcsolatok multiplicitása

```
class Ivó { …
    relationship Set<Sör> szereti inverse Sör::kedvelői;
    relationship Sör kedvencSör inverse Sör::legjobbanSzeretik;
    }
class Sör { …
    relationship Set<Ivó> kedvelői inverse Ivó::szereti;
    relationship Set<Ivó> legjobbanSzeretik inverse Ivó::kedvencSör;
    }
```

Sok-sok: mindkét végén Set<…> típus áll.

Sok-egy: csak az "egy" végén áll Set<…> típus.

Példa: A kapcsolatok multiplicitása

```
class Ivó {
    attribute … ;
    relationship Ivó feleség inverse férj;
    relationship Ivó férj inverse feleség;
    relationship Set<Ivó> ivócimborái inverse ivócimborái;
}
```

A férj és feleség egy-egy típusú és egymás inverzei.

Az ivócimborái sok-sok típusú és a SAJÁT INVERZE. Ilyenkor nem kell :: jelölés.

Többágú kapcsolatok átalakítása

- **Az ODL nem támogatja a többágú kapcsolatokat.**
- **Többágú kapcsolat helyett felvehetünk egy kapcsoló osztályt. Ennek az objektumai feleljenek meg azoknak az objektum n-eseknek, amelyek az n-ágú kapcsolatban összetartoznak.**

Kapcsoló osztályok

- **Tegyük fel, hogy egy R 3-ágú kapcsolatot akarunk reprezentálni az *X*, *Y és Z* osztályok között.**

- **Vezessünk be egy *C* osztályt, amely objektumai (*x*, *y*, *z*) objektumhármasoknak felelnek meg, ahol x,y,z rendre az *X*, *Y és Z* osztálynak objektuma, úgy hogy x,y és z R kapcsolatban állnak.**

- **Létesítsünk 3 sok-egy kapcsolatot C és X,Y illetve Z között. Az (*x*, *y*, *z*) objektumhoz tartozzon *x*, *y és z* ezekben a kapcsolatokban.**

Példa: Kapcsoló osztály

- Tegyük fel, hogy a Kocsma és Sör osztályok mellett szeretnénk reprezentálni, hogy melyik kocsma milyen áron árulja a söröket.
  - Szemben az E/R modellel, most a Kocsma és Sör osztályok közti sok-sok kapcsolatnak nem lehet saját attribútuma.

- Megoldás:

  Készítsünk 2 osztályt:

  egy Ár osztályt és a Kocsma, Ár és Sör osztályokhoz egy KÁS kapcsoló osztályt, amelynek objektumai az összetartozó kocsma,sör és ár objektumhármasoknak felelnek meg.

Példa: Kapcsoló osztály

- **Mivel az Ár osztály objektumai közönséges számok, jobb megoldást kapunk a következő módon:**
  1. **A KÁS osztályban adjuk meg az ár attribútumot. Így az Ár osztályra nincs szükség.**
  2. **Vegyünk fel 2 sok-egy kapcsolatot, egyet a KÁS osztály és a Kocsma, illetve egyet a KÁS és Sör között.**

Példa: Kapcsoló osztály

- Így a következőt kapjuk a KÁS osztályra:

```
class KÁS {
   attribute int ár;
   relationship Kocsma kocsmaKomponens
   inverse Kocsma::kásKapcsolat;
   relationship Sör sörKomponens inverse
   Sör::kásKapcsolat;
   }
```

- A Kocsma és Sör osztályokba fel kell venni a kásKapcsolat kapcsolatokat, és mindkét kapcsolat Set<KÁS> típusú legyen.

A rekordtípus (Struct) és a felsorolástípus (Enum)

- Az attribútumok típusa lehet rekordtípus (struktúra) és felsorolástípus is.
- Ezek megadása:

attribute Struct <a rekordtípus neve>
{ <mezőnév mezőtípus párok listája> }
<attribútumnév>;

attribute Enum <a felsorolástípus neve>
{ <a lehetséges konstans elemek felsorolása> }
<attribútumnév>;

Példa: A rekordtípus (Struct) és a felsorolástípus (Enum)

```
class Kocsma {
   attribute string név;
   attribute Struct Címrekord
        {string város, string utca, int házszám} cím;
   attribute Enum Minősítés
        { I.OSZT, II.OSZT, III.OSZT} színvonal;
   relationship …
}
```

Metódusok definiálása

- Az osztály definíciója tartalmazhat metódusdeklarációkat is.
- Csak az osztály metódusaival lehet elérni, kezelni az osztály objektumait.
- A metódusok szignatúráját kell csak megadni, a metódust kódoló algoritmus nem része az osztálydefiníciónak.

- A metódus szignatúrája a következőkből áll:

1. Ha van visszaadott érték, akkor a visszaadott érték típusa.
2. A metódus neve.
3. A paraméterek használati módja és típusa (a paraméter neve nélkül).
   - A használati mód lehet in, out vagy inout.
4. A kivételes esetek elnevezése.

Példa: Metódusok

```
real átlag(in string)raises(félévetHalasztott);
```

1. A metódus egy valós számot fog visszaadni, feltehetően egy megadott hallgató neve alapján a féléves átlagát.

2. Egy argumentumot adtunk meg, ez csak bemenő adat (in használati módú), azaz nem módosítja a metódus az értékét.

   Feltehetően ez a paraméter lesz a hallgató neve.

3. Az átlag nevű metódus kivételes esete, mikor nincs átlag, mert a hallgató félévet halasztott. Ennek a kivételnek adtuk meg a nevét.

Az ODL típusai

- **Alaptípusok:**

  int, real/float, string, felsorolási típus és osztálytípus.

- **Konstruktorokkal összetett típusokat lehet definiálni. A konstruktorok a következők:**
  - **Struct , azaz rekordtípus konstruktor.**
  - ***Kollekciótípusú konstruktorok*: Set, Bag, List, Array és Dictionary (= egy T1 adattípusról T2 adattípusra képező függvény).**

- **A kapcsolatok típusa a következők valamelyike:**
  - **osztálytípus vagy**
  - **egy osztályra alkalmazott kollekciótípusú konstruktor.**

Öröklődés: az ODL alosztályai

- Az öröklődésben szereplő általános osztályt kettősponttal és a nevével adjuk meg.
- Az alosztálynak csak az általános osztályban nem szereplő attribútumait kapcsolatait, metódusait kell megadni, mert az általános osztály minden jellemzőjét örökli.
- Több általános osztálytól is örökölhet az alosztály.
- Probléma lehet, ha ugyanazt a jellemzőt, például attribútumot több helyről is örökli az osztály.

Példa: Alosztályok

- **Az Alkoholmentes a Sör alosztálya:**

```
class Alkoholmentes:Sör {
  attribute string szín;
  }
```

Kulcsok az ODL-ben

- **Egy osztályhoz tetszőleges számú kulcsot definiálhatunk.**

- **Az osztálynév után kell a kulcsokat megadni:**

  (key <kulcsok listája>)

- **Egy kulcs lehet egyszerű, azaz egy attribútumos, vagy összetett, ahol a kulcshoz tartozó attribútumokat zárójelek közé kell tenni.**

Példa: Kulcsok

```
class Ivó (key név) { …
```

- A név az Ivó egyetlen egyszerű kulcsa.

```
class Tárgy (key
  (tanszék,tárgykód),(terem, időpont)){…
```

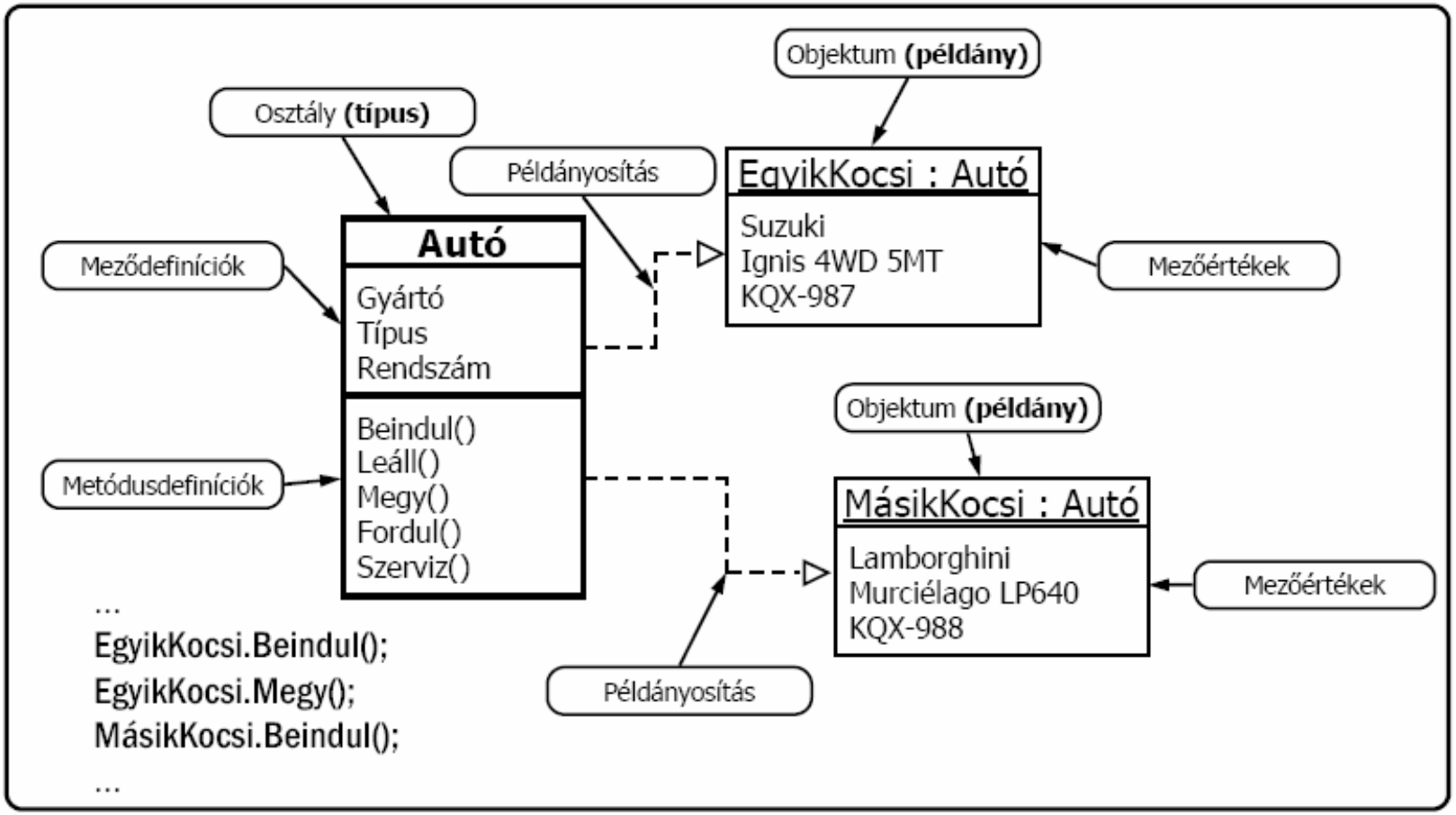
- Két összetett kulcs tartozik a Tárgy osztályhoz, a tanszék és tárgykód együtt alkot kulcsot, illetve a terem és időpont szintén együtt alkot kulcsot.

UML

- **Az UML-t alapvetően szoftverfejlesztéshez tervezték, de adatmodellező nyelvként is használható.**
- **Átmenetet jelent E/K modell és az ODL között.**
  - **Csak bináris kapcsolat adható meg, szemben az E/K modellel.**
  - **A bináris kapcsolatnak lehet saját attribútuma, szemben az ODL modellel.**
  - **Grafikus jelölésrendszere van, szemben az ODL modellel.**

Osztály

Példa: az Autó osztály és két példánya (UML)



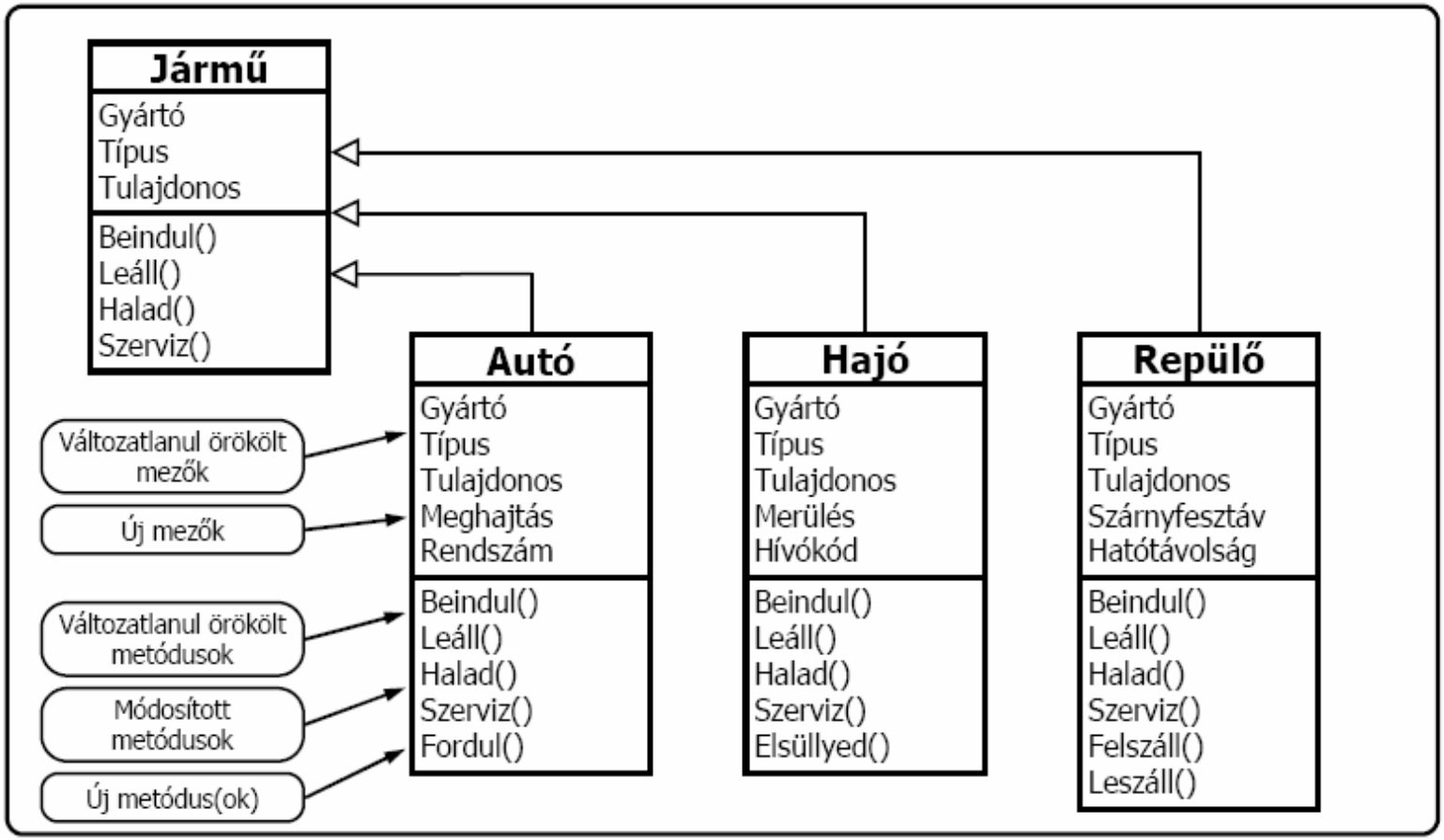
Osztályok közötti kapcsolatok

1. Leszármazás (ún. „IS-A” kapcsolat)

- Egy osztály leszármazottai „öröklik” az osztály jellemzőit
  - Rendelkeznek ugyanazokkal a mezőkkel
  - Tartalmazzák ugyanazokat a metódusokat
  - Ebben a specializációs kapcsolatban a leszármazottat „utódnak”, az eredeti osztályt „ősosztálynak” nevezzük
- A leszármazottak bővíthetik az ősosztály adatait és algoritmusait
  - Új mezőket adhatnak a meglévőkhöz
  - Új metódusokat definiálhatnak
  - Meglévő metódusok működését módosíthatják
- A leszármazottak minden műveletre képesek, amelyre az ősosztály képes volt
  - Az utódosztály példányai használhatók az ősosztály példányai helyett

Osztályok közötti kapcsolatok

Példa: leszármazás (UML)



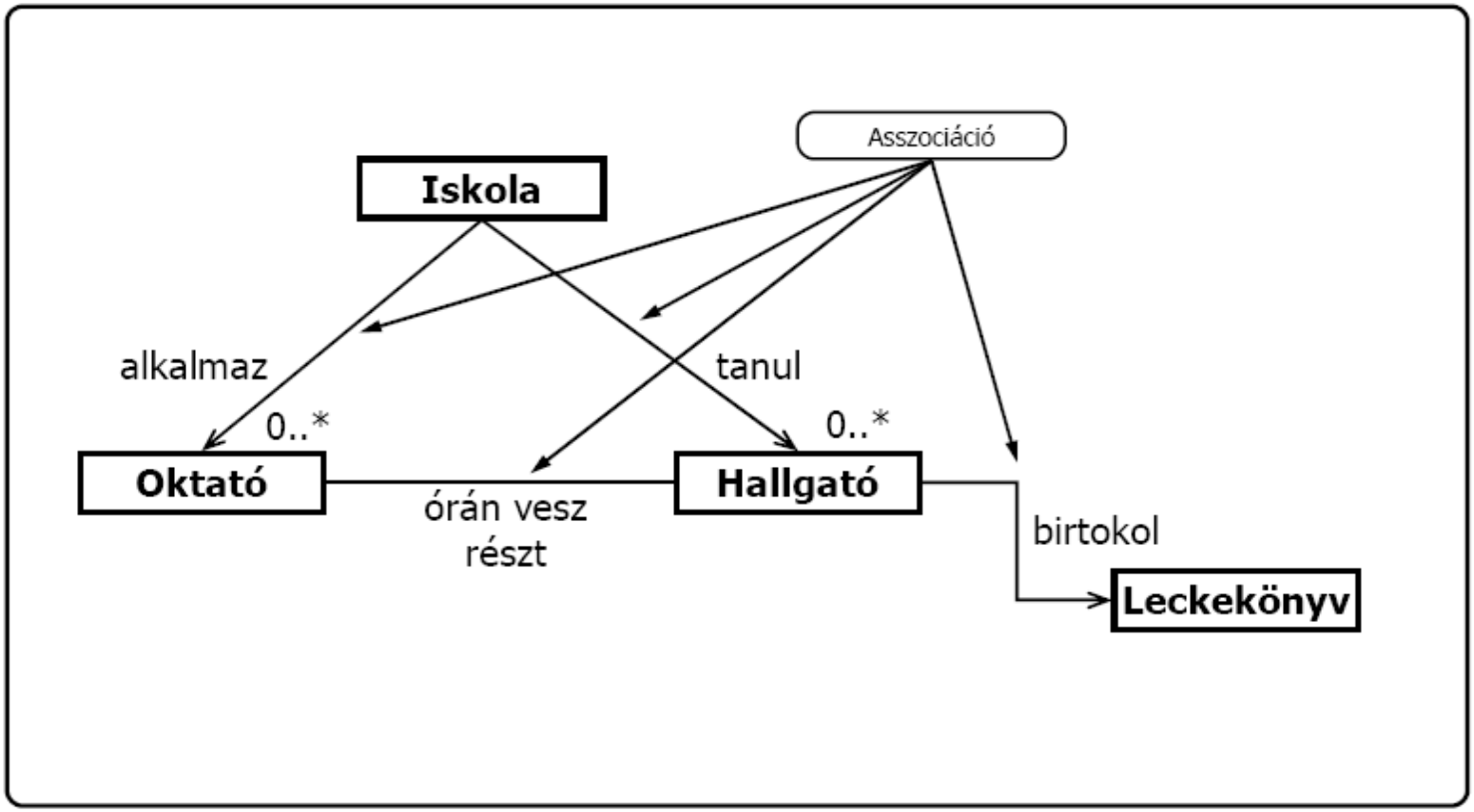
Osztályok közötti kapcsolatok

2. Asszociáció

- Az asszociáció osztályok közötti tetszőleges típusú viszony
  - Általában az asszociáció konkrét elnevezése fejezi ki a viszonyt
    - Szerep (pl. az ember az autó **tulajdonosa**)
    - Cselekvés (pl. az ember **vezeti** az autót)
  - Multiplicitás: vannak „egy-egy", „egy-több" és „több-több" típusú asszociációk
  - Irányultság: az asszociáció lehet egy- vagy kétirányú
- Asszociációs kapcsolat áll fenn két osztály között, ha az egyiknek a saját helyes működéséhez ismernie kell a másikat
  - Példa: egy osztály használ egy másik osztályt (ún. „USES-A" kapcsolat)

Osztályok közötti kapcsolatok

Példa: asszociáció (UML)



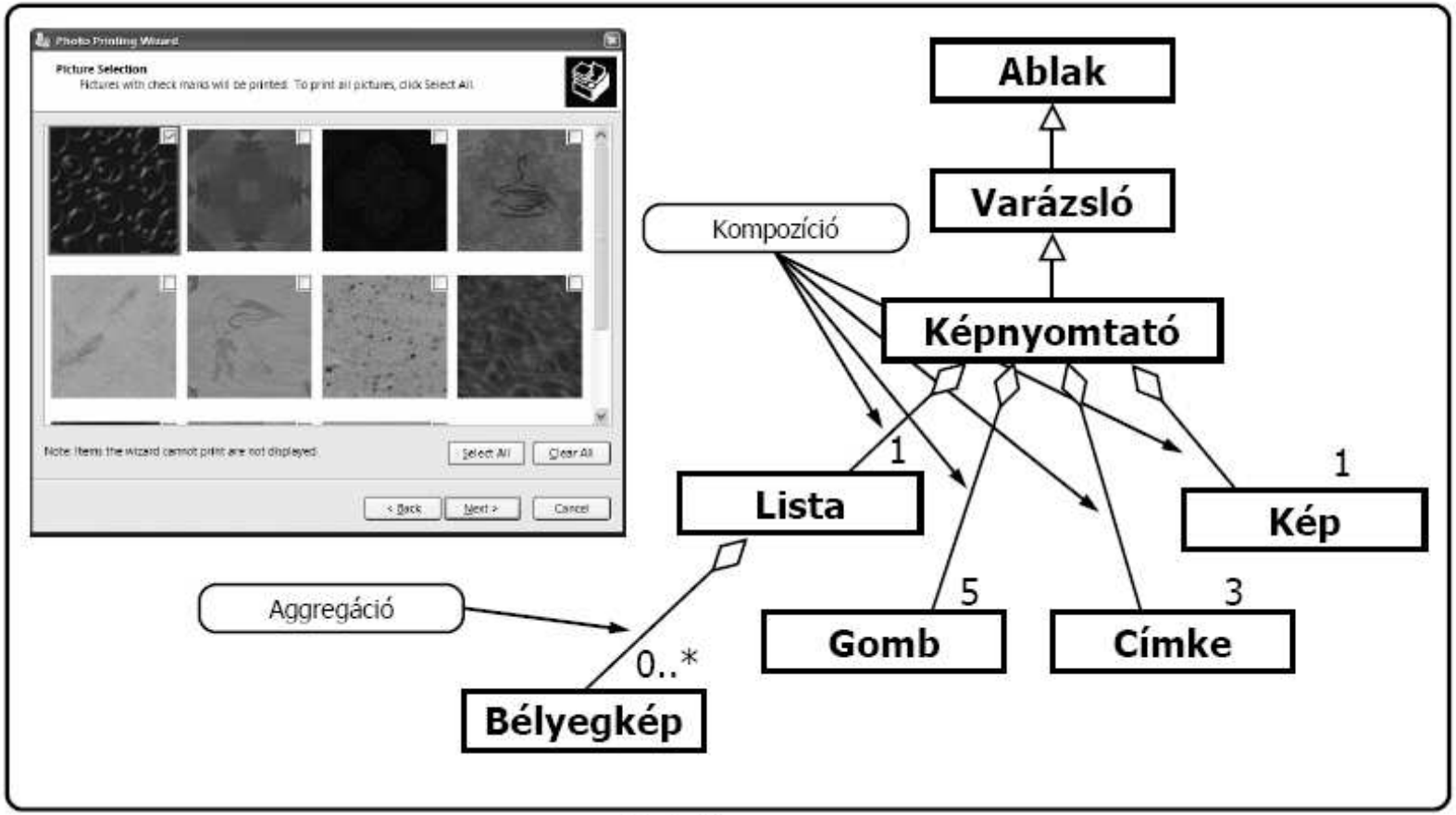
Osztályok közötti kapcsolatok

3. Aggregáció és kompozíció (ún. „HAS-A” kapcsolat)

- Az aggregáció az asszociáció speciális esete: tartalmazási kapcsolat
  - A kapcsolat aszimmetrikus és tranzitív
  - A tartalmazó osztály példányai magukban foglalják a tartalmazott osztály egy vagy több példányát
  - A tartalmazó és a tartalmazott osztály egymástól függetlenül létezhetnek
    - A tartalmazott átveheti (de nem örökli) a tartalmazó egyes jellemzőit
- A kompozíció az aggregáció speciális esete: szigorú tartalmazási kapcsolat
  - Egy tartalmazottnak mindig csak egy tartalmazója lehet
    - Egy tartalmazó viszont tetszőleges számú tartalmazott példánnyal rendelkezhet
  - A tartalmazó és a tartalmazott életciklusa közös

Osztályok közötti kapcsolatok

Példa: aggregáció és kompozíció (UML)



Az OOP néhány csapdája

Mire kell ügyelni az osztályok kialakításánál?

- Rossz döntés az osztályok közötti kapcsolat típusának megválasztásánál
  - Pl. leszármazás helyett aggregáció vagy fordítva
- Leszármazási kapcsolat helytelen kialakítása
  - Az ősosztály túl keveset vagy túl sokat „tud”
- Önkéntelenül (nem látható módon) beépített feltételezések
  - Az ősosztály feltételez bizonyos állapotváltozásokat, a metódusok végrehajtási sorrendjét – ez nem biztos, hogy a leszármazottaknál is igaz lesz
- „Felduzzasztott” osztályok
  - A túl sok mező és metódus, a túl hosszú metódusok azt jelezhetik, hogy valamilyen módon fel kell bontani az osztályt
- Elnevezési anomáliák

UML osztályok

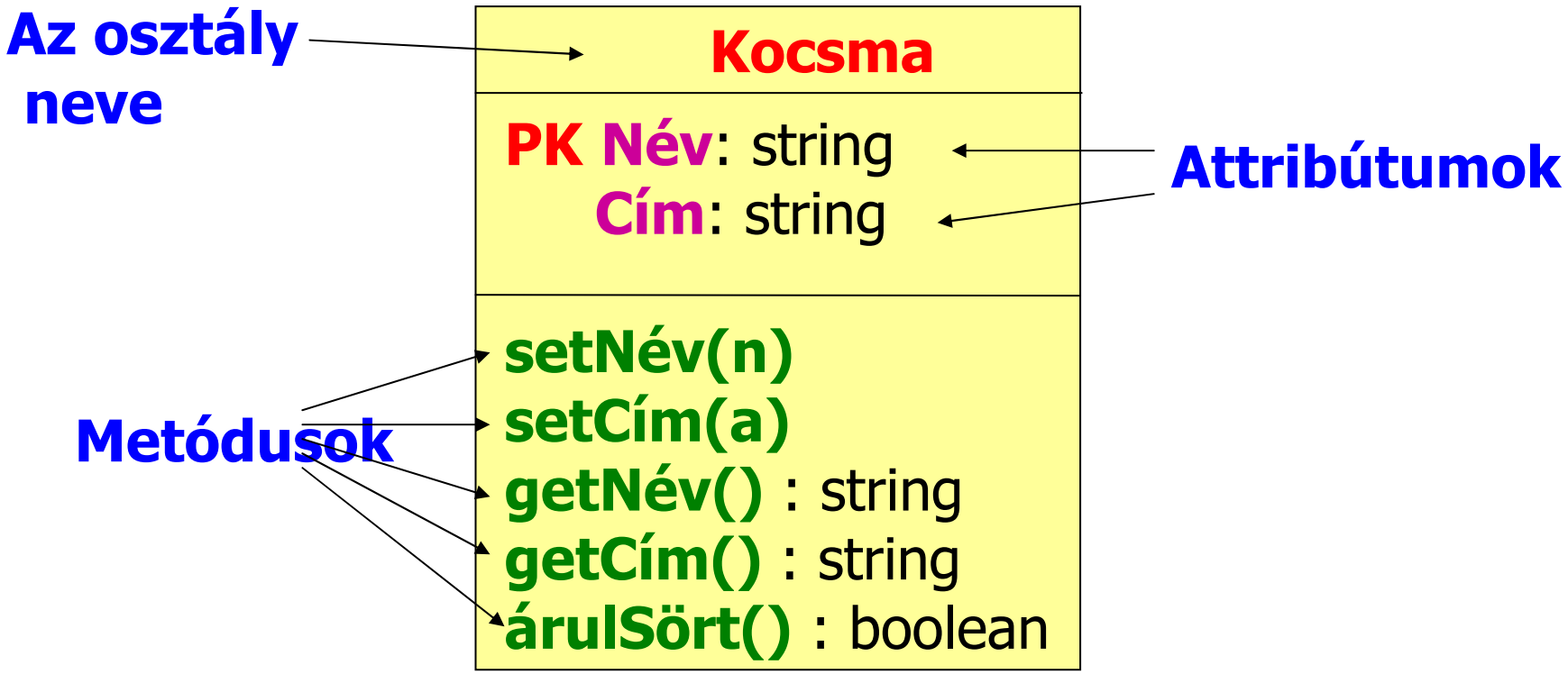
- **Az osztály objektumok halmaza, attribútumokkal, másképpen állapotokkal (*state*) és metódusokkal, másképpen viselkedéssel (*behavior*).**

- **Az attribútumokhoz típusok tartoznak.**

- **PK jelöli az attribútumot, ha ez az attribútum az objektum elsődleges kulcsához tartozik.**

- **A metódusok deklarációját kell megadni, azaz az argumentumokat, ha van egyáltalán, és a visszatérési érték típusát.**

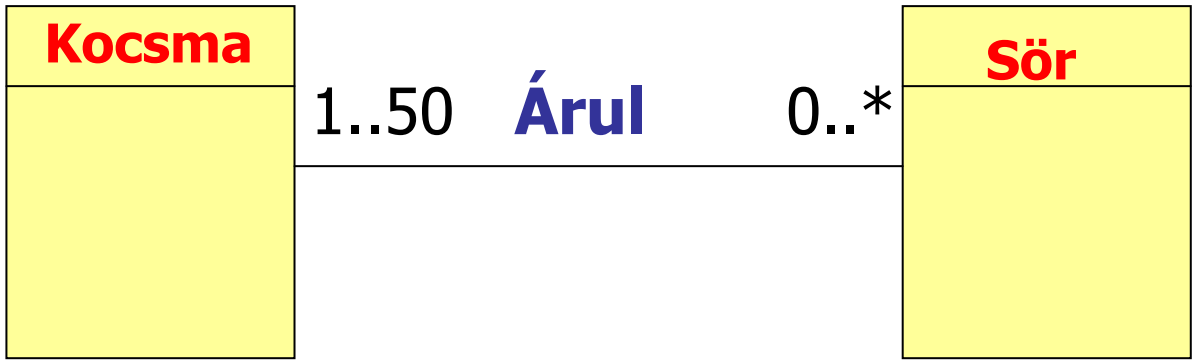
Példa: Kocsma osztály



Asszociációk

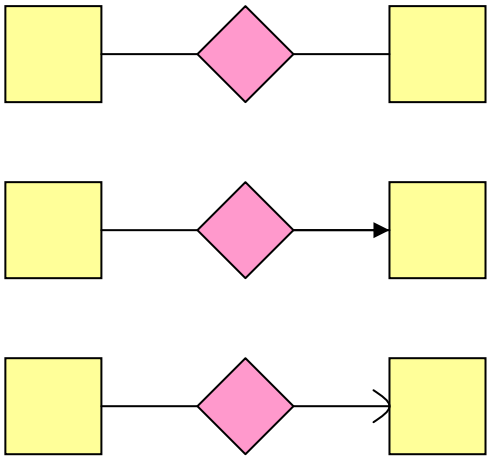
- **Az asszociáció osztályok közti bináris kapcsolatot jelent.**
- **Vonallal jelöljük, amelyre ráírjuk az asszociáció nevét (nem kell rombusz mint az E/K modellben).**
- **Mindkét végére felírjuk a megfelelő multiplicitást.**
  - ***m..n*** **azt jelenti, hogy a másik oldal egy objektumához *m* és *n* közötti objektum tartozik ennél az asszociációnál.**
  - *** = "végtelen"; például 1..* jelentése "legalább 1".**

Példa: Asszociáció

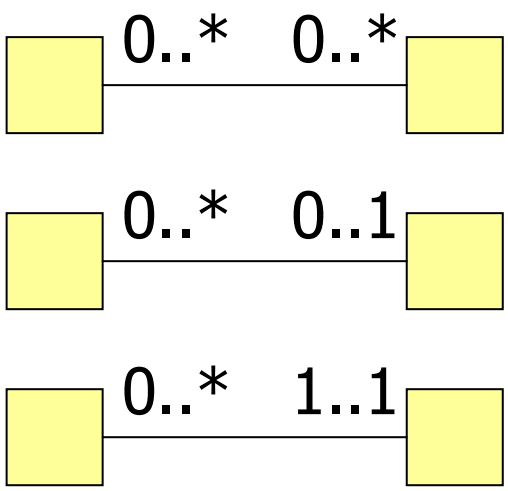


Összehasonlítás az E/K multiplicitásaival

E/R



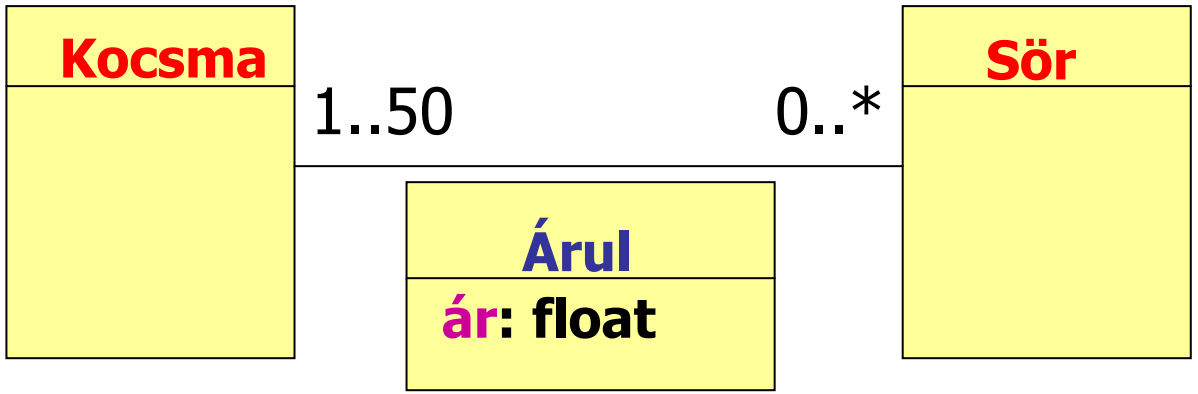
UML



Asszociációs osztályok

- **Az asszociációknak lehet saját attribútumuk.**
  - **Ebben az esetben *asszciációs osztályról beszélünk*.**
  - **Az E/K modellben is megengedtük, hogy egy kapcsolatnak lehessen saját attribútuma.**

Példa: Asszociációs osztály

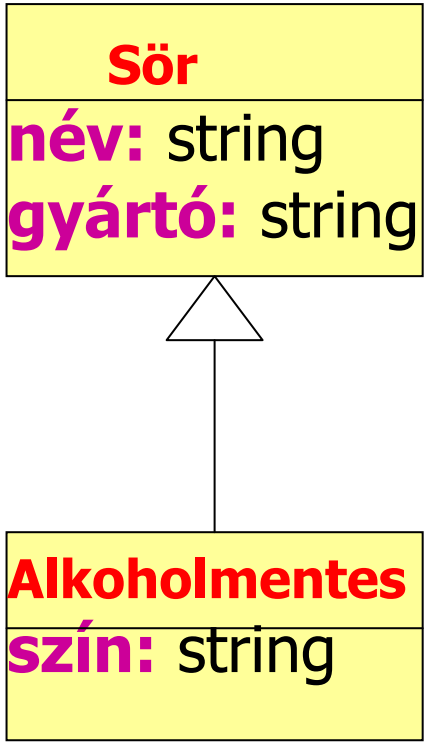


Öröklődés: alosztályok

- **Hasonló az E/K modellhez, csak más a jelölés, az alosztálytól vezet egy vonal az általános osztályhoz, és a vonal háromszögben végződik.**

- **Egy általános osztályhoz tartozó alosztályok halmaza a következő lehet:**
  - ***Teljes (Complete)*** **(minden objektum legalább egy alosztályhoz tartozik) vagy különben *részleges (partial)*.**
  - ***Diszjunkt (Disjoint)*** **(minden objektum legfeljebb egy alosztályban szerepelhet) vagy különben *átfedő (overlapping)*.**

Példa: Alosztályok



Az öröklödés reprezentálása relációkkal

- **Az E/K modellhez hasonlóan háromféleképpen lehet az általános osztályt és az alosztályait relációkká alakítani:**
  1. **E/K-stílusban: minden alosztályhoz tartozó reláció csak a saját attribútumait tárolja, és egy azonosítót (kulcsot).**
  2. **OO-stílusban: minden alosztályhoz tartozó reláció a saját attribútumain kivül az általános osztály összes attribútumát is tartalmazza.**
  3. **Nullértékekkel: Egy relációban tároljuk az általános osztály objektumait és az alosztályok objektumait is, és ahol nincs értelme az attribútumnak, oda nullértéket írunk.**

Aggregáció

- Olyan speciális kapcsolat, amely azt jelenti, hogy az egyik oldalon szereplő objektum a másik oldalon szereplő objektum része, vagy az objektum birtokolja.

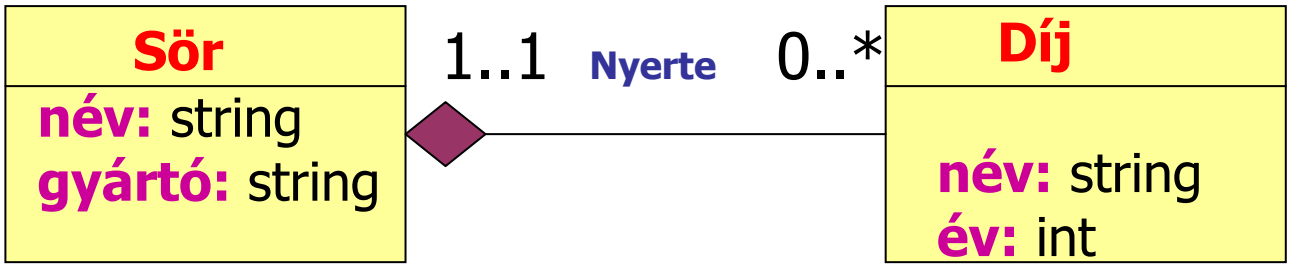
- Jelölésben egy rombuszt teszünk a "tulajdonos" oldalára.

Példa: Aggregáció

Kompozíció

- **Az aggregáció speciális esete, azzal a különbséggel, hogy minden objektumot pontosan egy objektum birtokolhat a másik oldalról.**
- **Tömör rombusszal jelöljük.**
- **Felhasználható részobjektumok, vagy struktúrával rendelkező attribútumok reprezentálásához.**

Példa: Kompozíció

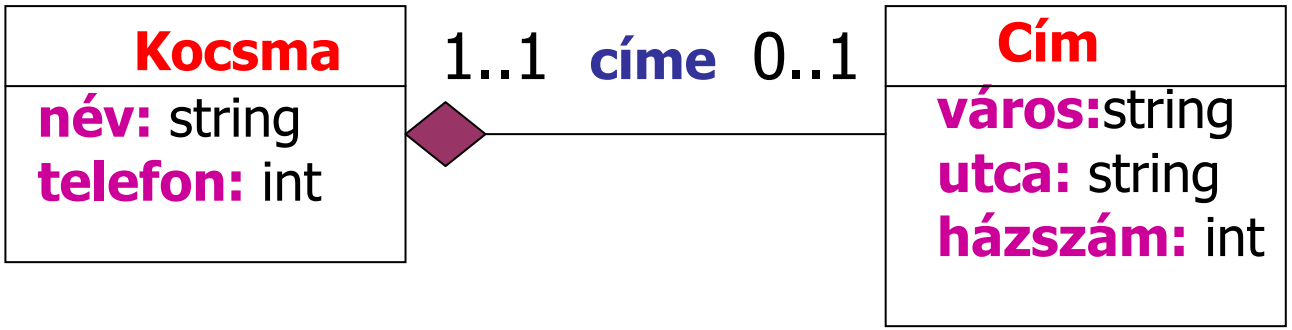


Reprezentálás relációs adatmodellben

- Egy sör által nyert díjak adatait a sörnek megfelelő sorban tároljuk.

- Objektum-relációs modellre vagy beágyazott relációs modellre (ahol a sorokban szereplő értékek táblázatok is lehetnek) van szükség, ha nem lehet korlátozni, hogy hány darab díjat nyerhet egy sörfajta.

Példa: Kompozíció



Reprezentálás relációs adatmodellben

- **Relációs adatmodellben:**
  - **Mivel egy kocsmának legfeljebb egy címe lehet, így a címhez tartozó város, utca, házszám attribútumokat felvehetjük a kocsmáknak megfelelő táblában.**
- **Objektum-relációs adatmodellben:**
  - **A strukturával (város, utca, házszám) rendelkező cím attribútumot vegyük fel a kocsmáknak megfelelő relációban.**

Redundáns relációsémák

Tekintsük egy vállalat dolgozóit nyilvántartó
DOLGOZÓ (Név, Adószám, Cím, Osztálykód, Osztálynév, VezAdószám)
sémát.

Előny: egyetlen táblában a dolgozók és osztályok adatai is nyilvántartva.
Hátrány: redundancia, mivel Osztálynév, VezAdószám több helyen szerepel.

Név	*Adószám*	*Cím*	*Osztálykód*	*Osztálynév*	*VezAdószám*
Kovács	1111	Pécs, Vár u.5.	2	Tervezési	8888
Tóth	2222	Tata, Tó u.2.	1	Munkaügyi	3333
Kovács	3333	Vác, Róka u.1.	1	Munkaügyi	3333
Török	8888	Pécs, Sas u.8.	2	Tervezési	8888
Kiss	4444	Pápa, Kő tér 2.	3	Kutatási	4444
Takács	5555	Győr, Pap u. 7.	1	Munkaügyi	3333
Fekete	6666	Pécs, Hegy u.5.	3	Kutatási	4444
Nagy	7777	Pécs, Cső u.25.	3	Kutatási	4444

Redundáns relációsémák

Név	*Adószám*	*Cím*	*Osztálykód*	*Osztálynév*	*VezAdószám*
Kovács	1111	Pécs, Vár u.5.	2	Tervezési	8888
Tóth	2222	Tata, Tó u.2.	1	Munkaügyi	3333
Kovács	3333	Vác, Róka u.1.	1	Munkaügyi	3333
Török	8888	Pécs, Sas u.8.	2	Tervezési	8888
Kiss	4444	Pápa, Kő tér 2.	3	Kutatási	4444
Takács	5555	Győr, Pap u. 7.	1	Munkaügyi	3333
Fekete	6666	Pécs, Hegy u.5.	3	Kutatási	4444
Nagy	7777	Pécs, Cső u.25.	3	Kutatási	4444

A redundancia aktualizálási anomáliák okozhat:

(i) Módosítás esetén:

– Ha egy osztály neve vagy vezetője megváltozik, több helyen kell a módosítást elvégezni.

(ii) Új felvétel esetén:

– új dolgozó felvételénél előfordulhat, hogy az osztálynevet máshogy adják meg (például *Tervezési* helyett *tervezési* vagy *Tervező*).

– Ha új osztály létesül, amelynek még nincsenek alkalmazottai, akkor ezt csak úgy tudjuk felvenni, ha a (név, adószám, cím) mezőkhöz 'null' értéket veszünk. Később, ha lesznek alkalmazottak, ez a rekord fölöslegessé válik.

(iii) Törlés esetén:

– Ha egy osztály valamennyi dolgozóját töröljük, akkor az osztályra vonatkozó információk is elvesznek.

Redundáns relációsémák

Megoldás:
a DOLGOZÓ (Név, Adószám, Cím, Osztálykód, Osztálynév, VezAdószám)
séma szétválasztása (dekompozíció):
DOLG (Név, Adószám, Cím, Osztálykód)
OSZT (Osztálykód, Osztálynév, VezAdószám)

Név	*Adószám*	*Cím*	*Osztálykód*
Kovács	1111	Pécs, Vár u.5.	2
Tóth	2222	Tata, Tó u.2.	1
Kovács	3333	Vác, Róka u.1.	1
Török	8888	Pécs, Sas u.8.	2
Kiss	4444	Pápa, Kő tér 2.	3
Takács	5555	Győr, Pap u. 7.	1
Fekete	6666	Pécs, Hegy u.5.	3
Nagy	7777	Pécs, Cső u.25.	3

Osztálykód	*Osztálynév*	*VezAdószám*
1	Munkaügyi	3333
2	Tervezési	8888
3	Kutatási	4444

Funkcionális függőség

Definíció. Legyen R(U) egy relációséma, továbbá X és Y az U attribútumhalmaz részhalmazai. X-től *funkcionálisan függ* Y (jelölésben $X \rightarrow Y$), ha bármely R feletti T tábla esetén valahányszor két sor megegyezik X-en, akkor megegyezik Y-on is. Ez lényegében azt jelenti, hogy az X-beli attribútumok értéke egyértelműen meghatározza az Y-beli attribútumok értékét.

Elnevezések:

– Az $X \rightarrow Y$ függést *triviális*nak nevezzük, ha Y részhalmaza X-nek, ellenkező esetben *nemtriviális.*

– Az $X \rightarrow Y$ függést *teljesen nemtriviális*nak nevezzük, ha X és Y nem tartalmaz közös attribútumot ($X \cap Y = 0$*).*

A gyakorlatban általában teljesen nemtriviális függőségeket adunk meg.

DOLGOZÓ (Adószám, Név, Cím, Osztálykód, Osztálynév, VezAdószám)
tábla jellemző függőségei:
f1: {Adószám} $\rightarrow$ {Név, Cím, Osztálykód}
f2: {Osztálykód} $\rightarrow$ {Osztálynév, VezAdószám}

Példa további függőségekre:
f3: {Adószám} $\rightarrow$ {Osztálynév}
f4: {Cím, Osztálykód} $\rightarrow$ {VezAdószám}

Funkcionális függőség

Példa. **Egy számla tételeit tartalmazó**
SZÁMLA(cikkszám, megnevezés, egységár, mennyiség, összeg)
tábla esetén az alábbi függőségeket állapíthatjuk meg:
{cikkszám} → {megnevezés, egységár}
{egységár, mennyiség} → {összeg}

Megjegyzések:
– A függőség nem az aktuális tábla, hanem a séma tulajdonsága. Ha az attribútumhalmazra megállapítunk egy funkcionális függőséget, akkor ez egy feltételt jelent az adattáblára nézve. Ha pl. Adószám → Cím funkcionális függőség fennáll, akkor egy személyhez több lakcímet nem tudunk tárolni.

– A "funkcionális" kifejezés arra utal, hogy ha X → Y fennáll, akkor adott tábla esetén létezik egy dom(X) → dom(Y) függvény, amely X minden konkrét értékéhez egyértelműen meghatározza Y értékét. Ez a függvény általában csak elméletileg létezik, pl. Adószám → Cím függés esetén nem tudunk olyan algoritmust adni, amely az adószámból a lakcímet előállítaná. A SZÁMLA tábla esetén azonban az {egységár, mennyiség} → {összeg} függőség már számítható, mivel *egységár*mennyiség = összeg* teljesül.

Funkcionális függőség

- **Belátható, hogy egy R(U) relációséma esetén az *A* attribútumhalmaz egy K részhalmaza akkor és csak akkor szuperkulcs, ha a K → U függés teljesül. A kulcsot tehát a függőség fogalma alapján is lehet definiálni: olyan K attribútumhalmazt nevezünk kulcsnak, amelytől az összes többi attribútum függ, de K-ból bármely attribútumot elhagyva ez már nem teljesül.**

- **A DOLGOZÓ sémához a fentiekben az f1, f2, f3 és f4 függőségeket írtuk fel. Vegyük észre, hogy valójában csak az f1, f2 függések fontosak, f3 és f4 ezekből származtatható, és még számos további származtatott függést írhatnánk fel. *Bázis*nak (vagy fedésnek) nevezzük azt a függéshalmazt, amelyből az összes többi függés levezethető. A DOLGOZÓ séma esetén tehát {f1, f2} alkotja a bázist. Általában minimális bázist keresünk.**

- **A fentiek alapján egy R(U, F) párt nevezhetünk relációsémának, ahol U egy attribútumhalmaz, F pedig az U-n érvényes függőségek bázisa. F alapján a séma kulcsai algoritmikusan meghatározhatók.**

Armstrong-axiómák

Legyen $X,Y \subseteq R$, és XY jelentse az X és Y attribútumhalmazok egyesítését.
F legyen funkcionális függőségek tetszőleges halmaza.

Armstrong axiómák:

- **A1 (reflexivitás): $Y \subseteq X$ esetén $X \rightarrow Y$.**
- **A2 (tranzitivitás): $X \rightarrow Y$ és $Y \rightarrow Z$ esetén $X \rightarrow Z$.**
- **A3 (bővíthetőség): $X \rightarrow Y$ és tetszőleges Z esetén $XZ \rightarrow YZ$.**

- **$X \rightarrow Y$ levezethető F-ből, ha van olyan $X1 \rightarrow Y1$, ..., $Xk \rightarrow Yk$,..., $X \rightarrow Y$ véges levezetés, hogy $\forall$k-ra**
  - **$Xk \rightarrow Yk \in F$ vagy**
  - **$Xk \rightarrow Yk$ az A1,A2,A3 axiómák alapján kapható a levezetésben előtte szereplő függőségekből.**

- **Jelölés: $F \vdash X \rightarrow Y$, ha $X \rightarrow Y$ levezethető F-ből**

Armstrong-axiómák

- További levezethető szabályok:

1. (Egyesítési szabály): F|—X→Y és F|—X→Z esetén F|—X→YZ.
2. (Pszeudotranzitivitás): F|—X→Y és F|—WY→Z esetén F|—XW→Z.
3. (Dekomponáló szabály): F|—X→Y és Z⊆ Y esetén F|—X→Z.

Bizonyítás (1): Bővítési axióma miatt F|—XX→YX és F|—YX→YZ, és XX=X, valamint a tranzitivitási axióma miatt F|—X→YZ.

Bizonyítás (2): Bővítési axióma miatt F|—XW→YW, és YW=WY, valamint a tranzitivitási axióma miatt F|—XW→Z.

Bizonyítás (3): Reflexivitási axióma miatt F|—Y→Z, és tranzitivitási axióma miatt F|—X→Z.

- Következmény: F|—X→Y ⇔ ∀Ai∈ Y esetén F|—X→Ai

- A következmény miatt feltehető, hogy a függőségek jobb oldalai 1 attribútumból állnak.

- Definíció: F-nek X→Y logikai következménye, ha minden olyan táblában, amelyben F összes függősége teljesül, X→Y is teljesül.

- Tétel: Az Armstrong-axiómarendszer helyes és teljes, azaz minden levezethető függőség logikai következmény, illetve minden logikai következmény levezethető.

Attribútumhalmaz lezárása

- **A bal oldalak nem szedhetők szét:**

- $\{AB \rightarrow C\} \not\vdash A \rightarrow C$.

- **Ehhez elég egy ellenpélda**
  ***(mert az axiómarendszer helyes és teljes)*:**

  r-ben teljesül $AB \rightarrow C$, de
  r-ben nem teljesül $A \rightarrow C$.

A	B	C
0	0	0
0	1	2
0	2	1

- $\{A \rightarrow C\} \vdash AB \rightarrow C$ **viszont igaz:**

  $A \subseteq AB$ **miatt** $\{A \rightarrow C\} \vdash AB \rightarrow A$,
  és $\{A \rightarrow C\} \vdash A \rightarrow C$, **majd a tranzitivitás axiómát alkalmazzuk.**

(R,F) séma esetén legyen $X \subseteq R$.

Definíció: $X^{*(F)} := \{A \mid F \vdash X \rightarrow A\}$ az X attribútumhalmaz lezárása F-re nézve.

Attribútumhalmaz lezárása

Következmény: $F \vdash X \rightarrow Y \Leftrightarrow Y \subseteq X^*$.

Bizonyítás:

($\Rightarrow$) $\forall A \in Y$ esetén a reflexivitás és tranzitivitás miatt $F \vdash X \rightarrow A$, azaz $Y \subseteq X^*$.

($\Leftarrow$) $\forall A \in Y \subseteq X^*$ esetén $F \vdash X \rightarrow A$, és az egyesítési szabály miatt $F \vdash X \rightarrow Y$. q.e.d.

- **Megjegyzés:** Az X^* lezárási operátor, azaz
  1. $X \subseteq X^*$
  2. $X \subseteq Y$ esetén $X^* \subseteq Y^*$
  3. $X^{**}=X^*$.

Attribútumhalmaz lezárása

$$F \vdash X \rightarrow Y \Leftrightarrow Y \subseteq X^*$$

- **Az implikációs probléma megoldásához elég az X*-ot hatékonyan kiszámolni.**

- **Algoritmus X* kiszámítására:**

  /* Iteráció, amíg X(n) változik */

  $X(0) := X$

  $X(n+1) := X(n) \cup \{A \mid Y \rightarrow Z \in F, A \in Z, Y \subseteq X(n)\}$

  Ha $X(v+1)=X(v)$, akkor $X(v)=X^*$.

- **Ha az (R,F,X) input leírási hossza k, akkor az algoritmus lépésszáma $O(k^2)$.**
- **A hatékonyság megfelelő könyveléssel lineárissá, O(k) lépésűvé tehető.**

Attribútumhalmaz lezárása

R=ABCDEFG, {AB→C, B→G, CD→EG, BG →E}

X=ABF, X*=?

X(0):=ABF

X(1):=ABF∪{C,G}=ABCFG

X(2):=ABCFG ∪{C,G,E}=ABCEFG

X(3):=ABCEFG

X*=X(2)=ABCEFG

Következmény: Hatékonyan ellenőrizhető, hogy egy oszlophalmaz szuperkulcs-e.

R(U)-nak K⊆U szuperkulcsa F-re nézve:

- $F \vdash K \rightarrow U \Leftrightarrow U \subseteq K^*$

R(U)-nak K⊆U kulcsa F-re nézve:

- $F \vdash K \rightarrow U \Leftrightarrow U \subseteq K^*$
- minden X ⊂ K valódi részhalmazra:

  $F \not\vdash X \rightarrow U \Leftrightarrow U \neq X^*$

Felbontás (dekompozíció)

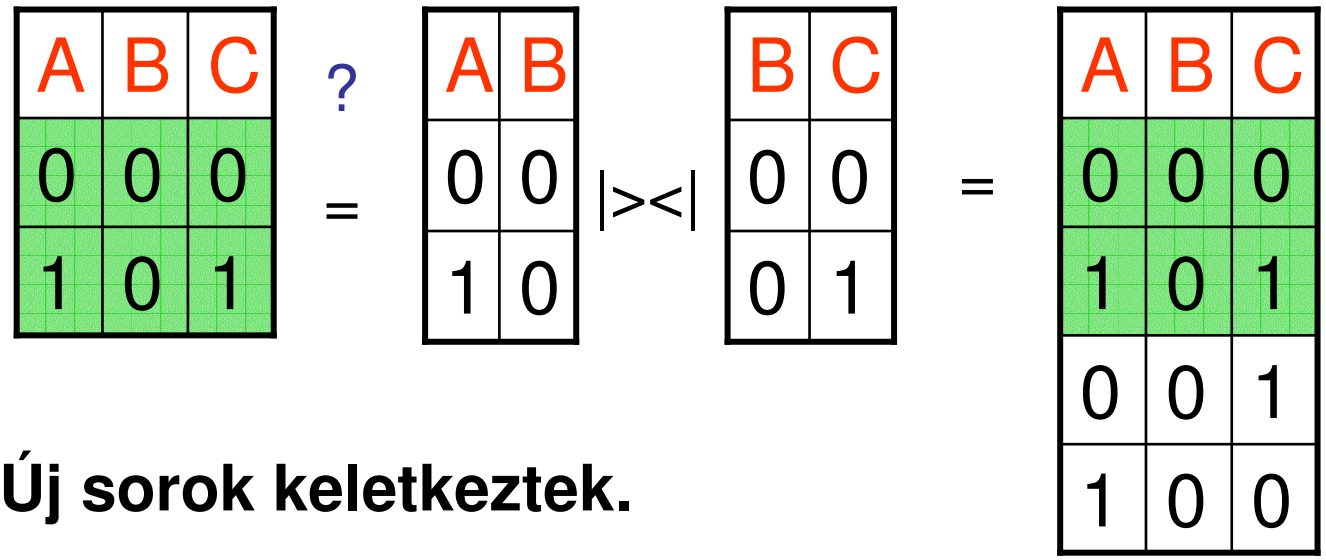
Definíció: (Az adattábla felbontását projekcióval végezzük). d={R1,...,Rk} az (R,F) dekompozíciója, ha nem marad ki attribútum, azaz R1∪...∪Rk=R.

Például:
R=ABCDE, d={AD,BCE,ABE} 3 tagú dekompozíció, ahol R1=AD, R2=BCE, R3=ABE,

Definíció:
d={R1,...,Rk} az (R,F) veszteségmentes dekompozíciója, ha minden olyan r-re, amelyben F összes függősége teljesül igaz hogy a dekompozícióban szereplő sémákra vett vetületek természetes összekapcsolása (|><|) megegyezik az r táblával:

$$r = \Pi_{R1}(r) |><| \ldots |><| \Pi_{Rk}(r)$$

Veszteségmentes dekompozíció



- **Új sorok keletkeztek.**
- **$r \subseteq \Pi_{R1}(r)|><|...|><| \Pi_{Rk}(r)$ mindig teljesül:**

A veszteségmentességhez elég ellenőrizni, hogy minden olyan r-re, amelybe F összes függősége teljesül $r \supseteq \Pi_{R1}(r)|><|...|><| \Pi_{Rk}(r)$.

Veszteségmentes dekompozíció

- *Példa*. Nem veszteségmentes felbontást kapunk, ha a DOLGOZÓ táblát a *VezAdószám* mentén bontjuk fel:
  DOLG (Név, Adószám, Cím, VezAdószám)
  OSZT (Osztálykód, Osztálynév, VezAdószám)
- Ugyanis a DOLGOZÓ definiálásakor megengedtük, hogy egy személy több osztálynak is vezetője legyen (vagyis nincs VezAdószám → Osztálykód függés). Ha például Takács dolgozó az 1-es osztályon dolgozik, de ennek vezetője azonos az 5-ös osztály vezetőjével, akkor a DOLG|><|OSZT táblában Takács kétszer fog szerepelni: egyszer az 1-es, egyszer az 5-ös osztály dolgozójaként. ***A DOLGOZÓ tábla:***

A DOLGOZÓ tábla:

Név	Adószám	Cím	Osztálykód	Osztálynév	VezAdószám
Takács	5555	Győr, Pap u. 7.	1	Munkaügyi	3333
Rácz	9999	Vác, Domb u. 1.	5	Pénzügyi	3333

A DOLG tábla:

Név	Adószám	Cím	VezAdószám
Takács	5555	Győr, Pap u. 7.	3333
Rácz	9999	Vác, Domb u. 1.	3333

Az OSZT tábla:

Osztálykód	Osztálynév	VezAdószám
1	Munkaügyi	3333
5	Pénzügyi	3333

A DOLG|><|OSZT tábla:

Név	Adószám	Cím	Osztálykód	Osztálynév	VezAdószám
Takács	5555	Győr, Pap u. 7.	1	Munkaügyi	3333
Takács	5555	Győr, Pap u. 7.	5	Pénzügyi	3333
Rácz	9999	Vác, Domb u. 1.	1	Munkaügyi	3333
Rácz	9999	Vác, Domb u. 1.	5	Pénzügyi	3333

Veszteségmentes dekompozíció

- **2 részre vágás esetén a veszteségmentesség ellenőrzése az alábbi tétel alapján történhet:**
- **Tétel: d=(R1,R2) veszteségmentes F-re akkor és csak akkor, ha**

  $F \vdash R1 \cap R2 \rightarrow R1 - R2$ vagy

  $F \vdash R1 \cap R2 \rightarrow R2 - R1$.

Például: d={AB,BC} és F={$B \rightarrow C$} esetén

$AB \cap BC \rightarrow BC - AB$

mivel $AB \cap BC = B$ és $BC - AB = C$

Veszteségmentes dekompozíció

- **A DOLGOZÓ (név, adószám, cím, osztálykód, osztálynév, vezAdószám) tábla felbontása a**

  DOLG (név, adószám, cím, osztálykód)

  OSZT (osztálykód, osztálynév, vezAdószám)

  táblákra veszteségmentes, mert az

  {osztálykód} → {osztálynév, vezAdószám} függőség teljesül.

  Ugyanis ez a függőség pont a DOLG ∩ OSZT → OSZT – DOLG

Veszteségmentes dekompozíció

- **Chase algoritmus a veszteségmentesség eldöntésére (több tagú dekompozíció esetén):**

  INPUT: (R(A1,...,An),F,d=(R1,...,Rk))

  OUTPUT: d veszteségmentes F-re $\Leftrightarrow (a_1,...,a_n) \in r(v)$.

  MÓDSZER: Képezünk egy r(0),r(1),r(2),...,r(v) relációkból álló sorozatot.

  Az r(0) kiindulási reláció ti sor j-ik eleme (i=1..k, j=1..n):

  - **$t(i,j) := a_j$, ha $Aj \in Ri$,**
  - **$t(i,j) := b_{i,j}$,ha $Aj \notin Ri$.**

  A sorozatban az r(p+1) relációt úgy kapjuk, hogy az F valamelyik alkalmazható függőségét alkalmazzuk az r(p)-re, és ezt addig csináljuk, amíg lehet, azaz r(v)-re már F egyik függősége sem alkalmazható.

Veszteségmentes dekompozíció

Egy $X \rightarrow Aj \in F$ alkalmazható r(p)-re, ha r(p) megsérti az $X \rightarrow Aj$ függőséget.

Az alkalmazás azt jelenti, hogy értékeket azonosítunk a reláció Aj oszlopában, pontosabban

> **$t,t' \in r(p)$, és $t[X]=t'[X]$, de $t(Aj) \neq t'(Aj)$, akkor a $t(Aj)$ és $t'(Aj)$ érték közül az egyiket lecseréljük a másikra a teljes Aj oszlopban.**

Preferencia a cserénél:

- ha mindkettő $b._j$ szimbólum, akkor mindegy, hogy melyiket cseréljük a másikra,

- ha egyik a_j, a másik $b._j$ szimbólum, akkor a $b._j$-t cseréljük a_j-re.

- az algoritmus garantálja, hogy az Aj oszlopban csak j indexű a_j szerepelhet, így a különböző a_j-k esete nem fordulhat elő.

MÓDSZER VÉGE.

Veszteségmentes dekompozíció

Példa: d={AB,BC} és F={B →C} alkalmazásával kapjuk r(0)-ból r(1)-et, ami tovább nem változik. $a_1a_2a_3$ benne van, tehát d veszteségmentes.

r(0)

A	B	C
a_1	a_2	~~$b_{1,3}$~~ a_3
$b_{2,1}$	a_2	a_3

r(1)

A	B	C
a_1	a_2	a_3
$b_{2,1}$	a_2	a_3

Függőségőrzés

- **A dekompozíciókban érvényes függőségekből következzen az eredeti sémára kirótt összes függőség.**
- **Emiatt nem kell összekapcsolni a dekomponált táblákat ahhoz, hogy biztosak legyünk benne, hogy az eredetileg kirótt összes függőség teljesül.**
- **Milyen függőségek lesznek érvényesek a dekompozíció sémáiban?**

- **Definíció. Függőségek vetülete:**

  Adott (R,F), és $Ri \subseteq R$ esetén:

  $$\Pi_{Ri}(F) := \{ X \rightarrow Y \mid F \vdash X \rightarrow Y , XY \subseteq Ri \}$$

Függőségőrzés

- **Definíció: Adott (R,F) esetén d=(R1,...,Rk) függőségőrző dekompozíció akkor és csak akkor, ha minden F-beli függőség levezethető a vetületi függőségekből:**

  minden $X \rightarrow Y \in F$ esetén

  $$\Pi_{R1}(F) \cup ... \cup \Pi_{Rk}(F) \vdash X \rightarrow Y$$

Függőségőrzés

- *Hogy lehet a függőségőrzést ellenőrizni?*
- $\Pi_{R1}(F) \cup ... \cup \Pi_{Rk}(F) \vdash X \rightarrow Y$ implikációt kell minden $X \rightarrow Y \in F$ függőségre ellenőrizni.

- Jelöljük G-vel a bal oldalt: $G := \Pi_{R1}(F) \cup ... \cup \Pi_{Rk}(F)$.
- $G \vdash X \rightarrow Y \Leftrightarrow Y \subseteq X^{*(G)}$ miatt elő kell állítani az F függőségeiben szereplő bal oldalak G szerinti lezárását.

- A probléma az, hogy G számossága nagyon nagy lehet, még kis számosságú F esetén is, hiszen F összes következményei közül kell kiválasztani a vetületekbe eső függőségeket.

- *Hogy lehet $X^{*(G)}$ –t kiszámolni anélkül, hogy G-t előállítanánk?*

Függőségőrzés

- **$X^{*(G)}$ kiszámítása G előállítása nélkül:**

  Z:=X

  while Z változik do

  for i:=1..k do

  $Z:=Z \cup ((Z \cap Ri)^{*(F)} \cap Ri)$

- **Az F szerinti lezárást hatékonyan tudjuk kiszámolni!**

Függőségőrzés

- **R=ABCD, F={A→B,B→C,C→D,D→A}, d={AB,BC,CD}**
  $A\to B\in\Pi_{AB}(F)$, $B\to C\in\Pi_{BC}(F)$, $C\to D\in\Pi_{CD}(F)$
  Vajon megőrzi-e a D→A függőséget?

Z=D
$Z=D\cup((D\cap AB)^*\cap AB)=D\cup(\varnothing^*\cap AB)=D$
$Z=D\cup((D\cap BC)^*\cap BC)=D\cup(\varnothing^*\cap BC)=D$
$Z=D\cup((D\cap CD)^*\cap CD)=D\cup(D^*\cap CD)=D\cup(ABCD\cap CD)=CD$
$Z=CD\cup((CD\cap AB)^*\cap AB)=CD\cup(\varnothing^*\cap AB)=CD$
$Z=CD\cup((CD\cap BC)^*\cap BC)=CD\cup(C^*\cap BC)=CD\cup(ABCD\cap BC)=BCD$
$Z=BCD\cup((BCD\cap CD)^*\cap CD)=BCD\cup(CD^*\cap CD)=BCD\cup(ABCD\cap CD)=BCD$
$Z=BCD\cup((BCD\cap AB)^*\cap AB)=BCD\cup(B^*\cap AB)=BCD\cup(ABCD\cap AB)=ABCD$

Tovább már nem változik a Z. Tehát $A\in D^{*(G)}=ABCD$, vagyis ezt a függőséget is megőrzi a dekompozíció.

Függőségőrzés

- **A függőségőrzésből nem következik a veszteségmentesség:**

  R=ABCD, F= {A→B,C→D}, d={AB,CD} függőségőrző, de nem veszteségmentes.

- **A veszteségmentességből nem következik a függőségőrzés:**

  R=ABC, F= {AB→C,C→A}, d={AC,BC} veszteségmentes, de nem függőségőrző,

  mert $C \notin AB^{*(G)}=AB$, az algoritmus alapján.

Normálformák (BCNF)

- A nem triviális függőségek redundanciát okozhatnak.
- A normálformák a redundanciát csökkentik.
- BCNF esetén csak kulcsok miatt függőségek maradhatnak.
- *Feltesszük, hogy a lehetséges relációkat leíró F halmazban már minden függőség jobb oldala 1 attribútumot tartalmaz.*

Két ekvivalens definíció:

- 1. Definíció. R relációséma Boyce-Codd normálformában (BCNF-ben) van az F-re nézve, ha tetszőleges $XA \subseteq R$, $A \notin X$ és $F \models X \rightarrow A$ esetén $F \models X \rightarrow R$ (azaz X az R szuperkulcsa F-re nézve).

- 2. Definíció. R relációséma Boyce-Codd normálformában (BCNF-ben) van az F-re nézve, ha tetszőleges $XA \subseteq R$, $A \notin X$ és $X \rightarrow A \in F$ esetén $F \models X \rightarrow R$ (azaz X az R szuperkulcsa F-re nézve).

- Adott (R,F) és d=(R1,...,Rk) esetén d az R BCNF dekompozíciója F-re nézve, ha minden i-re Ri BCNF a $\Pi_{Ri}(F)$-re nézve.

Normálformák (BCNF)

- ***Vajon van-e mindig függőségőrző BCNF dekompozíció?***

- **R=ABC, F={AB→C,C→A}**
  R kulcsai: AB, BC (lezárási algoritmussal ellenőrizhető).
  Az F függőségeinek baloldalai közül AB tartalmaz kulcsot, de C nem, azaz R nincs BCNF-ben F-re, ezért nem triviális módon dekomponálni kell BCNF sémákra. Az összes lehetséges felbontást végignézve az AB→C függőséget nem fogja megőrizni egyik dekompozíció sem. Tehát ebben a példában nem létezik függőségőrző BCNF dekompozíció.

- **Következmény: Tetszőleges (R,F) esetén nincs mindig függőségőrző, BCNF dekompozíció.**

- **Ok: Túl erős volt a BCNF, ezért majd gyengítjük a definíciót, és így kapjuk majd a 3. normálforma definícióját.**

- ***Van-e minden (R,F) esetén veszteségmentes BCNF dekompozíció?* (A válasz IGEN lesz.)**

Normálformák (BCNF)

- **Állítás:**

  Ha d=(R1,...,Rk) az (R,F) veszteségmentes dekompozíciója, d'=(S1,S2) az (R1,Π_{R1}(F)) veszteségmentes dekompozíciója, akkor

  d''=(S1,S2,R2,...,Rk) az (R,F) veszteségmentes dekompozíciója.

- **Következmény: veszteségmentes kéttévágások egymás utáni alkalmazásával veszteségmentes dekompozícióhoz jutunk.**

- **Állítás: Minden 2 attribútumos séma tetszőleges F-re nézve BCNF-ben van.**

- **Következmény: Ha a kettévágásokkal kétoszlopos sémához jutunk, akkor ez BCNF, így hozzávehetjük az eredményhez.**

Normálformák (BCNF)

Naiv algoritmus veszteségmentes BCNF dekompozíció előállítására:

Ha R BCNF-ben van, akkor megállunk,

egyébként

van olyan nem triviális $X \rightarrow A$, amely R-ben teljesül, de megsérti a BCNF-et, azaz X nem szuperkulcs.

Ekkor R helyett vegyük az (XA,R-A) dekompozíciót.

A kettévágásokat addig hajtjuk végre, amíg minden tag BCNF-ben nem lesz. ALGORITMUS VÉGE

- **$XA \neq R$, mert különben X szuperkulcs lenne.**
  Így mindkét tagban csökken az attribútumok száma.

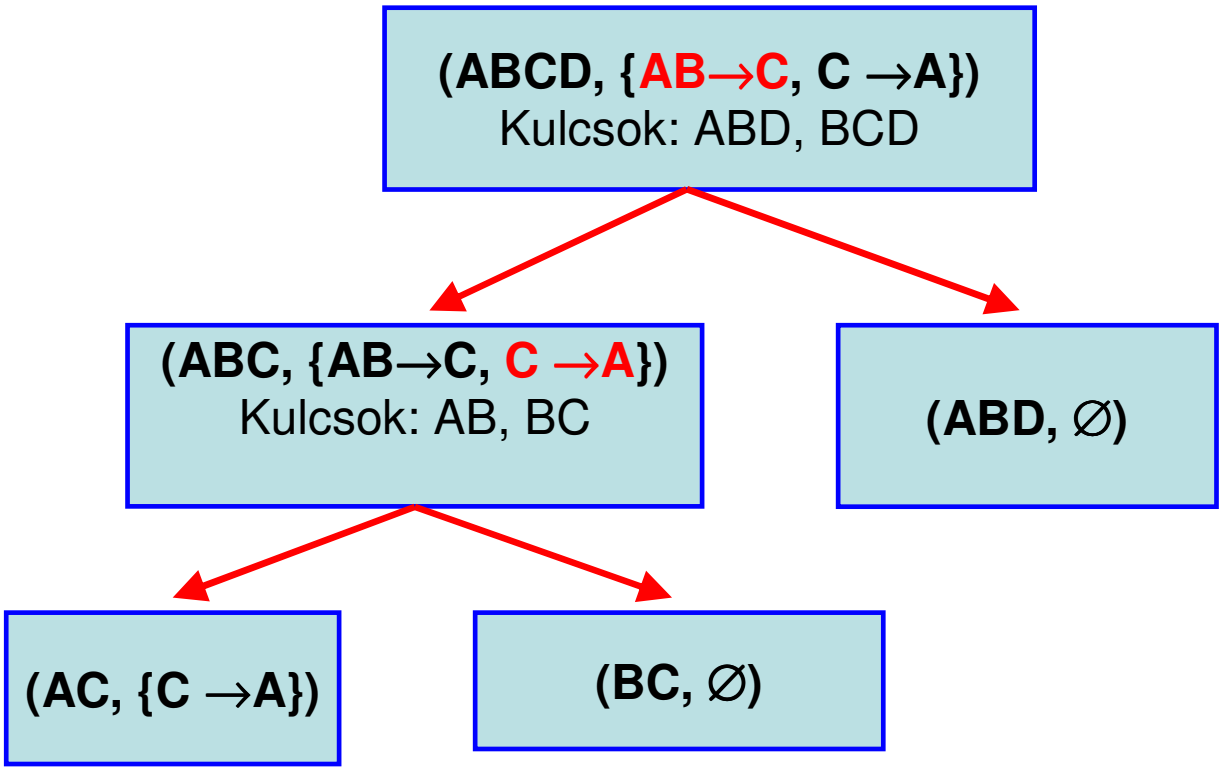
- **$XA \cap (R-A) = X \rightarrow A = XA-(R-A)$, azaz a kéttagú dekompozícióknál bizonyított állítás miatt *veszteségmentes kettévágást kaptunk*.**

- **Legrosszabb esetben a 2 oszlopos sémákig kell szétbontani.**

- **Tetszőleges (R,F) esetén veszteségmentes, BCNF dekompozíciót kapunk, de nem hatékonyan, mivel nem mondtuk meg, hogy lehet $X \rightarrow A$ függőséget találni, ha nincs BCNF-ben a séma.**

Normálformák (BCNF)

R=ABCD, F={AB→C, C →A}



Tehát d=(AC,BC,ABD) veszteségmentes BCNF dekompozíció.

(∅ azt jelenti, hogy csak a triviális függőségek teljesülnek a sémában.)

Normálformák (BCNF)

- **Előfordulhat, hogy függőségőrző is a kapott dekompozíció, de ezt le kell ellenőrizni.**

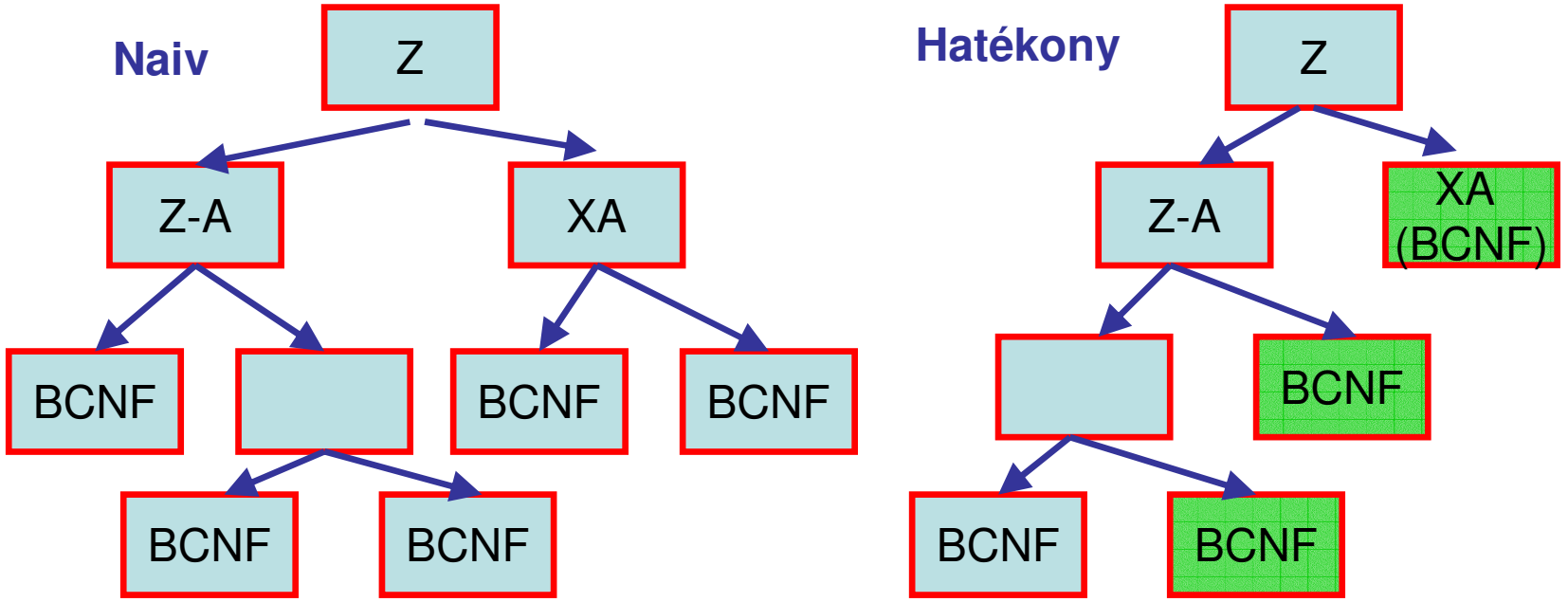
- **Nem egyértelmű a veszteségmentes BCNF dekompozíció (más sorrendben választva a függőségeket más kettévágásokat kaphatunk).**
- **Hogy lehet hatékonnyá tenni a módszert?**

- **Állítás: Ha (R,F) nincs BCNF-ben, akkor van olyan $A,B \in R$, amelyre $F \not\vdash (R-AB) \rightarrow A$.**

- **Következmény: Ha az összes A,B párt véve $F \not\vdash (R-AB) \rightarrow A$, akkor (R,F) BCNF-ben van. (Elégséges, de nem szükséges feltétel.)**
- **Az összes A,B pár esetén $A \notin (R-AB)^*$ vizsgálata polinomiális, mert ($O(|R|^2)$) párra a lezárást kell képezni ($O(|R|+|F|)$).**

Normálformák (BCNF)

- **A hatékony algoritmus a párok keresése, és a lezárások kiszámítása miatt polinomiális lépés számú.**
- **Szemben a naiv módszerrel a kettévágások során az egyik tag mindig BCNF-ben van.**



Normálformák (BCNF)

- **Hatékony algoritmus veszteségmentes BCNF dekompozíció előállítására:**
- **Input:(R,F), Output: d veszteségmentes BCNF dekompozíció**

- **Z:=R**
  repeat
    dekomponáljuk Z-t (Z-A,XA) részekre úgy, hogy XA BCNF a Π_{XA}(F)-re nézve, és F|—X→A.
    Legyen XA∈ d
    Z:=Z-A
  until Z-t nem lehet tovább dekomponálni
  Legyen Z∈ d

- **Ha Z-ben nincs olyan A,B, amelyre F|— (Z-AB)→A, akkor Z BCNF-ben van és tovább nem dekomponálható.**
  Különben legyen A,B olyan Z-beli attribútum, amelyre F|— (Z-AB)→A
    Y:=Z-B
    while van olyan A,B∈ Y, amelyre F|— (Y-AB)→A
      Y:=Y-B
    Return (Z-A,Y) /* A a while ciklusban utoljára beállított A attribútum, X:=Y-A */

Normálformák (BCNF)

- **Órarend adatbázis: Kurzus(K), Oktató(O), Időpont(I), Terem(T), Diák(D), Jegy(J)**

- **Feltételek:**

  Egy kurzust csak egy oktató tarthat: $K \rightarrow O$.

  Egy helyen, egy időben csak egy kurzus lehet: $IT \rightarrow K$.

  Egy időben egy tanár csak egy helyen lehet: $IO \rightarrow T$.

  Egy diák egy tárgyból csak egy jegyet kaphat: $KD \rightarrow J$.

  Egy időben egy diák csak egy helyen lehet: $ID \rightarrow T$.

- **R=KOITDJ F= {$K \rightarrow O$, $IT \rightarrow K$, $IO \rightarrow T$, $KD \rightarrow J$, $ID \rightarrow T$ }**

Normálformák (BCNF)

- **R=KOITDJ F= {K→O, IT→K, IO→T, KD→J, ID→T }**
- **Z:=KOITDJ**
- **AB:=KO, mert F |— ITDJ→K**

  Y:=KITDJ

  AB:=TK, mert F |— IDJ→T

  Y:=ITDJ

  AB:=TJ, mert F |— ID→T

  Y:=ITD tovább nem dekomponálható (BCNF)
- **Z:=KOITDJ-T=KOIDJ**
- **AB:=OI, mert F |— KDJ→O**

  Y:= KODJ

  AB:=OD, mert F |— KJ→O

  Y:= KOJ

  AB:=OJ, mert F |— K→O

  Y:=KO tovább nem dekomponálható (BCNF)
- **Z:=KOIDJ-O:=KIDJ**

  AB:=JI, mert F |— KD→J

  Y:=KDJ tovább nem dekomponálható (BCNF)
- **Z:=KIDJ-J=KID tovább nem dekomponálható (BCNF)**
- **Output: d=(ITD,KO,KDJ,KID) az (R,F) veszteségmentes BCNF dekompozíciója.**

Normálformák (3NF)

- **Adott (R,F) esetén $A \in R$ az R elsődleges attribútuma F-re nézve, ha A szerepel az R valamelyik F-re vonatkoztatott kulcsában.**
- **A 3NF több redundanciát enged meg, mint a BCNF.**
- **A következő két definíció ekvivalens.**
- **1. Definíció. R relációséma 3. normálformában (3NF-ben) van az F-re nézve, ha tetszőleges $XA \subseteq R$, $A \notin X$ és $F \vdash X \rightarrow A$ esetén $F \vdash X \rightarrow R$ (azaz X az R szuperkulcsa F-re nézve) vagy A az R elsődleges attribútuma F-re nézve.**

- **2. Definíció. R relációséma 3. normálformában (3NF-ben) van az F-re nézve, ha tetszőleges $XA \subseteq R$, $A \notin X$ és $X \rightarrow A \in F$ esetén $F \vdash X \rightarrow R$ (azaz X az R szuperkulcsa F-re nézve) vagy A az R elsődleges attribútuma F-re nézve.**

Normálformák (3NF)

- **Adott (R,F) és d=(R1,...,Rk) esetén d az R 3NF dekompozíciója, ha minden i-re Ri 3NF a Π_{Ri}(F)-re nézve.**

- **R=ABC, F={AB→C,C→A}**

  R kulcsai: AB, BC

  Láttuk, hogy nincs BCNF-ben, és nincs függőségőrző BCNF dekompozíciója.

- **R elsődleges attribútumai: A,B,C, így R 3NF-ben van F-re nézve, tehát nem kell tovább dekomponálni.**

- **Következmény: Mivel a definíciót gyengítettük, így ha (R,F) BCNF, akkor (R,F) 3NF is.**

Normálformák (3NF)

- Definíció: $F^* := \{X \rightarrow Y \mid F \vdash X \rightarrow Y\}$ F-ből levezethető összes függőség (F levezethetőség szerinti lezártja).

- Nézzük meg, hogy lehet minimálisra csökkenteni egy F függőségi halmazt.

- G az F minimális bázisa (másképpen minimális fedése), ha
  - $F^* = G^*$ (bázis, vagy másképpen fedés),
  - G minden függőségében a jobb oldalak egyeleműek, (jobb oldalak minimálisak)
  - G-ből nem hagyható el függőség, hogy F bázisa maradjon, (minimális halmaz)
  - G függőségeinek bal oldala nem csökkenthető, hogy F bázisa maradjon (bal oldalak minimálisak).

- Kérdések:
  - Minden F-nek létezik minimális bázisa?
  - Egyértelmű-e a minimális bázis?
  - Hogyan lehet adott F esetén meghatározni egy minimális bázist, lehetőleg hatékonyan?

Normálformák (3NF)

Mohó algoritmus minimális bázis előállítására:

1. **Jobb oldalak minimalizálása:**

   $X \to A1,...,Ak$ függőséget cseréljük le az

   $X \to A1, ... , X \to Ak$ k darab függőségre.

2. **A halmaz minimalizálása:**

   Hagyjuk el az olyan $X \to A$ függőségeket, amelyek a bázist nem befolyásolják, azaz

   while F változik

   if $(F-\{X \to A\})^* = F^*$ then $F := F-\{X \to A\}$;

3. **Bal oldalak minimalizálása:**

   Hagyjuk el a bal oldalakból azokat az attribútumokat, amelyek a bázist nem befolyásolják, azaz

   while F változik

   for all $X \to A \in F$

   for all $B \in X$

   if $((F-\{X \to A\}) \cup \{(X-\{B\}) \to A\})^* = F^*$ then $F := (F-\{X \to A\}) \cup \{X-\{B\}) \to A\}$

- Belátható, hogy a 3. lépés nem rontja el a halmaz minimalizálást, így minimális bázist kapunk.

Normálformák (3NF)

- **Az algoritmusban különböző sorrendben választva a függőségeket, illetve attribútumokat, különböző minimális bázist kaphatunk.**

- **F={A→B, B→A, B→C, A→C, C→A}**
  (F-{B→A})*=F*, mivel F-{B→A} |— B→A
  F:=F-{B→A}
  (F-{A→C})*=F*, mivel F-{A→C} |— A→C
  F:=F-{A→C}= {A→B,B→C,C→A} minimális bázis, mert több függőség és attribútum már nem hagyható el.

- **F={A→B, B→A, B→C, A→C, C→A}**
  (F-{B→C})*=F*, mivel F-{B→C} |— B→C
  F:=F-{B→C}={A→B,B→A,A→C,C→A} is minimális bázis, mert több függőség és attribútum már nem hagyható el.

Normálformák (3NF)

- **Az algoritmusban különböző sorrendben választva a függőségeket, illetve attribútumokat, különböző minimális bázist kaphatunk.**

- **F={AB→C, A→B, B→A}**

  (F-{AB→C}∪{A→C})*=F*, mivel

  (F-{AB→C})∪{A→C} ⊢ AB→C és F ⊢ A→C.

  F:=(F-{AB→C}∪{A→C})= {A→C,A→B,B→A} minimális bázis, mert több függőség és attribútum már nem hagyható el.

- **F={AB→C, A→B, B→A}**

  (F-{AB→C}∪{B→C})*=F*, mivel

  (F-{AB→C})∪{B→C} ⊢ AB→C és F ⊢ B→C.

  F:=(F-{AB→C}∪{B→C})= {B→C,A→B,B→A} is minimális bázis, mert több függőség és attribútum már nem hagyható el.

Normálformák (3NF)

- **Algoritmus függőségőrző 3NF dekompozíció előállítására:**
- **Input: (R,F)**
  - **Legyen G:={X→A,X→B,...,Y→C,Y→D,....} az F minimális bázisa.**
  - **Legyen S az R sémának G-ben nem szereplő attribútumai.**
  - **Ha van olyan függőség G-ben, amely R összes attribútumát tartalmazza, akkor legyen d:={R}, különben legyen**

    d:={S,XA, XB,...,YC,YD,...}.

Normálformák (3NF)

- **Algoritmus függőségőrző és veszteségmentes 3NF dekompozíció előállítására:**
- **Input: (R,F)**
  - **Legyen G:={X→A,X→B,...,Y→C,Y→D,....} az F minimális bázisa.**
  - **Legyen S az R sémának G-ben nem szereplő attribútumai.**
  - **Ha van olyan függőség G-ben, amely R összes attribútumát tartalmazza, akkor legyen d:={R}, különben legyen K az R egy kulcsa, és legyen**

d:={K,S,XA, XB,...,YC,YD,...}.

Normálformák (3NF)

- **Algoritmus függőségőrző és veszteségmentes 3NF redukált (kevesebb tagból álló) dekompozíció előállítására:**
- **Input: (R,F)**
  - **Legyen G:={X→A,X→B,...,Y→C,Y→D,....} az F minimális bázisa.**
  - **Legyen S az R sémának G-ben nem szereplő attribútumai.**
  - **Ha van olyan függőség G-ben, amely R összes attribútumát tartalmazza, akkor legyen d:={R}, különben legyen K az R egy kulcsa, és legyen**

    d:={K,S,XAB...,...,YCD...,...}.
  - **Ha K része valamelyik sémának, akkor K-t elhagyhatjuk.**

Normálformák (3NF)

- **Órarend adatbázis: Kurzus(K), Oktató(O), Időpont(I), Terem(T), Diák(D), Jegy(J)**

- **Feltételek:**
  Egy kurzust csak egy oktató tarthat: $K \rightarrow O$.
  Egy helyen, egy időben csak egy kurzus lehet: $IT \rightarrow K$.
  Egy időben egy tanár csak egy helyen lehet: $IO \rightarrow T$.
  Egy diák egy tárgyból csak egy jegyet kaphat: $KD \rightarrow J$.
  Egy időben egy diák csak egy helyen lehet: $ID \rightarrow T$.
- **$R=KOITDJ$ $F= \{K \rightarrow O, IT \rightarrow K, IO \rightarrow T, KD \rightarrow J, ID \rightarrow T\}$**

- **F-nek F, azaz saját maga minimális bázisa, ID kulcs, és $ID \subseteq IDT$, így**
  $d=\{KO,ITK,IOT,KDJ,IDT\}$ 3NF függőségőrző, veszteségmentes dekompozíció

Többértékű függőségek

- **Dolgozó adatbázis: Név(N), Diploma(D), Telefon(T)**
- **R=NDT**

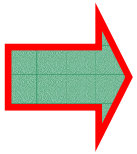
N	D	T
Kovács	{programozó, közgazdász}	{1234567, 7654321, 1212123}
Szabó	{programozó, jogász}	{1234123, 1234512}

0. normálforma: az értékek halmazok is lehetnek.

Többértékű függőségek

- **Átírás 1. normálfomára (az értékek atomi értékek)**

N	D	T
Kovács	{programozó, közgazdász}	{1234567, 7654321, 1212123}
Szabó	{programozó, jogász}	{1234123, 1234512}



N	D	T
Kovács	programozó	1234567
Kovács	programozó	7654321
Kovács	programozó	1212123
Kovács	közgazdász	1234567
Kovács	közgazdász	7654321
Kovács	közgazdász	1212123
Szabó	programozó	1234123
Szabó	programozó	1234512
Szabó	jogász	1234123
Szabó	jogász	1234512

- **Adott névhez diplomák halmaza és telefonszámok halmaza tartozik, egymástól függetlenül.**
- **Név →→ Diploma**
- **Név →→ Telefon**

30 értéket tárolunk (redundancia)!

Többértékű függőségek

- **Dekomponáljuk 2 táblára veszteségmentesen:**

N	D
Kovács	programozó
Kovács	közgazdász
Szabó	programozó
Szabó	jogász

N	T
Kovács	1234567
Kovács	7654321
Kovács	1212123
Szabó	1234123
Szabó	1234512

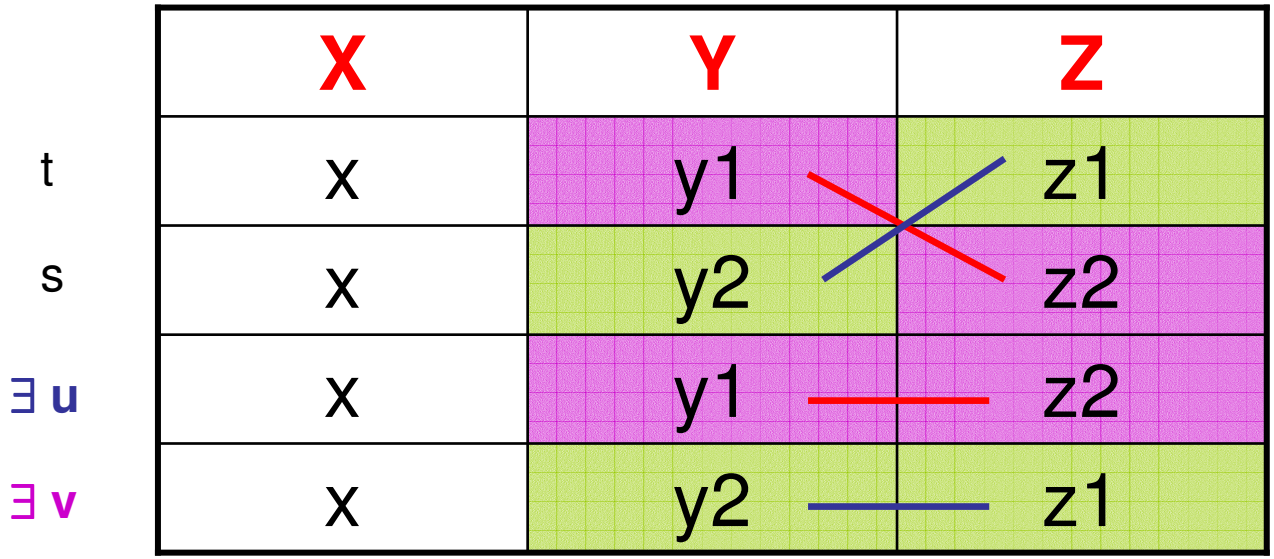
18 értéket tárolunk (csökkent a redundancia)

- **A 2 tábla összekapcsolása visszaadná az eredeti (redundáns) táblát, vagyis veszteségmentes lenne a dekompozíció.**

- **A funkcionális függőség speciális többértékű függőség, például Név → Telefon esetén 1 elemű halmaz (1 telefonszám) tartozik minden névhez, azaz Név →→ Telefon.**

Többértékű függőségek

- **Definíció: X,Y⊆R, Z:=R–XY esetén X→→Y többértékű függőség. (tf)**
- **A függőség akkor teljesül egy táblában, ha bizonyos mintájú sorok létezése garantálja más sorok létezését.**
- **A formális definiciót az alábbi ábra szemlélteti.**
- **Ha létezik t és s sor, akkor u és v soroknak is léteznie kell, ahol az azonos szimbólumok azonos értékeket jelölnek.**



Többértékű függőségek

Definíció (Formálisan): Egy R sémájú r reláció kielégíti az $X \rightarrow\rightarrow Y$ függőséget, ha $t,s \in r$ és $t[X]=s[X]$ esetén létezik olyan $u,v \in r$, amelyre $u[X]=v[X]=t[X]=s[X]$, $u[Y]=t[Y]$, $u[Z]=s[Z]$, $v[Y]=s[Y]$, $v[Z]=t[Z]$.

Állítás: Elég az u,v közül csak az egyik létezését megkövetelni.

	X	Y	Z
t	x	y1	z1
s	x	y2	z2
∃ u	x	y1	z2

Többértékű függőségek

- **Hasonló utat járunk be, mint a funkcionális függőségek esetén:**
  - **implikációs probléma**
  - **axiomatizálás**
  - **levezethető függőségek hatékony meghatározása (lezárás helyett a séma particiója (másképpen függőségi bázisa)**
  - **veszteségmentes dekompozíció**
  - **4. normálforma**
  - **veszteségmentes 4NF dekompozíció előállítása**

- **Mivel kijön majd, hogy minden 4NF egyben BCNF is, amire nincs egyszerre függőségőrző és veszteségmentes dekompozíció, így 4NF-re sincs mindig.**

Többértékű függőségek

- **Axiomatizálás**

Funkcionális függőségek	**Többértékű függőségek**	**Vegyes függőségek**
A1 (reflexivitás): $Y \subseteq X$ **esetén** $X \rightarrow Y$.	**A4 (komplementer):** $X \rightarrow\rightarrow Y$ **és** $Z=R-XY$ **esetén** $X \rightarrow\rightarrow Z$.	**A7 (funkcionálisból többértékű):** $X \rightarrow Y$ **esetén** $X \rightarrow\rightarrow Y$.
A2 (tranzitivitás): $X \rightarrow Y$ **és** $Y \rightarrow Z$ **esetén** $X \rightarrow Z$.	**A5 (tranzivitás):** $X \rightarrow\rightarrow Y$ **és** $Y \rightarrow\rightarrow S$ **esetén** $X \rightarrow\rightarrow S-Y$.	**A8 (többértékűből és funcionálisból funkcionális):** $X \rightarrow\rightarrow Y$ **és** $W \rightarrow S$, **ahol** $S \subseteq Y$, $W \cap Y = \varnothing$ **esetén** $X \rightarrow S$.
A3 (bővíthetőség): $X \rightarrow Y$ **és tetszőleges** Z **esetén** $XZ \rightarrow YZ$.	**A6 (bővíthetőség):** $X \rightarrow\rightarrow Y$ **és tetszőleges** $V \subseteq W$ **esetén** $XW \rightarrow\rightarrow YV$.	

Többértékű függőségek

- **Jelölés a továbbiakban:**
  - **F funkcionális függőségek halmaza**
  - **M többértékű függőségek halmaza**
  - **D vegyes függőségek (funkcionális és többértékű függőségek) halmaza**

- **Tétel (helyes és teljes axiómarendszerek):**
  - **A1,A2,A3 helyes és teljes a funkcionális függőségekre,**
  - **A4,A5,A6 helyes és teljes a többértékű függőségekre,**
  - **A1,A2,A3,A4,A5,A6,A7,A8 helyes és teljes a vegyes függőségekre.**

Többértékű függőségek

- **Állítás (további levezetési szabályok):**
  1. $X \rightarrow\rightarrow Y$ és $X \rightarrow\rightarrow V$ esetén $X \rightarrow\rightarrow YV$.
  2. $X \rightarrow\rightarrow Y$ és $WX \rightarrow\rightarrow V$ esetén $WX \rightarrow\rightarrow V\text{-}WY$.
  3. $X \rightarrow\rightarrow Y$ és $XY \rightarrow V$ esetén $X \rightarrow V\text{-}Y$.
  4. $X \rightarrow\rightarrow Y$ és $X \rightarrow\rightarrow V$ esetén $X \rightarrow\rightarrow Y \cap V$
     és $X \rightarrow\rightarrow V\text{-}Y$
     és $X \rightarrow\rightarrow Y\text{-}V$.

Többértékű függőségek

- **Állítás: X→→Y-ből nem következik, hogy X→→A, ha A∈ Y. (A jobb oldalak nem szedhetők szét!)**
- **Bizonyítás: A következő r tábla kielégíti az X→→AB-t, de nem elégíti ki az X→→A-t. q.e.d.**

X	A	B	C
x	a	b	c
x	e	f	g
x	a	b	g
x	e	f	c

X→→A esetén ennek a sornak is benne kellene lenni a táblában.

x	a	f	g

Többértékű függőségek

- **Állítás: X→→Y és Y→→V-ből nem következik, hogy X→→V. (A szokásos tranzitivitás nem igaz általában!)**
- **Bizonyítás: A következő r tábla kielégíti az X→→AB-t, AB→→BC-t, de nem elégíti ki az X→→BC-t. q.e.d.**

X	A	B	C
x	a	b	c
x	e	f	g
x	a	b	g
x	e	f	c

X→→BC esetén ennek a sornak is benne kellene lenni a táblában.

x	e	b	c

Többértékű függőségek

- A veszteségmentesség, függőségőrzés definíciójában most F funkcionális függőségi halmaz helyett D függőségi halmaz többértékű függőségeket is tartalmazhat.

- Így például d=(R1,...,Rk) veszteségmentes dekompozíciója R-nek D-re nézve, akkor és csak akkor, ha minden D-t kielégítő r tábla esetén $r=\Pi_{R1}(r)|><|...|><|\ \Pi_{Rk}(r)$

- A következő tétel miatt a veszteségmentesség implikációs problémára vezethető vissza, így hatékonyan eldönthető.

- Tétel: A d=(R1,R2) akkor és csak akkor veszteségmentes dekompozíciója R-nek, ha
  $D \models R1 \cap R2 \rightarrow\rightarrow R1-R2$.

Többértékű függőségek

- A 4.normálforma definiálása előtt foglaljuk össze, hogy melyek a triviális többértékű függőségek, vagyis amelyek minden relációban teljesülnek.

- Mivel minden funkcionális függőség többértékű függőség is, így a triviális funkcionális egyben triviális többértékű függőség is.

1. $Y \subseteq X$ esetén $X \rightarrow\rightarrow Y$ triviális többértékű függőség.

- Speciálisan $Y=\emptyset$ választással $X \rightarrow\rightarrow \emptyset$ függőséget kapjuk, és alkalmazzuk a komplementer szabályt, azaz $Z=R-X\emptyset$, így az $X \rightarrow\rightarrow R-X$ függőség is mindig teljesül, azaz:

2. $XY=R$ esetén $X \rightarrow\rightarrow Y$ triviális többértékű függőség.

- A szuperkulcs, kulcs definíciója változatlan, azaz X szuperkulcsa R-nek D-re nézve, ha $D \vdash X \rightarrow R$.
- A minimális szuperkulcsot kulcsnak hívjuk.

Többértékű függőségek

- A 4.normálforma hasonlít a BCNF-re, azaz minden nem triviális többértékű függőség bal oldala szuperkulcs.

- **Definíció:** R **4NF**-ben van D-re nézve, ha $XY \neq R$, $Y \not\subset X$, és $D \vdash X \rightarrow\rightarrow Y$ esetén $D \vdash X \rightarrow R$.

- **Definíció:** d={R1,...,Rk} dekompozíció **4NF**-ben van D-re nézve, ha minden Ri **4NF**-ben van $\Pi_{Ri}(D)$-re nézve.

- **Állítás:** Ha R **4NF**-ben van, akkor **BCNF**-ben is van.
- Bizonyítás. Vegyünk egy nem triviális $D \vdash X \rightarrow A$ **funkcionális** függőséget. Ha $XA=R$, akkor $D \vdash X \rightarrow R$, ha $XA \neq R$, akkor a $D \vdash X \rightarrow\rightarrow A$ nem triviális többértékű függőség és a **4NF** miatt $D \vdash X \rightarrow R$. q.e.d.

- **Következmény:** Nincs mindig **függőségőrző** és **veszteségmentes 4NF** dekompozíció.

Többértékű függőségek

- **Veszteségmentes 4NF dekompozíciót mindig tudunk készíteni a naiv BCNF dekomponáló algoritmushoz hasonlóan.**

- **Naiv algoritmus veszteségmentes 4NF dekompozíció előállítására:**

  Ha R 4NF-ben van, akkor megállunk,

  egyébként

  van olyan nem triviális $X \rightarrow\rightarrow Y$, amely R-ben teljesül, de megsérti a 4NF-et, azaz X nem szuperkulcs.

  Ekkor R helyett vegyük az (XY,R-Y) dekompozíciót.

  A kettévágásokat addig hajtjuk végre, amíg minden tag 4NF-ben nem lesz. ALGORITMUS VÉGE.

- **Az is feltehető, hogy X és Y diszjunkt, mert különben Y helyett az Y-X-et vehettük volna jobb oldalnak.**

- **$XY \neq R$, így *mindkét tagban csökken az attribútumok száma.***

- **$XY \cap (R-Y) = X \rightarrow\rightarrow Y = XY-(R-Y)$, azaz a kéttagú dekompozícióknál bizonyított állítás miatt *veszteségmentes kettévágást kaptunk*.**

- **Legrosszabb esetben a 2 oszlopos sémákig kell szétbontani, amelyek mindig 4NF-ben vannak, mivel nem lehet bennük nem triviális többértékű függőség.**