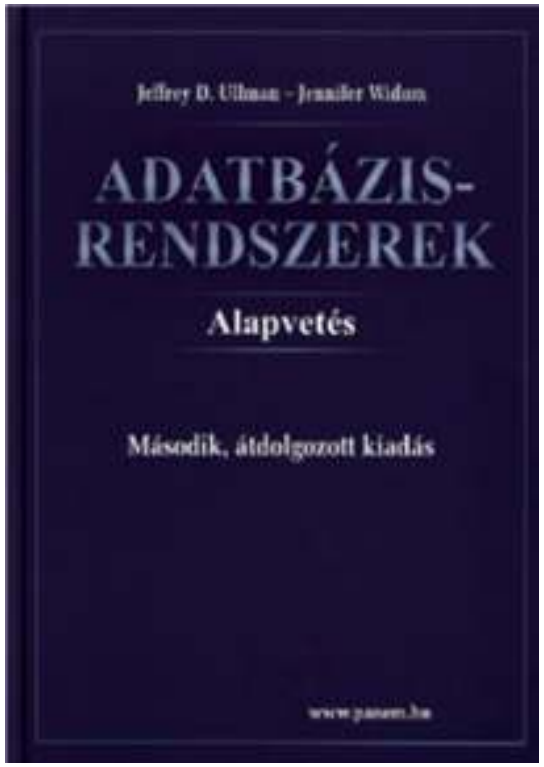
Tervezés: Egyed-kapcsolat modell és az SQL DDL: táblák, nézetek

Tankönyv: Ullman-Widom:
Adatbázisrendszerek Alapvetés
Második, átdolgozott kiad, 2009



4.1.- 4.4. E/K-modell elemei (folyt)

4.5.- 4.6. E/K-diagram átírása
relációs modellé (folyt/bef)

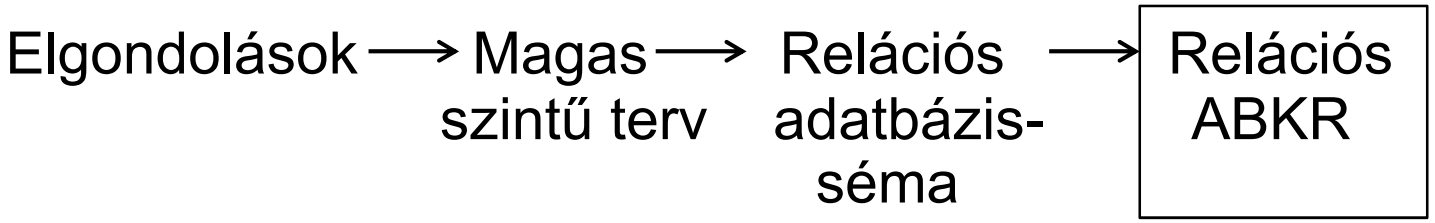
7.fej. Megvalósítás - SQL DDL:
Táblák, megszorítások és triggerek

8.fej. Nézettáblák létrehozása és használata

6.3.5. Alkérdések a FROM záradékban (inline nézet)

Magas szintű adatbázismodellek

- Vizsgáljuk meg azt a folyamatot, amikor egy új adatbázist létrehozunk, vegyük példaként a sörivós adatbázist.
- Az adatbázis-modellezés és implementálás eljárása



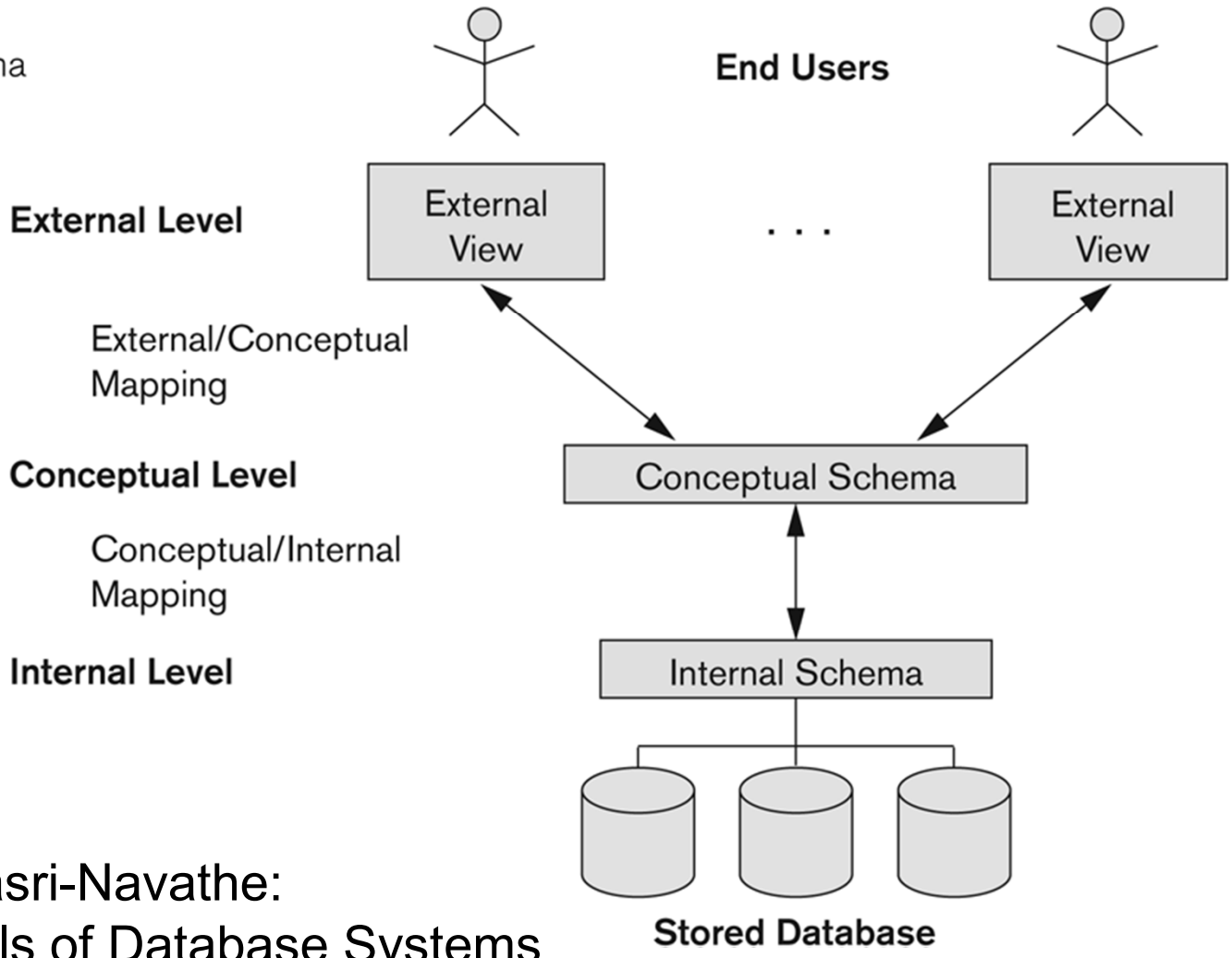
- Modellezés
  - komplex valós világ leképezése, absztrakció
- Tervezési fázis:
  - Milyen információkat kell tárolni?
  - Mely információelemek kapcsolódnak egymáshoz?
  - Milyen megszorításokat kell figyelembe venni? stb…

Az adatmodellek 3 szintje

- Hogyan látjuk az adatbázist?
- A 3 szintű ANSI/SPARC architektúra
  - Logikai (külső, a felhasználói szemléletnek megfelelő szinten, nézetek)
  - Fogalmi (conceptual) (absztrakt, szintetizálja az összes felhasználói szemléletet)
  - Fizikai (belső, az adatbázis valamilyen fizikai adatstruktúrában letárolva a háttértárolón)

Az adatmodellek 3 szintje

Figure 2.2
The three-schema architecture.



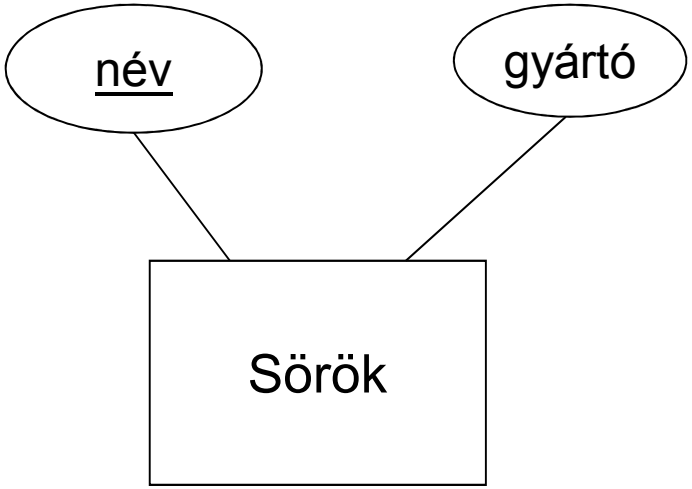
Forrás: Elmasri-Navathe: Fundamentals of Database Systems

E/K elemei (ismétlés 2.előadás)

- Egyed-kapcsolat modell: E/K modell
  (Entity-relationship ER) alapfogalmak:
- Egyedhalmazok (absztrakt objektumok osztálya)
  - Miről gyűjtünk adatokat? (egyedhalmaz, osztály)
  - Mit tegyünk egy gyűjteménybe? - hasonlóság
  - Hasonló egyedek összessége (egyed előfordulás)
- Attribútumok
  - Megfigyelhető tulajdonságok, megfigyelt értékek
  - Az egyedek tulajdonságait írják le
- Kapcsolatok
  - Más egyedhalmazokkal való kapcsolatuk

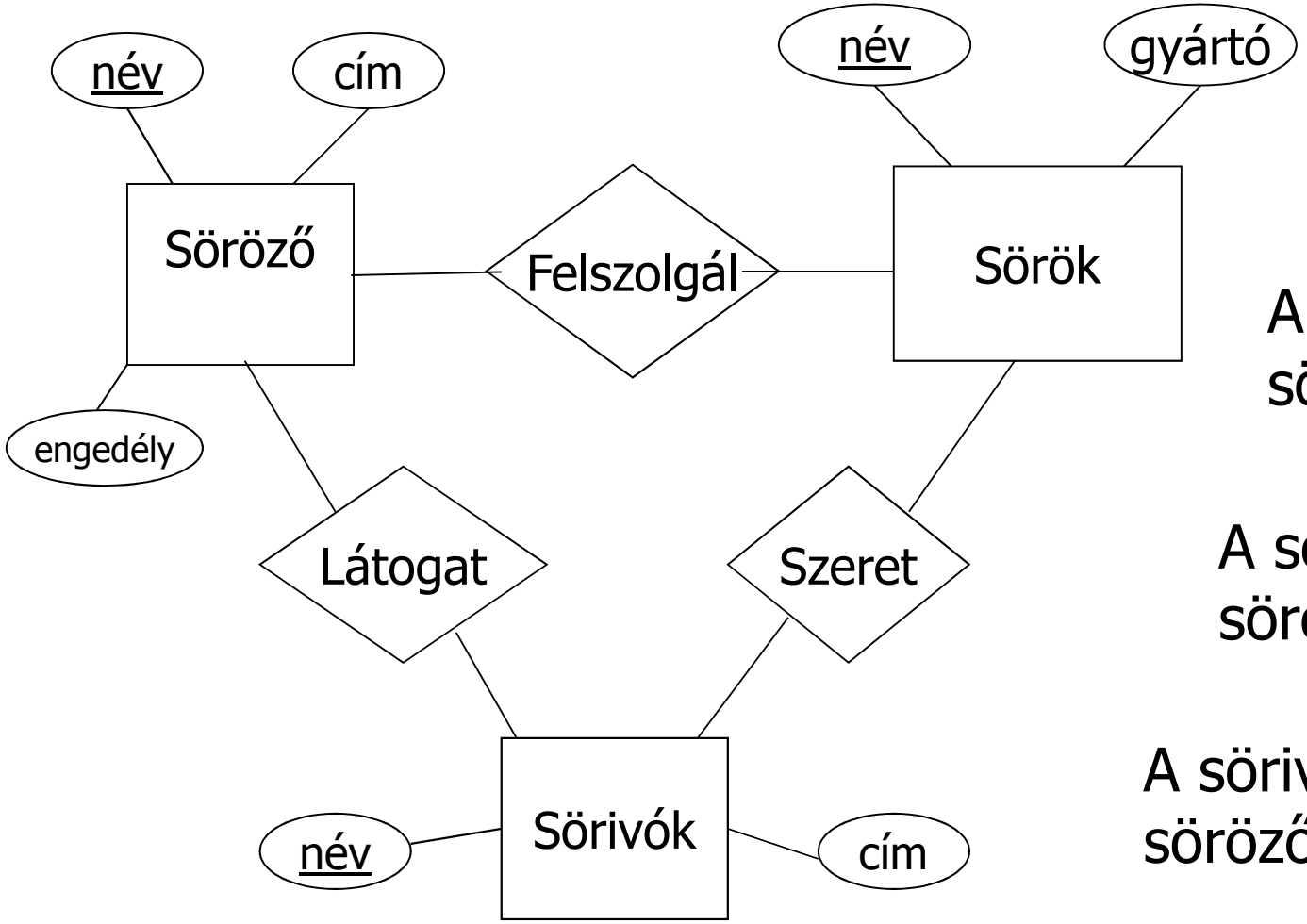
E/K-diagram: Egyedhalmazok

- E/K diagram: séma-szinten grafikusan ábrázoljuk
- Egyedhalmazok: téglalap
- Tulajdonságok: ovális
  - az elsődleges kulcshoz tartozó tulajdonságokat aláhúzzuk.



E/K-diagram: Kapcsolatok

- A kapcsolatok jele: rombusz



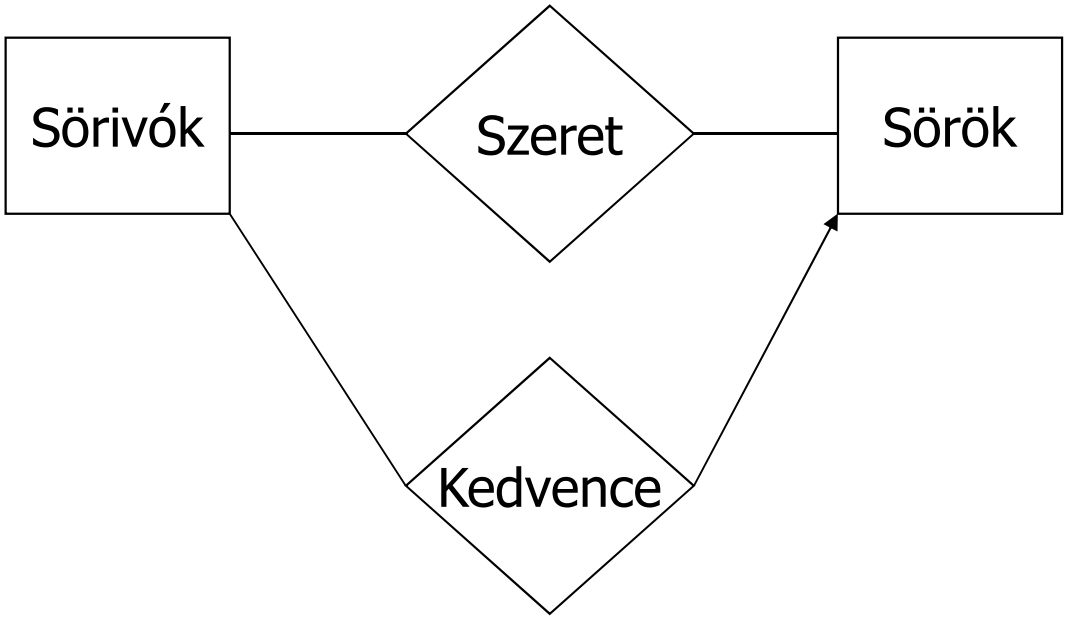
A sörözőkben söröket árulnak.

A sörivók söröket szeretnek.

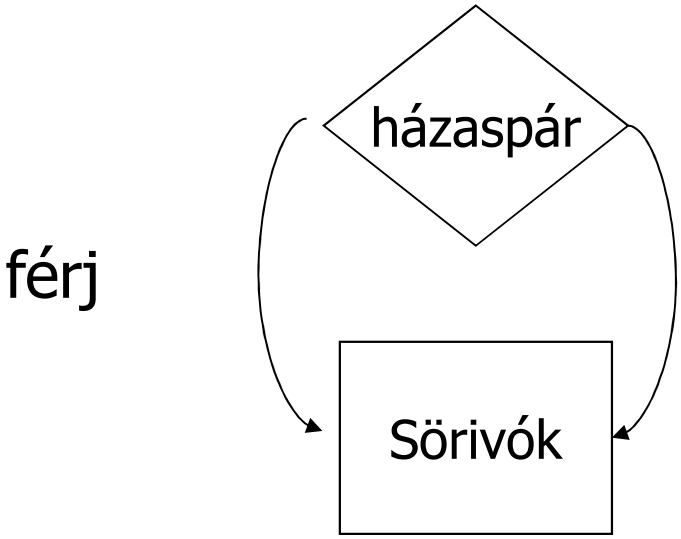
A sörivók sörözőkbe járnak.

Bináris (két egyedhalmaz közötti) kapcsolatok típusai (sok-egy, sok-sok)

(két egyedhalmaz között több kapcsolat is lehet)



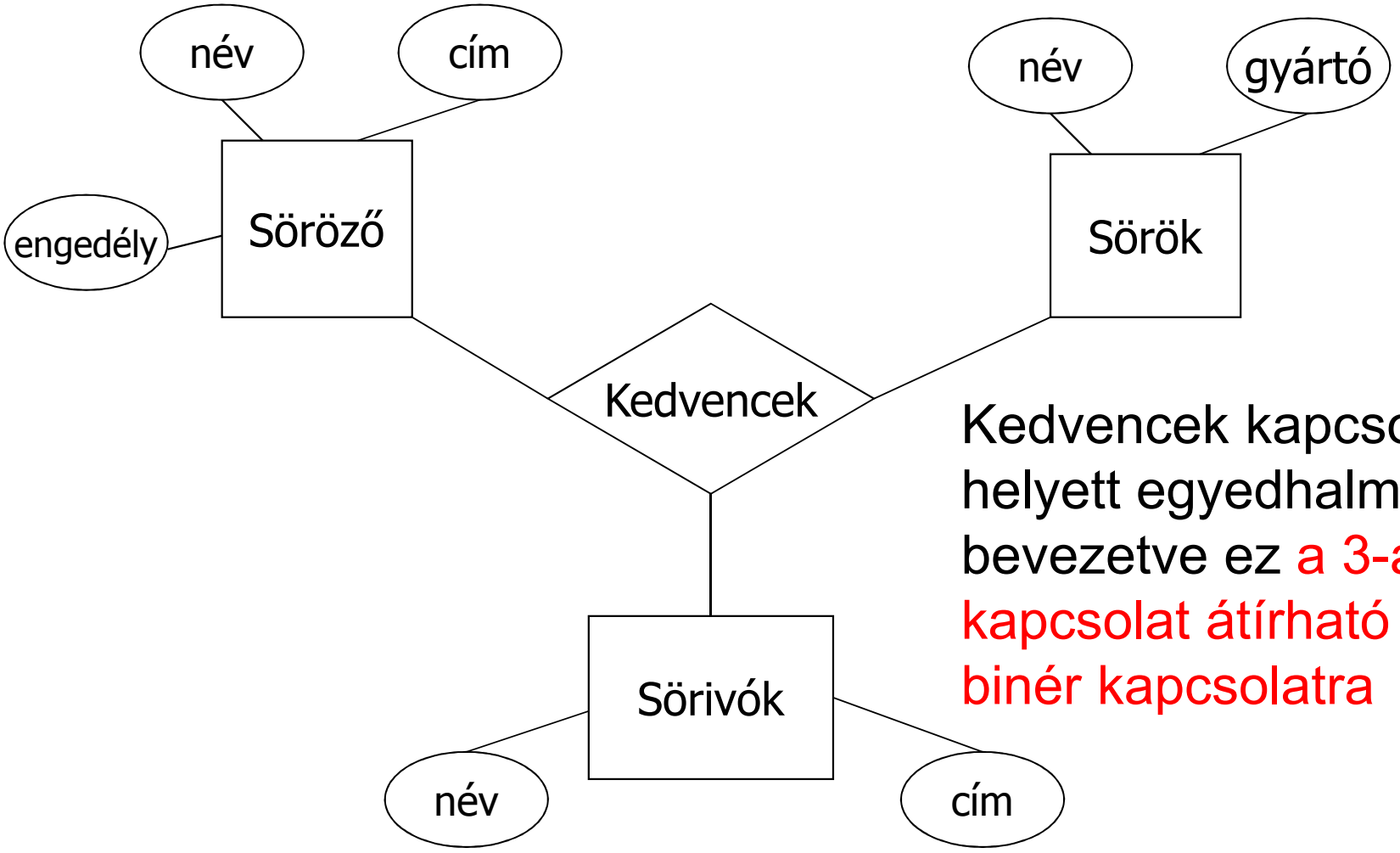
Egy egyedhalmaz önmagával is kapcsolódhat: Szerepek (Roles)



A kapcsolat előfordulása

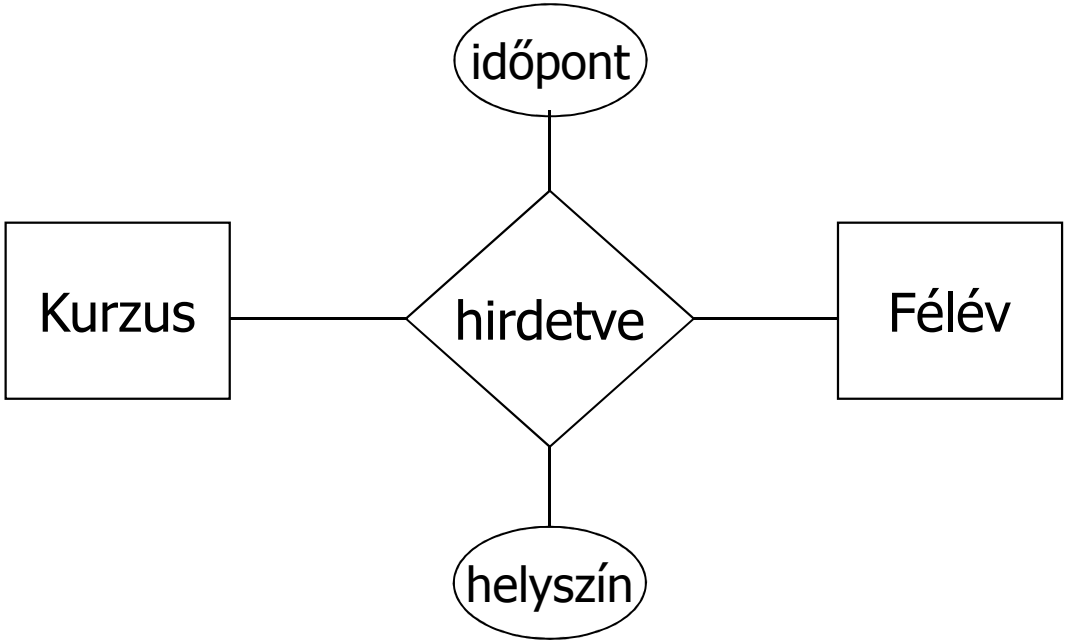
Férj	Feleség
Bob	Ann
Joe	Sue
…	…

Többágú (példa: 3-ágú) kapcsolat



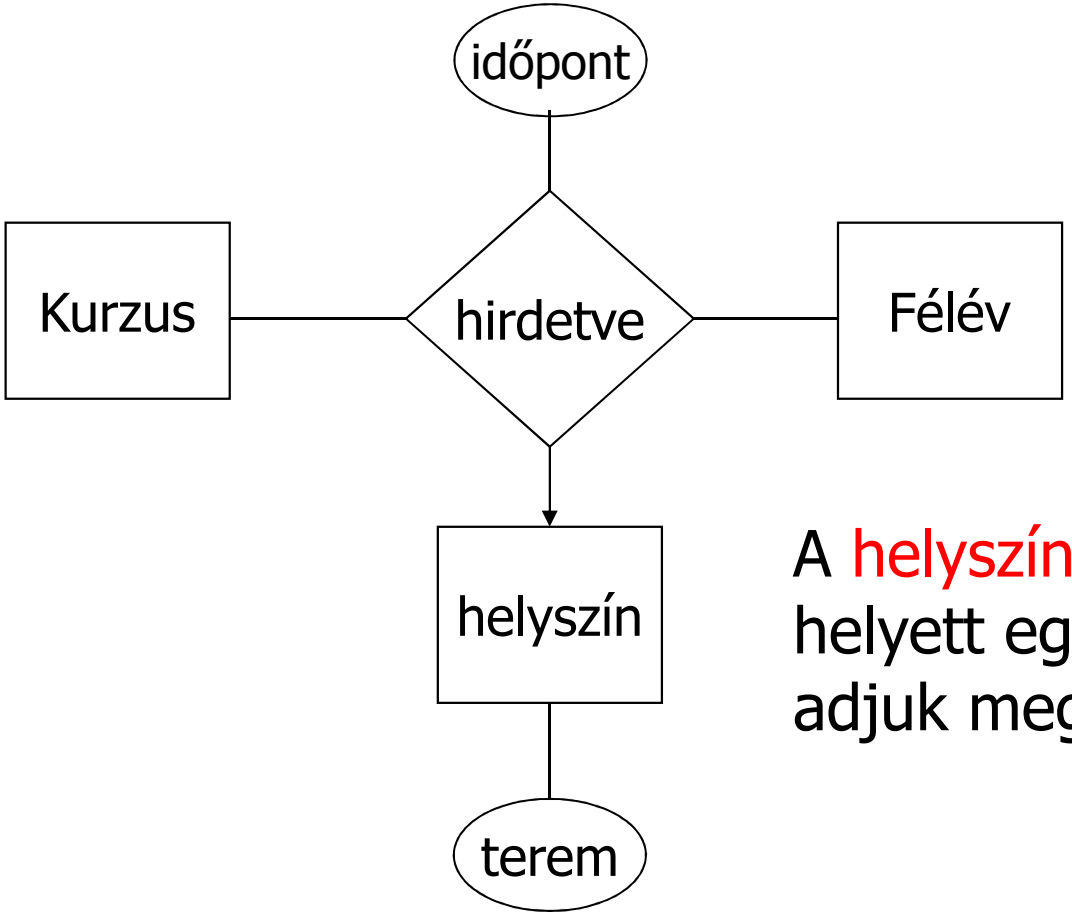
Kedvencek kapcsolat helyett egyedhalmazt bevezetve ez a 3-ágú kapcsolat átírható 3db binér kapcsolatra

Kapcsolatnak is lehet attribútuma



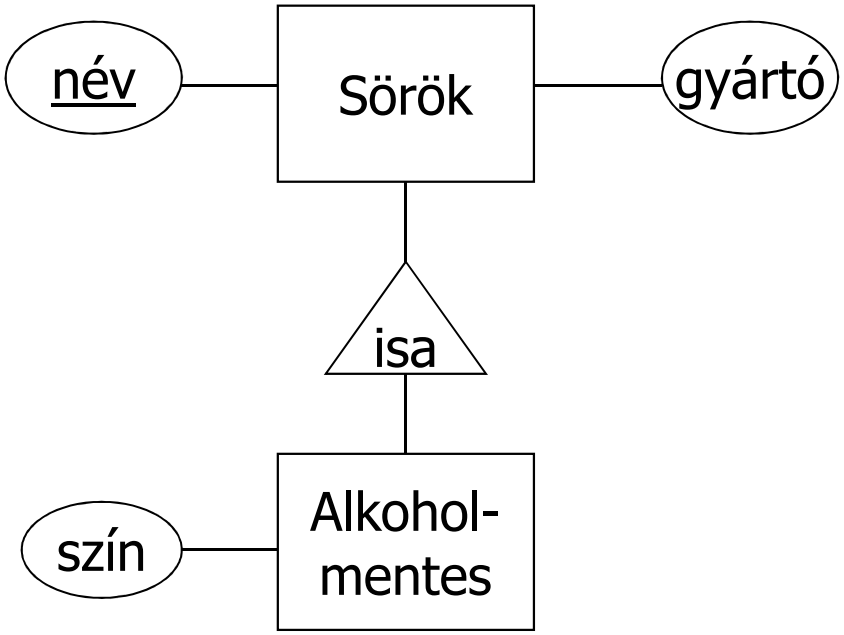
Az időpont és helyszín a Kurzus és Félév együttes függvénye, de egyiké sem külön.

Tervezési kérdés: Attribútum vagy egyedhalmaz?

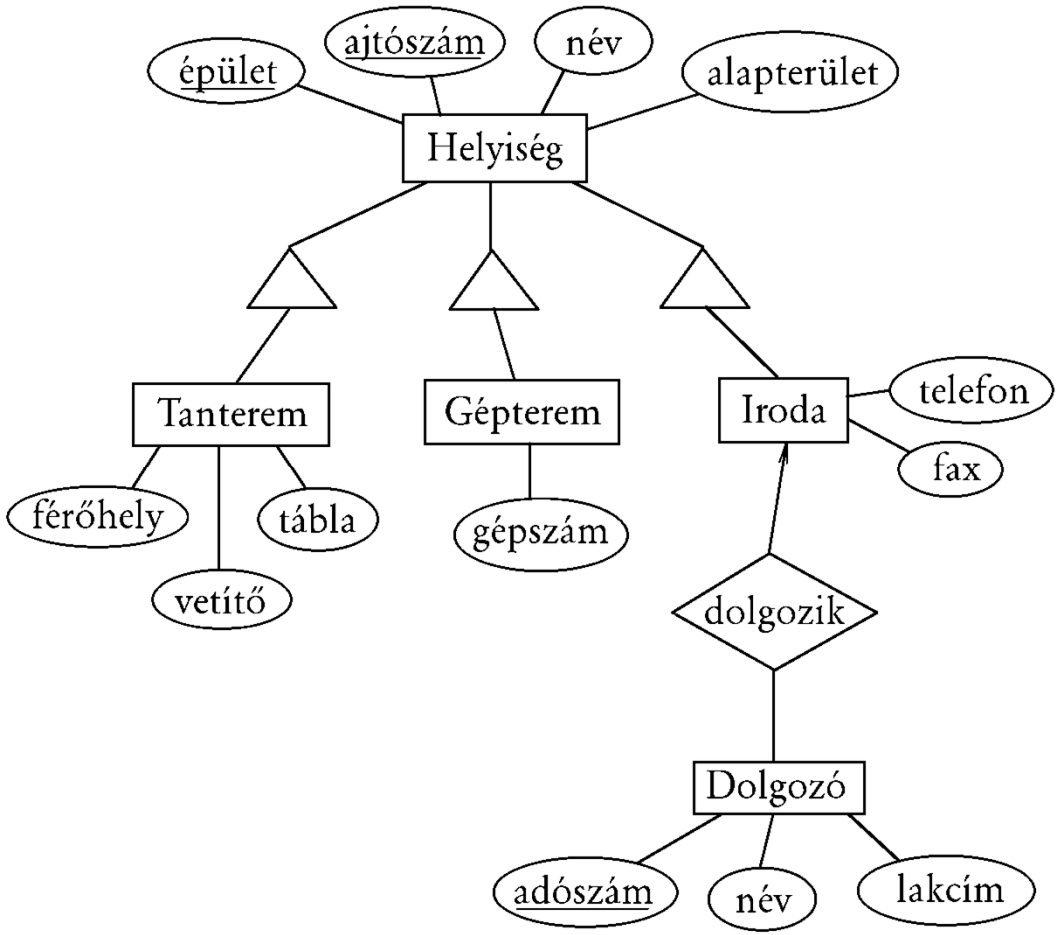


A helyszínt itt attribútum helyett egyedhalmazként adjuk meg

Új anyag: Alosztályok és öröklődés
Speciális „is-a” (az-egy) kapcsolat



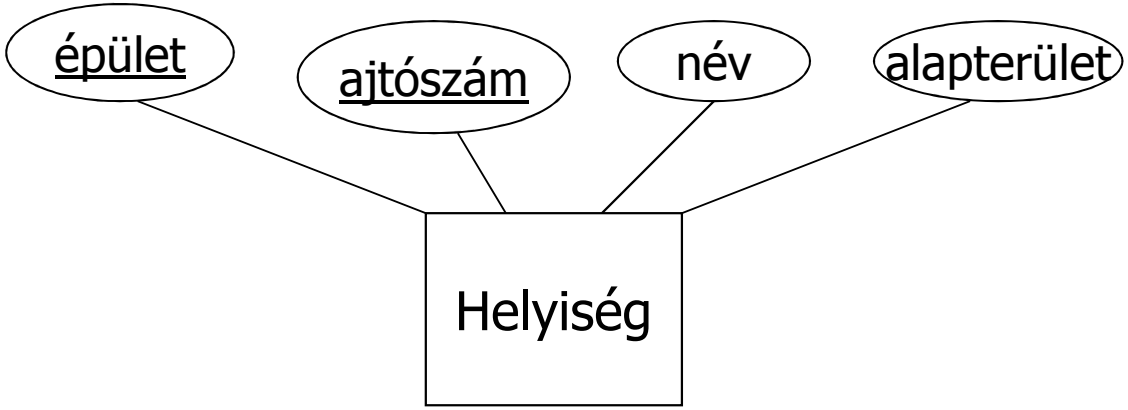
Példa: „is-a" (az-egy) kapcsolatra



Kulcs megszorítás jele: aláhúzás

Példa egyszerű kulcsra: <u>név</u> a Sörök elsődleges kulcsa:

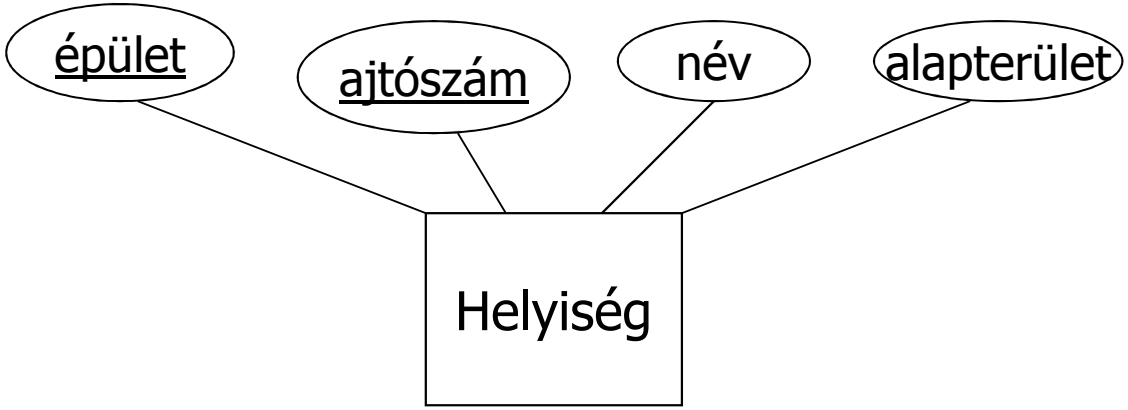
Példa összetett kulcsra: <u>épület</u>, <u>ajtószám</u> két-attribútumos elsődleges kulcsa a Helyiség-nek:



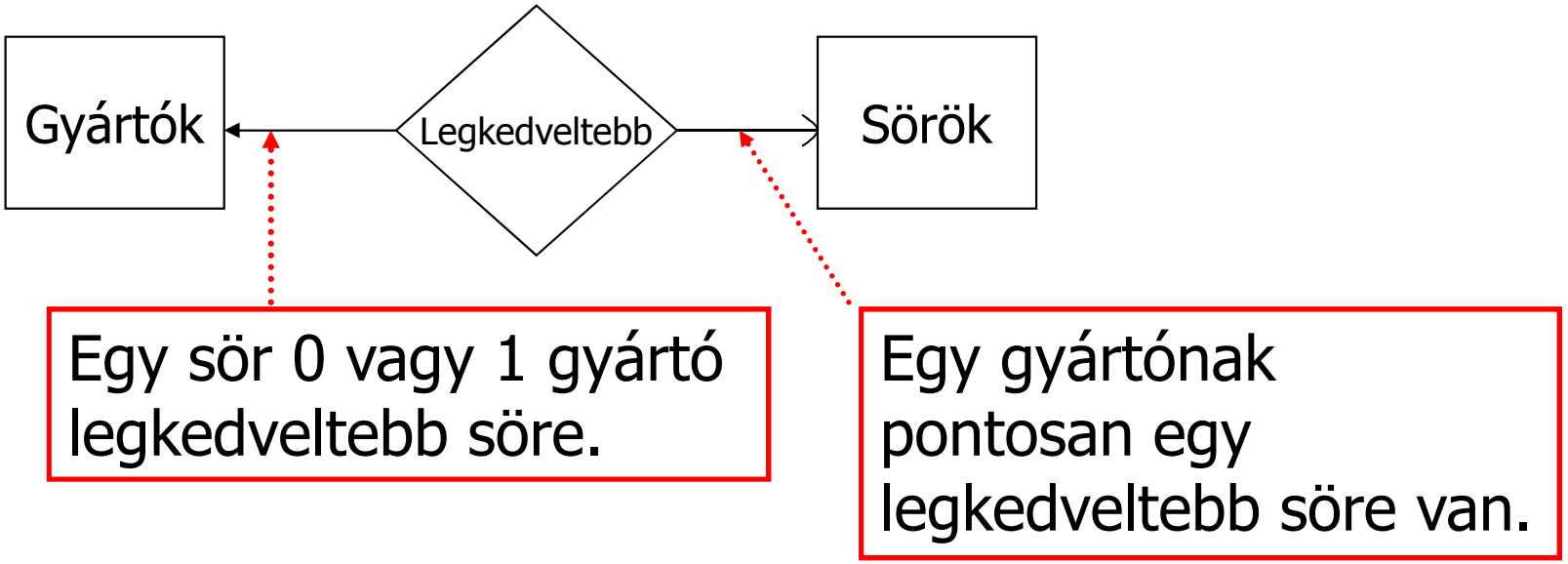
Kulcs megszorítás jele: aláhúzás

Példa egyszerű kulcsra: <u>név</u> a Sörök elsődleges kulcsa:

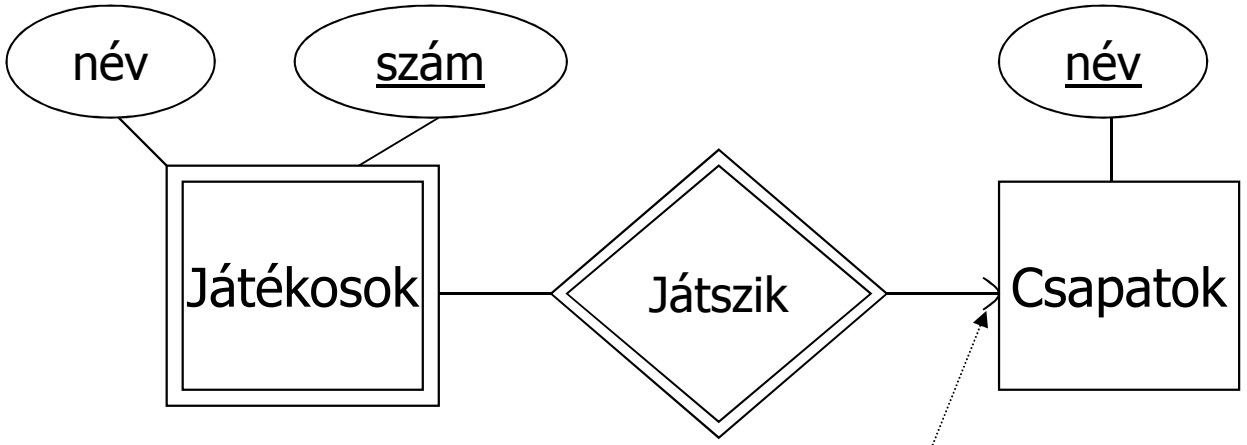
Példa összetett kulcsra: <u>épület</u>, <u>ajtószám</u> két-attribútumos elsődleges kulcsa a Helyiség-nek:



Hivatkozási épség megszorítás jele a kerek végződés —)



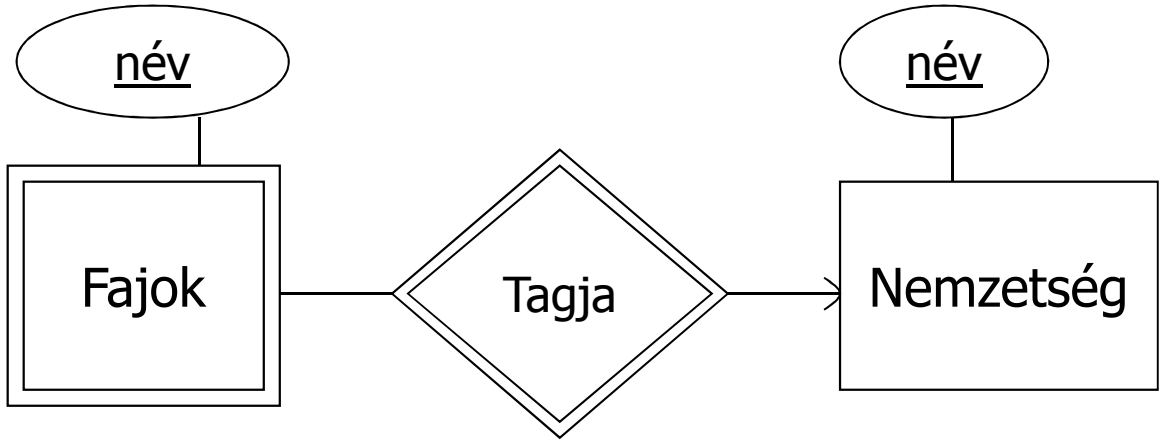
Erős és gyenge egyedhalmaz



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

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

Erős és gyenge egyedhalmaz



Tankönyv 4.21. példája:
például az emberek a Homo sapiens fajhoz tartoznak, ahol Homo a nemzetség neve, a sapiens a faj neve (sajnos maguk a fajok nevei nem egyértelműek, két vagy több nemzettségben is lehet ugyanolyan fajnév).

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 egyedhalmaz:
  - például a vizsgajegy osztály helyett jegy tulajdonságot használjunk.

Modellezési feladatok (Tankönyv)

- **4.1.1. feladat.** Tervezzünk egy bank részére adatbázist, amely tartalmazza az ügyfeleket és azok számláit. Az ügyfelekről tartsuk nyilván a nevüket, címüket, telefonszámukat és TAJ-számukat. A számláknak legyen számlaszámuk, típusuk (pl. takarékbetét-számla, folyószámla stb.) és egyenlegük. Továbbá, meg kell jelölni azokat az ügyfeleket, akiknek van számlájuk. Adjuk meg az E/K diagramját ennek az adatbázisnak. Alkalmazzunk nyilakat a kapcsolatokban a multiplicitások jelölésére.

Modellezési feladatok (Tankönyv)

- **4.1.3. feladat.** Adjuk meg az E/K modelljét egy olyan adatbázisnak, amely csapatokat, játékosokat és azok szurkolóit tartja nyilván:
  - Minden csapatról tároljuk a nevét, játékosait, csapatkapitányát (ő is egy játékos), mezük színét.
  - Minden játékosnak legyen neve.
  - Minden rajongóról tartsuk nyilván a nevét, kedvenc csapatát, kedvenc játékosát és kedvenc színét.
- Vigyázzunk, a színek halmaza nem lehet a csapatok egy attribútumának típusa. Hogyan lehet ezzel a megszorítással együtt megfelelő modellt készíteni?

Modellezési feladatok (Tankönyv)

- **4.1.9. feladat.** Tervezzünk adatbázist egy tanulmányi osztály számára. Ez az adatbázis tartalmazza a hallgatókat, oktatókat, tanszékeket és kurzusokat. Ezenkívül tartsuk nyilván, hogy a hallgatók milyen kurzusokat vettek fel, az adott kurzust mely oktató oktatja, a hallgatók jegyeit, a kurzusoknál az oktató munkáját segítő hallgatókat, egy adott kurzust mely tanszék ajánlotta, és minden olyan információt, ami a fentiek megvalósításához szükséges. Megjegyezzük, hogy ez a feladat nagy szabadságot enged a korábbiakhoz képest. Dönteni kell a kapcsolatok típusáról (sok-sok, sok-egy vagy egy-egy), az alkalmas típus megválasztásról, illetve arról, hogy milyen segédinformációkat használunk.

E/K-diagram átírása

Tankönyv 4.5.-4.6. E/K-diagram átírása relációkká

- Egyedhalmazok átírása relációkká
- E/K-kapcsolatok átírása relációkká
- Egyszerűsítés, összevonások

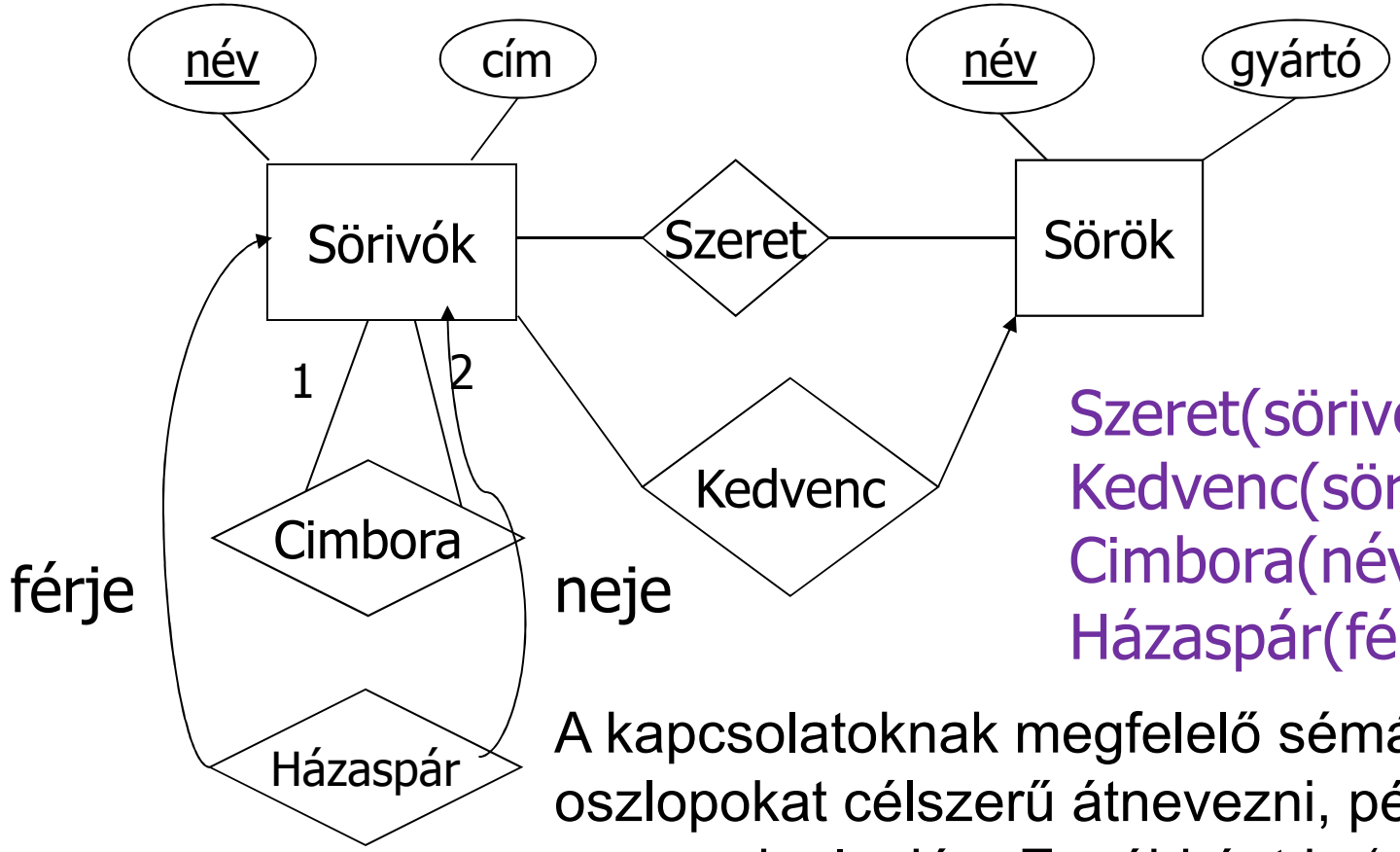
- Gyenge egyedhalmazok kezelése
- Osztályhierarchia átalakítása relációkká

E/K diagram átírása relációs adatbázistervre

Mi minek felel meg:

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

Példa: E/K diagram átírása relációkká
Az egyedek átírása után a kapcsolatok átírása:



Szeret(sörivó, sör)
Kedvenc(sörivó, 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.

Relációk összevonása

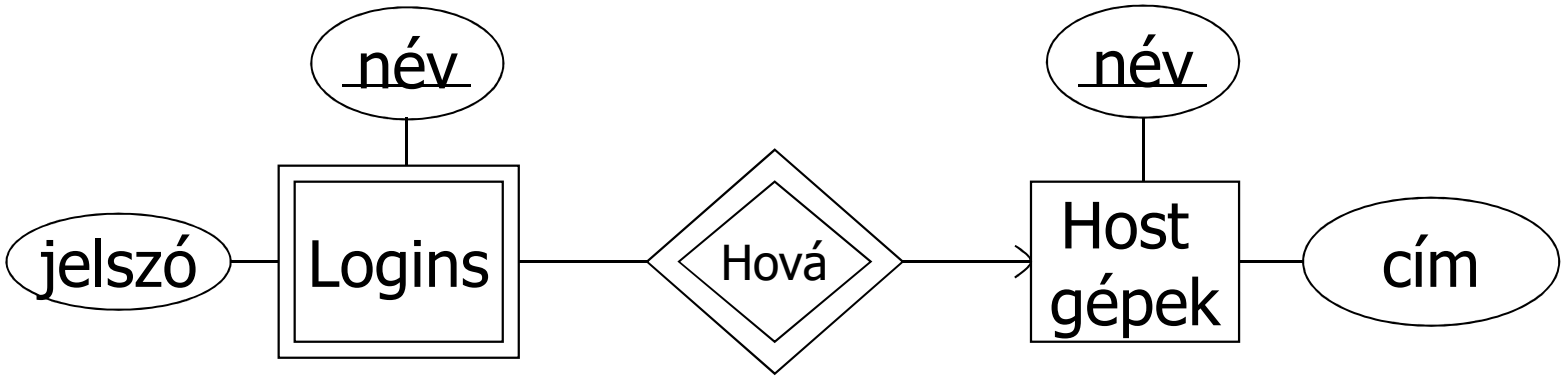
- Összevonhatunk 2 relációt, ha

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

- Példa:

  Sörivók(név, cím) és Kedvenc(ivó,sör) összevonható, és kapjuk az Sörivó1(név,cím,kedvencSöre) sémát.

Gyenge egyedhalmaz átírása



Hostgépek(hostNév, cím)

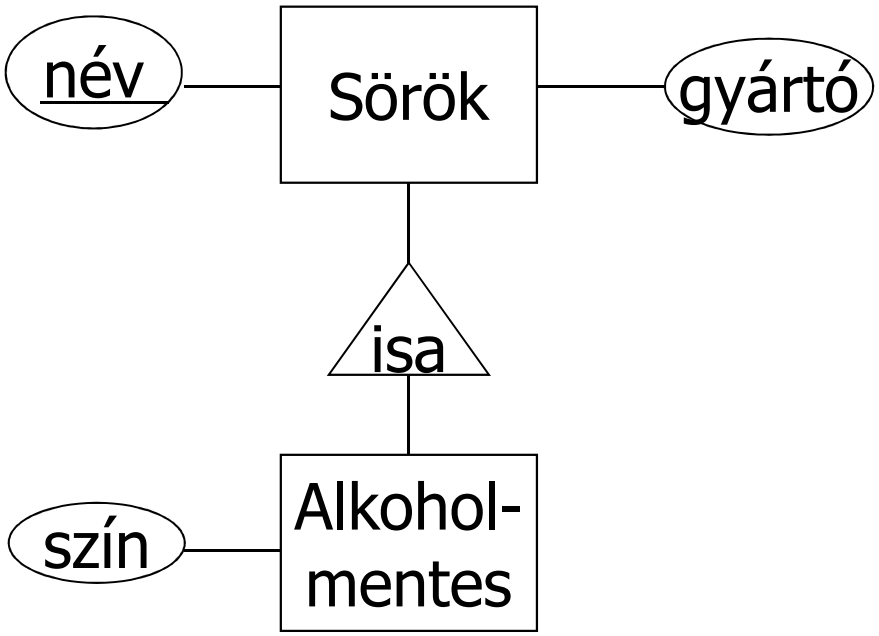
Logins(loginNév, hostNév, jelszó)

~~Hová(loginNév, hostNév, hostNév2)~~

Beolvasztjuk a Logins relációba

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

Alosztály átírására relációkká



Alosztályok átírása: három megközelítés

- E/R stílusban: Egy reláció minden alosztályra, de az általános osztályból csak a kulcsokat vesszük hozzá a saját attribútumokhoz.
- Objektumorientált stílusban: Egy reláció minden alosztályra, felsorolva az összes tulajdonságot, beleértve az örökölteket is.
- 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.

E/K tipusú átalakítás ---1

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.

Objektumorientált megközelítés ---2

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.

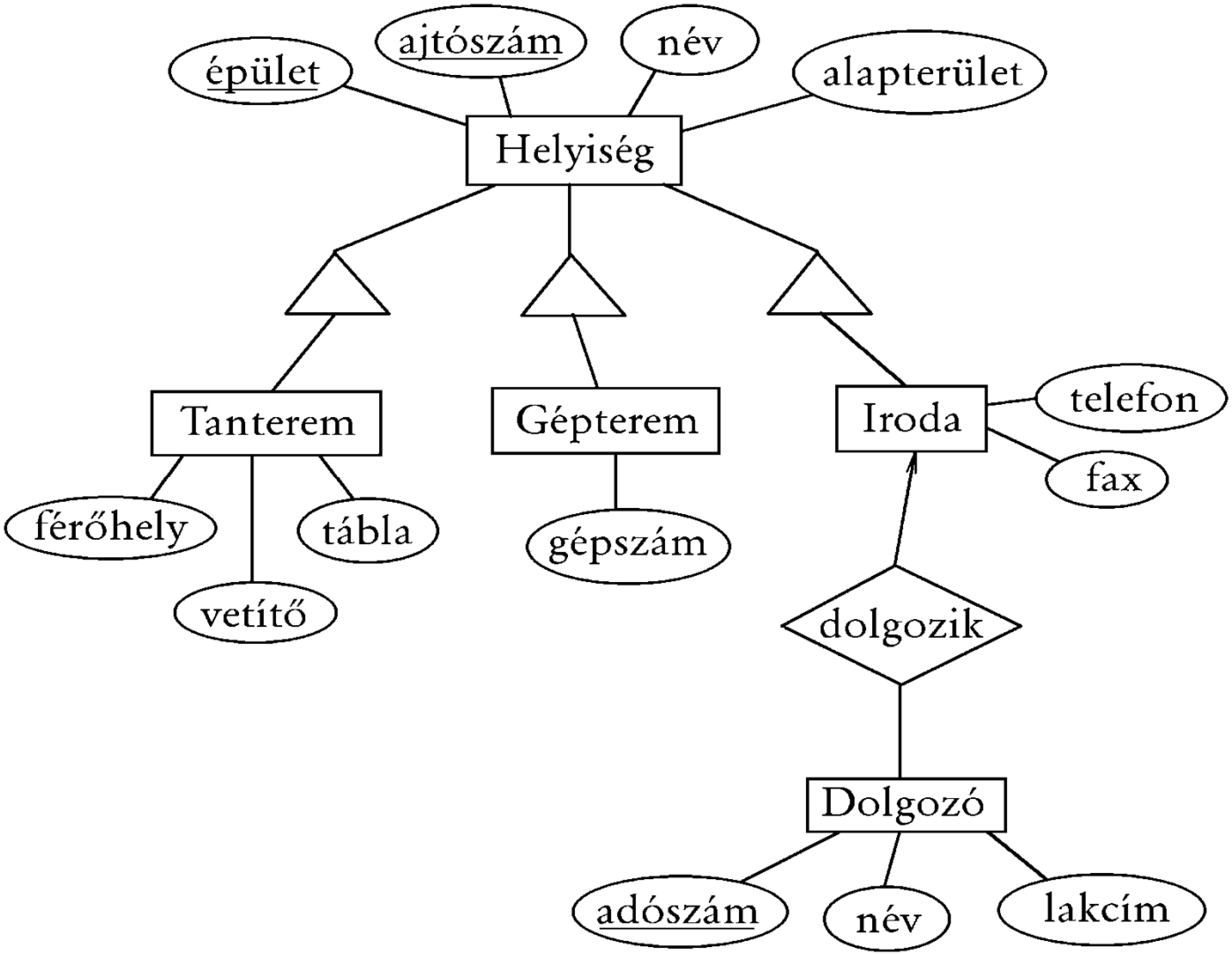
Nullértékek használatával ---3

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.

Példa: Alosztály átírása relációkká



E/K típusú átalakítás --- 1

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: Előfordulhat, hogy több táblában kell keresni (például ha a tantermek nevére és férőhelyére vagyunk kíváncsiak, akkor össze kell kapcsolni a táblákat).

Objektumorientált megközelítés --- 2

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 0-821. számú terem alapterületét keressük).
– Kombinált altípus (például számítógépes tanterem) csak új altípus felvételével kezelhető.

Nullértékek használata relációk egyesítéséhez --- 3

3. Egy közös tábla felvétele, az attribútumok uniójával.
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*)

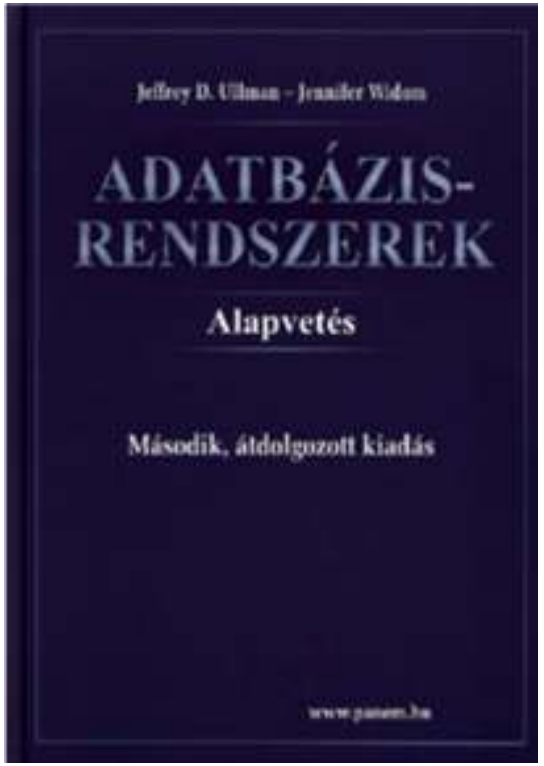
Hátrányok:

– 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).

SQL DDL: Táblák, megszorítások (constraints), triggerek, nézettáblák

Tankönyv: Ullman-Widom:
Adatbázisrendszerek Alapvetés
Második, átdolgozott kiadás,
Panem, 2009



7.1.-7.4. Megszorítások

7.5.-7.6. Triggerek

8.1.-8-2. Nézettáblák

8.5.-8.6. Tárolt nézettáblák

-- Megj.: 8.3.-8.4. Indexek (Adatbázisok-2 kurzuson lesznek)

6.3.5. Alkérdések a FROM záradékban (inline nézet)

Ismétlés: Relációs lekérdező nyelvek

Adatbázisok-1 kurzuson háromféle nyelvet tanulmányozunk:

- Relációs algebra: procedurális, algebrai megközelítés, megadjuk a kiértékelési tervet, többféle lehetőség összevetése, hatékonysági vizsgálatok.
- Datalog: deklaratív, logika alapú megközelítés, amely az összetett lekérdéseknél, például rekurziónál segítség.
- SQL szabvány relációs lekérdező nyelv: gyakorlatban, SQL története, szabványok, az SQL fő komponensei:
  SQL DDL (sémaleíró nyelv) milyen objektumok lehetnek?
  DML (adatkezelő és lekérdező nyelv), tranzakció-kezelés,
  DCL (vezérlő nyelv) milyen jogosultságok, privilégiumok?
  SQL-2003/PSM, ezt a gyakorlatban: PL/SQL (Oracle)

--- Ma az SQL DDL sémaleíró nyelv utasításait nézzük át!

Ismétlés: relációsémák definiálása

- Az SQL tartalmaz adatleíró részt (DDL), az adatbázis objektumainak a leírására és megváltoztatására.
  Objektumok leíró parancsa a CREATE utasítás.
  - CREATE – létrehozni, az objektumok leíró parancsa
  - DROP – eldobni, a teljes leírást és mindazt, ami ehhez kapcsolódott hozzáférhetetlenné válik
  - ALTER – módosítani a leírást
- A relációt az SQL-ben táblának (TABLE) nevezik,
  az SQL alapvetően háromféle táblát kezel:
  - Alaptáblák [CREATE | ALTER | DROP] TABLE
  - Nézettáblák [CREATE [OR REPLACE] | DROP] VIEW
  - Átmeneti munkatáblák (WITH záradéka a SELECT-nek)
- Alaptáblák megadása: CREATE TABLE

Megszorítások (áttekintés) => (1)

(1) Kulcsok és idegen kulcsok megadása
- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

(2) Értékekre vonatkozó feltételek
- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

(3) Sorokra vonatkozó megszorítások
- Sorra vonatkozó CHECK feltételek

(4) Megszorítások módosítása (constraints)

(5) Önálló megszorítások (assertions)

(6) Triggerek (triggers)

Idegen kulcsok megadása

- Az első előadáson a táblák létrehozásánál vettünk kiegészítő lehetőségeket Kulcs és idegen kulcs (foreign key) hivatkozási épség megadása
- Az egyik tábla egyik oszlopában szereplő értékeknek szerepelnie kell egy másik tábla bizonyos attribútumának az értékei között.
- A hivatkozott attribútumoknak a másik táblában kulcsnak kell lennie! (PRIMARY KEY vagy UNIQUE)
- Példa: **Felszolgál(söröző, sör, ár)** táblára megszorítás, hogy a sör oszlopában szereplő értékek szerepeljenek a **Sörök(név, gyártó)** táblában a név oszlop értékei között.

Idegen kulcs megadása: attribútumként

REFERENCES kulcsszó használatának két lehetősége: attribútumként vagy sémaelemként lehet megadni.

1.) Attribútumonként (egy attribútumból álló kulcsra)

Példa:

```
CREATE TABLE Sörök (
    név     CHAR(20) PRIMARY KEY,
    gyártó CHAR(20) );

CREATE TABLE Felszolgál (
    söröző CHAR(20),
    sör   CHAR(20) REFERENCES Sörök(név),
    ár    REAL );
```

Idegen kulcs megadása: sémaelemként

2.) Sémaelemként (egy vagy több attr.-ból álló kulcsra)

FOREIGN KEY (attribútum lista)

REFERENCES relációnév (attribútum lista)

Példa:

```
CREATE TABLE Sörök (
  név       CHAR(20),
  gyártó    CHAR(20),
  PRIMARY KEY (név) );

CREATE TABLE Felszolgál (
  söröző    CHAR(20),
  sör       CHAR(20),
  ár        REAL,
  FOREIGN KEY(sör) REFERENCES Sörök(név));
```

Idegen kulcs megszorítások megőrzése

- Példa: *R* = Felszolgál, *S* = Sörök.
- Egy idegen kulcs megszorítás *R* relációról *S* relációra kétféleképpen sérülhet:
  1. Egy *R*-be történő beszúrásnál vagy *R*-ben történő módosításnál *S*-ben nem szereplő értéket adunk meg.
  2. Egy *S*-beli törlés vagy módosítás „lógó” sorokat eredményez *R*-ben.

Hogyan védekezzünk? --- (1)

- Példa: *R* = Felszolgál, *S* = Sörök.
- Nem engedjük, hogy Felszolgál táblába a Sörök táblában nem szereplő sört szúrjanak be vagy Sörök táblában nem szereplő sörre módosítsák (nincs választási lehetőségünk, a rendszer visszautasítja a megszorítást sértő utasítást)
- A Sörök táblából való törlés vagy módosítás, ami a Felszolgál tábla sorait is érintheti (mert sérül az idegen kulcs megszorítás) 3-féle módon kezelhető (lásd köv.oldal)

Hogyan védekezzünk? --- (2)

1. Alapértelmezés (Default) : a rendszer nem hajtja végre a törlést.
2. Továbbgyűrűzés (Cascade): a Felszolgál tábla értékeit igazítjuk a változáshoz.
   - Sör törlése: töröljük a Felszolgál tábla megfelelő sorait.
   - Sör módosítása: a Felszolgál táblában is változik az érték.
3. Set NULL: a sör értékét állítsuk NULL-ra az érintett sorokban.

Példa: továbbgyűrűzés

- Töröljük a Bud sort a Sörök táblából:
  - az összes sort töröljük a Felszolgál táblából, ahol sör oszlop értéke ’Bud’.
- A ’Bud’ nevet ’Budweiser’-re változtatjuk:
  - a Felszolgál tábla soraiban is végrehajtjuk ugyanezt a változtatást.

Példa: Set NULL

- A Bud sort töröljük a Sörök táblából:
  - a Felszolgál tábla sör = 'Bud' soraiban a Budot cseréljük NULL-ra.
- 'Bud'-ról 'Budweiser'-re módosítunk:
  - ugyanazt kell tennünk, mint törléskor.

A stratégia kiválasztása

- Ha egy idegen kulcsot deklarálunk megadhatjuk a SET NULL és a CASCADE stratégiát is beszúrásra és törlésre is egyaránt.
- Az idegen kulcs deklarálása után ezt kell írnunk:

  ON [UPDATE, DELETE][SET NULL CASCADE]
- Ha ezt nem adjuk meg, a default stratégia működik.

Példa: stratégia beállítása

```
CREATE TABLE Felszolgál (
  söröző CHAR(20),
  sör          CHAR(20),
  ár           REAL,
  FOREIGN KEY(sör)
     REFERENCES Sörök(név)
     ON DELETE SET NULL
     ON UPDATE CASCADE
);
```

Megszorítások ellenőrzésének késleltetése

- Körkörös megszorítások miatt szükség lehet arra, hogy a megszorításokat ne ellenőrizze, amíg az egész tranzakció be nem fejeződött.
- Bármelyik megszorítás deklarálható DEFERRABLE (késleltethető) vagy NOT DEFERRABLE-ként (vagyis minden adatbázis módosításkor a megszorítás közvetlenül utána ellenőrzésre kerül). DEFERRABLE-ként deklaráljuk, akkor lehetőségünk van arra, hogy a megszorítás ellenőrzésével várjon a rendszer a tranzakció végéig.
- Ha egy megszorítás késleltethető, akkor lehet
  - INITIALLY DEFERRED (az ellenőrzés a tranzakció jóváhagyásáig késleltetve lesz) vagy
  - INITIALLY IMMEDIATE (minden utasítás után ellenőrzi)

Megszorítások (áttekintés) => (2)

(1) Kulcsok és idegen kulcsok
- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

(2) Értékekre vonatkozó feltételek
- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

(3) Sorokra vonatkozó megszorítások
- Sorra vonatkozó CHECK feltételek

(4) Megszorítások módosítása (constraints)

(5) Önálló megszorítások (assertions)

(6) Triggerek (triggers)

Értékekre vonatkozó feltételek

- Egy adott oszlop értékeire vonatkozóan adhatunk meg megszorításokat.
- A CREATE TABLE utasításban az attribútum deklarációban NOT NULL kulcsszóval
- az attribútum deklarációban CHECK(<feltétel>)
  A feltétel, mint a WHERE feltétel, alkérdés is használható. A feltételben csak az adott attribútum neve szerepelhet, más attribútumok (más relációk attribútumai is) csak alkérdésben szerepelhetnek.

Példa: értékekre vonatkozó feltétel

```
CREATE TABLE Felszolgál (
  söröző CHAR(20) NOT NULL,
  sör          CHAR(20) CHECK ( sör IN
                (SELECT név FROM Sörök)),
  ár            REAL CHECK ( ár <= 5.00 )
);
```

Mikor ellenőrzi?

- Érték-alapú ellenőrzést csak beszúrásnál és módosításnál hajt végre a rendszer.
  - Példa: `CHECK (ár <= 5.00)` a beszúrt vagy módosított sor értéke nagyobb 5, a rendszer nem hajtja végre az utasítást.
  - Példa: `CHECK (sör IN (SELECT név FROM Sörök))`, ha a Sörök táblából törlünk, ezt a feltételt nem ellenőrzi a rendszer.

Megszorítások (áttekintés) => (3)

(1) Kulcsok és idegen kulcsok
- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

(2) Értékekre vonatkozó feltételek
- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

(3) Sorokra vonatkozó megszorítások
- Sorra vonatkozó CHECK feltételek

(4) Megszorítások módosítása (constraints)

(5) Önálló megszorítások (assertions)

(6) Triggerek (triggers)

Sorokra vonatkozó megszorítások

- A CHECK (<feltétel>) megszorítás a séma elemeként is megadható.
- A feltételben tetszőleges oszlop és reláció szerepelhet.
  - De más relációk attribútumai csak alkérdésben jelenhetnek meg.
- Csak beszúrásnál és módosításnál ellenőrzi a rendszer.

Példa: sor-alapú megszorítások

- Csak Joe bárja nevű söröző̋ben lehetnek drágábbak a sörök 5 dollárnál:

```
CREATE TABLE Felszolgál (
    söröző  CHAR(20),
    sör     CHAR(20),
    ár      REAL,
    CHECK (söröző= 'Joe bárja'
                OR ár <= 5.00)
);
```

Tankönyv példája sor alapú megszorításra

Attribútumokra és sorokra vonatkozó megszorítások

Példa: Ha egy színész neme férfi, akkor

a neve nem kezdődhet ‘Ms.’-el

```
CREATE TABLE FilmSzínész (
  név CHAR(30) PRIMARY KEY,
  cím VARCHAR(255) NOT NULL,
  nem CHAR(1),
  születésiDátum DATE,
  CHECK (nem = 'N' OR név NOT LIKE 'Ms.%')
);
```

Megszorítások (áttekintés) => (4)

(1) Kulcsok és idegen kulcsok
- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

(2) Értékekre vonatkozó feltételek
- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

(3) Sorokra vonatkozó megszorítások
- Sorra vonatkozó CHECK feltételek

(4) Megszorítások módosítása (constraints)

(5) Önálló megszorítások (assertions)

(6) Triggerek (triggers)

Megszorítások elnevezése

- Nevet tudunk adni a megszorításoknak, amire később tudunk hivatkozni (könnyebben lehet később majd törölni, módosítani)

Tankönyv példái:

- név CHAR(30) CONSTRAINT NévKulcs
  PRIMARY KEY,
- nem CHAR(1) CONSTRAINT FérfiVagyNő
  CHECK (nem IN ('F', 'N')),
- CONSTRAINT Titulus
  CHECK (nem = 'N' OR név NOT LIKE 'Ms.\%')

Megszorítások módosítása

Tankönyv példái:

- ALTER TABLE FilmSzínész ADD CONSTRAINT NévKulcs PRIMARY KEY (név);
- ALTER TABLE FilmSzínész ADD CONSTRAINT FérfiVagyNő CHECK (nem IN ('F', 'N'));
- ALTER TABLE FilmSzínész ADD CONSTRAINT Titulus CHECK (nem = 'N' OR név NOT LIKE 'Ms.\%');

Megszorítások (áttekintés) => (5)

(1) Kulcsok és idegen kulcsok
- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

(2) Értékekre vonatkozó feltételek
- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

(3) Sorokra vonatkozó megszorítások
- Sorra vonatkozó CHECK feltételek

(4) Megszorítások módosítása (constraints)

(5) Önálló megszorítások (assertions)

(6) Triggerek (triggers)

Önálló megszorítások: Assertions

- SQL aktív elemek közül a leghatékonyabbak nincs hozzárendelve sem sorokhoz, sem azok komponenseihez, hanem táblákhoz kötődnek.
- Ezek is az adatbázissémához tartoznak a relációsémákhoz és nézetekhez hasonlóan.
- CREATE ASSERTION <név>
  CHECK (<feltétel>);
- A feltétel tetszőleges táblára és oszlopra hivatkozhat az adatbázissémából.

Példa: önálló megszorítások

- A Felszolgál(söröző, sör, ár) táblában nem lehet olyan söröző, ahol a sörök átlagára 5 dollárnál több

```
CREATE ASSERTION CsakOlcsó CHECK
 (
NOT EXISTS (
    SELECT söröző
    FROM Felszolgál
    GROUP BY bár
    HAVING 5.00 < AVG(ár)
));
```

(SELECT .. olyan sörözők, ahol a sörök átlagosan drágábbak 5 dollárnál)

Példa: önálló megszorítások

- Az Sörvivó(név, cím, telefon) és Söröző(név, cím, engedély) táblákban nem lehet több bár, mint amennyi sörivó van.

```
CREATE ASSERTION KevésBár CHECK (
    (SELECT COUNT(*) FROM Söröző) <=
    (SELECT COUNT(*) FROM Sörivó)
);
```

Önálló megszorítások ellenőrzése

- Alapvetően az adatbázis bármely módosítása előtt ellenőrizni kell.
- Egy okos rendszer felismeri, hogy mely változtatások, mely megszorításokat érinthetnek.
  - Példa: a Sörök tábla változásai nincsenek hatással az iménti KevésBár megszorításra. Ugyanez igaz a Sörivók táblába történő beszúrásokra is.

Megszorítások (áttekintés) => (6)

(1) Kulcsok és idegen kulcsok
- A hivatkozási épség fenntartása
- Megszorítások ellenőrzésének késleltetése

(2) Értékekre vonatkozó feltételek
- NOT NULL feltételek
- Attribútumra vonatkozó CHECK feltételek

(3) Sorokra vonatkozó megszorítások
- Sorra vonatkozó CHECK feltételek

(4) Megszorítások módosítása (constraints)

(5) Önálló megszorítások (assertions)

(6) Triggerek (triggers)

Megszorítások v.s. triggerek

- Aktív elemek – olyan kifejezés vagy utasítás, amit egyszer eltároltunk az adatbázisban és azt várjuk tőle, hogy a megfelelő pillanatban lefusson (pl. adatok helyességének ellenőrzése)
- A megszorítás adatelemek közötti kapcsolat, amelyet az adatbázis-kezelő rendszernek fent kell tartania.
- Triggerek olyankor hajtódnak végre, amikor valamilyen megadott esemény történik, mint például sorok beszúrása egy táblába.

Miért hasznosak a triggerek?

- Az önálló megszorításokkal (assertions) sok mindent le tudunk írni, az ellenőrzésük azonban gondot jelenthet.
- Az attribútumokra és sorokra vonatkozó megszorítások ellenőrzése egyszerűbb (tudjuk mikor történik), ám ezekkel nem tudunk minden kifejezni.
- A triggerek esetén a felhasználó mondja meg, hogy egy megszorítás mikor kerüljön ellenőrzésre.

Esemény-Feltétel-Tevékenység szabályok

- A triggereket esetenként *ECA szabályoknak* (*event-condition-action*) *esemény-feltétel-tevékenység szabályoknak* is nevezik.
- Esemény: általában valamilyen módosítás a adatbázisban, INSERT, DELETE, UPDATE.
- Mikor?: BEFORE, AFTER, INSTEAD
- Mit?: OLD ROW, NEW ROW FOR EACH ROW
  OLD/NEW TABLE FOR EACH STATEMENT
- Feltétel : SQL igaz-hamis-ismeretlen feltétel.
- Tevékenység : SQL utasítás, BEGIN..END, PSM tárolt eljárás

Példa triggerre

- Ahelyett, hogy visszautasítanánk a Felszolgál(söröző, sör, ár) táblába történő beszúrást az ismeretlen sörök esetén, a Sörök(név, gyártó) táblába is beszúrjuk a megfelelő sort a gyártónak NULL értéket adva.

Példa: trigger definíció

```
CREATE TRIGGER SörTrig
  AFTER INSERT ON Felszolgál
  REFERENCING NEW ROW AS ÚjSor
  FOR EACH ROW
  WHEN (ÚjSor.sör NOT IN
     (SELECT név FROM Sörök))
  INSERT INTO Sörök(név)
     VALUES(ÚjSor.sör);
```

Esemény

Feltétel

Tevékenység

Triggerek --- 1

- A *triggerek,* amelyeket szokás *esemény-feltétel-tevékenység szabályoknak* is nevezni, az eddigi megszorításoktól három dologban térnek el:
- A triggereket a rendszer csak akkor ellenőrzi, ha bizonyos *események* bekövetkeznek.
  A megengedett események általában egy adott relációra vonatkozó beszúrás, törlés, módosítás, vagy a tranzakció befejeződése.

Triggerek --- 2

- A kiváltó esemény azonnali megakadályozása helyett a trigger először egy *feltételt* vizsgál meg
- Ha a trigger feltétele teljesül, akkor a rendszer végrehajtja a triggerhez tartozó *tevékenységet*. Ez a művelet ezután megakadályozhatja a kiváltó esemény megtörténtét, vagy meg nem történtté teheti azt.

Tankönyv példája (7.5. ábra)

```
-- Nem engedi csökkenteni a gyártásirányítók nettó bevételét:
   CREATE TRIGGER NetBevétTrigger
   AFTER UPDATE OF nettóBevétel ON GyártásIrányító
   REFERENCING
        OLD ROW AS RégiSor,
        NEW ROW AS ÚjSor
   FOR EACH ROW
   WHEN(RégiSor.nettóBevétel > ÚjSor.nettóBevétel)
        UPDATE GyártásIrányító
        SET nettóBevétel = RégiSor.nettóBevétel
        WHERE azonosító = ÚjSor.azonosító;
```

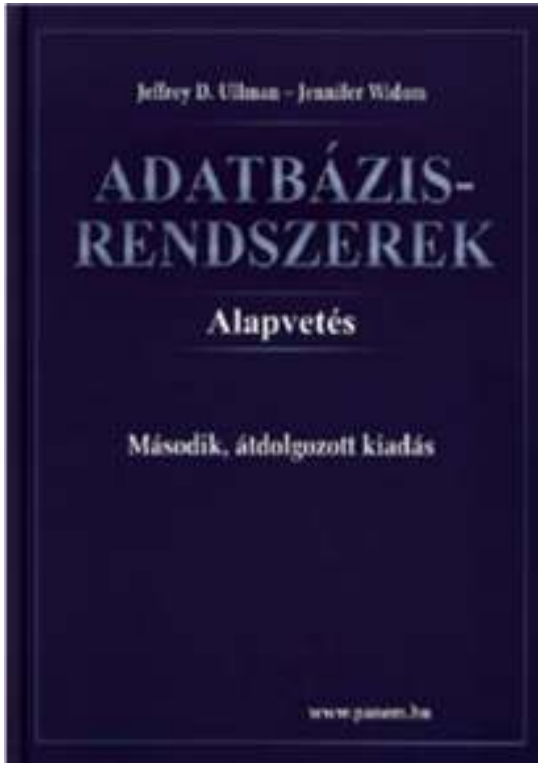
Tankönyv példája (7.6. ábra)

-- Az átlagos nettó bevétel megszorítása:

```
CREATE TRIGGER ÁtlagNetBevétTrigger
AFTER UPDATE OF nettóBevétel ON GyártásIrányító
REFERENCING
    OLD TABLE AS RégiAdat,
    NEW TABLE AS ÚjAdat
FOR EACH STATEMENT
WHEN(500000 > (SELECT AVG(nettóBevétel)
                FROM GyártásIrányító)
    DELETE FROM GyártásIrányító
    WHERE (név, cím, azonosító) IN ÚjAdat;
    INSERT INTO gyártásIrányító (SELECT…);
```

SQL DDL: nézettáblák(VIEW)
SQL DML: inline nézetek (SELECT)

Tankönyv: Ullman-Widom:
Adatbázisrendszerek Alapvetés
Második, átdolgozott kiadás,
Panem, 2009



8.1.-8-2. Nézettáblák (View)

8.5.-8.6. Tárolt nézettáblák

-- Megj.: 8.3.-8.4. Indexek
(Adatbázisok-2 kurzuson lesznek)

6.3.5. Alkérdések a FROM záradékban (inline nézet)

Nézettáblák

- Ez volt a Tankönyv 7.fejezete az integritási megszorításokról és a triggerekről.
- Ezután következik a Tankönyv 8.fejezete a nézettáblákról, és az adatok módosításáról a nézettáblákon keresztül.

Mik a nézettáblák?

- A nézettábla olyan reláció, amit tárolt táblák (vagyis alaptáblák) és más nézettáblák felhasználásával definiálunk.
- **`EMPLOYEES` table**

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALA
100	Steven	King	SKING	515.123.4567	17-JUN-87	AD_PRES	240
101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	170
102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	170
103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	90
104	Bruce	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	60
107	Diana	Lorentz	DLORENTZ	590.423.5567	07-FEB-99	IT_PROG	42
124	Kevin	Mourgos	KMOURGOS	650.123.5234	16-NOV-99	ST_MAN	58
141	Trenna	Rajs	TRAJS	650.121.8009	17-OCT-95	ST_CLERK	35
142	Curtis	Davies	CDAVIES	650.121.2994	29-JAN-97	ST_CLERK	31
143	Randall	Matos	RMATOS	650.121.2874	15-MAR-98	ST_CLERK	26
144	Peter	Vargas	PVARGAS	650.121.2004	09-JUL-98	ST_CLERK	25
149	Eleni	Zlotkey	EZLOTKEY	011.44.1344.429018	29-JAN-00	SA_MAN	105
174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-96	SA_REP	110
176	Jonathon	Taylor	JTAYLOR	011.44.1644.429265	24-MAR-98	SA_REP	86
178	Kimberely	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	70
200	Jennifer	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	44
201	Michael	Hartstein	MHARTSTE	515.123.5555	17-FEB-96	MK_MAN	130
202	Pat	Fay	PFAY	603.123.6666	17-AUG-97	MK_REP	60
205	Shelley	Higgins	SHIGGINS	515.123.8080	07-JUN-94	AC_MGR	120
206	William	Gietz	WGIETZ	515.123.8181	07-JUN-94	AC_ACCOUNT	83

20 rows selected.

EMPLOYEE_ID	LAST_NAME	SALARY
149	Zlotkey	10500
174	Abel	11000
176	Taylor	8600

A nézettáblák előnyei

Korlátozható az adatok elérése

Bonyolult lekérdezések egyszerűbben kifejezhetők

Az adatfüggetlenség biztosítása

Ugyanazon adatokat különböző nézőpontokból szemlélhetjük

Virtuális vagy materializált?

- Kétféle nézettábla létezik:
  - Virtuális = nem tárolódik az adatbázisban, csak a relációt megadó lekérdezés.
  - Materializált = kiszámítódik, majd tárolásra kerül.

Nézettáblák létrehozása és törlése

- Létrehozása:

```
CREATE [OR REPLACE][FORCE|NOFORCE]
[MATERIALIZED] VIEW <név>
AS <lekérdezés>

[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

- Alapesetben virtuális nézettábla jön létre.
- Nézettábla megszüntetése:

```
DROP VIEW <név>;
```

Példa: nézettábla létrehozása

- Mit_ihat(név, sör) nézettáblában a sörivók mellett azon söröket tároljuk, amelyeket legalább egy olyan söröző felszolgálnak, amelyet látogat:

```
CREATE VIEW Mit_ihat AS
    SELECT név, sör
    FROM Látogat, Felszolgál
    WHERE L.söröző = F.söröző;
```

Példa: nézettáblákhoz való hozzáférés

- A nézettáblák ugyanúgy kérdezhetők le, mint az alaptáblák.
  - A nézettáblákon keresztül az alaptáblák néhány esetben módosíthatóak is, ha a rendszer a módosításokat át tudja vezetni (lásd módosítások, SQL DML)
- Példa lekérdezés:

```
SELECT sör FROM Mit_ihat
WHERE név = 'Sally';
```

Módosítható nézettáblák

- Az SQL szabvány formálisan leírja, hogy mikor lehet egy nézettáblát módosítani és mikor nem, ezek a szabályok meglehetősen bonyolultak.
  - Ha a nézettábla definíciójában a SELECT után nem szerepel DISTINCT, további kikötések:
  - A WHERE záradékban R nem szerepelhez egy alkérdésben sem
  - A FROM záradékban csak R szerepelhet, az is csak egyszer és más reláció nem
  - A SELECT záradék listája olyan attribútumokat kell, hogy tartalmazzon, hogy az alaptáblát fel lehessen tölteni (vagyis kötelező a kulcsként vagy not null-nak deklarált oszlopok megadása)

Tankönyv példája: nézettáblára

Tk.8.1. Példa: Egy olyan nézettáblát szeretnénk, mely a Film(cím, év, hossz, színes, stúdióNév, producerAzon) reláció egy részét jelképezi, pontosabban a Paramount stúdió által gyártott filmek címét és gyártási évét

```
CREATE VIEW ParamountFilm AS
    SELECT cím, év
    FROM Film
    WHERE stúdióNév = 'Paramount';
```

Nézeteken instead-of-triggerek

Tk. 8.8. Példa: Az előző nézettábla módosítható, és hogy az alaptáblába való beszúráskor a stúdióNév attribútum helyes értéke , 'Paramount' legyen, ezt biztosítja ez az INSTEAD OF (helyette) típusú trigger:

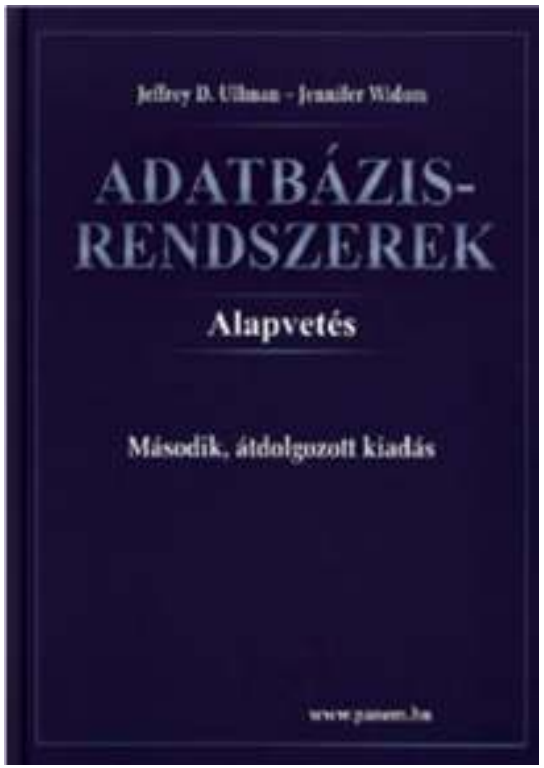
```
CREATE TRIGGER ParamountBeszúrás
    INSTEAD OF INSERT ON ParamountFilm
    REFERENCING NEW ROW AS ÚjSor
    FOR EACH ROW
    INSERT INTO Film(cím, év, stúdióNév)
    VALUES(Újsor.cím, ÚjSor.év, 'Paramount');
```

Materializált (tárolt) nézettáblák

- Adattárházaknál használják (MSc kurzusok)
- Probléma: minden alkalommal, amikor az alaptáblák valamelyike változik, a materializált nézettábla frissítése is szükségessé válhat.
  - Ez viszont néha túl költséges.
- Megoldás: Periodikus frissítése a materializált nézettábláknak, amelyek egyébként „nem aktuálisak”.

SQL DML: inline nézetek - alkérdések SELECT utasítás FROM záradékban

Tankönyv: Ullman-Widom:
Adatbázisrendszerek Alapvetés
Második, átdolgozott kiadás,
Panem, 2009



6.3.5. Alkérdések a FROM záradékban

Feladatok: lekérdezések megadása relációs algebrai kifejezések átírásai

Ismétlés: Alkérdések

- A FROM listán és a WHERE záradékban (valamint a GROUP BY HAVING záradékában) zárójelezett SFW SELECT-FROM-WHERE utasításokat (alkérdéseket) is használhatunk.
- Szintaktikus alakja: zárójelbe kell tenni a lekérdezést
- Hol használható? Ott, ahol relációnevet használunk:

(1) WHERE és HAVING záradékban: kifejezésekben, feltételekben

(2) FROM listában: új listaelem (rel.név változó SQL-ben)

(lekérdezés) [AS] sorváltozó

Ez felel meg annak, ahogyan a relációs algebrában tetsz.helyen használhattuk a lekérdezés eredményét.

Új anyag: Alkérdések a FROM listán

- A FROM listán és a WHERE záradékban valamint a HAVING záradékban zárójelezett SELECT-FROM-WHERE utasításokat (alkérdéseket) is használhatunk.

(1) A gyakorlaton az I.ZH-ban csak a WHERE záradékban valamint a HAVING záradékban lehet használni majd alkérdéseket, FROM listán nem! A nézettáblákkal és az ún.inline nézetekkel való megoldások csak II.ZH-ban.

(2) FROM listában: a tényleges relációk helyett alkérdések, másodnevet adunk, ezzel tudunk a soraira hivatkozni. FROM új listaelem: (lekérdezés) [AS] sorváltozó

Ez felel meg annak, ahogyan a relációs algebrában tetsz.helyen használhattuk a lekérdezés eredményét.

Alkérdések használata FROM listán

- FROM záradékban alkérdéssel létrehozott ideiglenes táblát is megadhatunk. Ilyenkor a legtöbb esetben meg kell adnunk a sorváltozó nevét. **Szintaktikus alakja**:

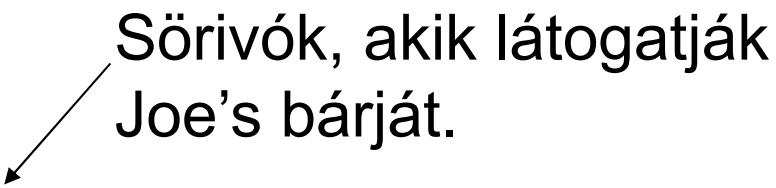
  (lekérdezés) [AS] sorváltozó

- **Szemantikája**: A FROM záradékban kiértékelődik az alkérdés, utána a sorváltozót ugyanúgy használjuk, mint a közönséges adatbázis relációkat.

- **Példa:** Keressük meg a Joe's bár vendégei által kedvelt söröket (a feladatnak sok megoldása van)

Alkérdések használata FROM listán

- FROM záradékban alkérdéssel létrehozott ideiglenes táblát is megadhatunk. Ilyenkor a legtöbb esetben meg kell adnunk a sorváltozó nevét.
- **Példa:** Keressük meg a Joe's bár vendégei által kedvelt söröket.




```
SELECT sör
FROM Szeret, (SELECT név
              FROM Látogat
              WHERE bár = 'Joe''s bar')JD
WHERE Szeret.név = JD.név;
```

Példák relációs algebrai lekérdezésekre

- Tk.2.4.1.feladat : Relációs algebra kifejezések használata a lekérdezések megadására, átírás SQL SELECT-re:
- **Példa:** Adottak az alábbi relációs sémák feletti relációk

Termék (gyártó, modell, típus)

PC (modell, sebesség, memória, merevlemez, cd, ár)

Laptop (modell, sebesség, memória, merevlemez, képernyő, ár)

Nyomtató (modell, színes, típus, ár)

d.) Adjuk meg valamennyi színes lézernyomtató modellszámát: $\Pi_m(\sigma_{sz='i'}(Ny)) \cap \Pi_m(\sigma_{t='lézer'}(Ny))$

-- elvégezhető más módon is: $\Pi_m(\sigma_{sz='i' \wedge t='lézer'}(Ny)) =$

$= \Pi_m(\sigma_{sz='i'}\, \sigma_{t='lézer'}(Ny)) = \Pi_m(\sigma_{t='lézer'}\, \sigma_{sz='i'}(Ny))$

Átírás SELECT-re: ennél a két utolsónál inline nézettel!

Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?
- Házi feladat: Gyakorlás az Oracle Példatár feladatai:
- DML-utasítások, tranzakciók (lásd 5EA)
  - Változóhasználat (Példatár 4.fej., + PL/SQL: 8.fej. is)
  - DML-utasítások: insert, update, delete (Példatár 5.fej.)
  - Adatbázis-tranzakciók: commit, rollback, savepoint
- DDL-utasítások (lásd 6EA)
  - DDL-utasítások: adattáblák létrehozása, módosítása, integritási megszorítások (Példatár 5.fejezet folyt.) és
  - Nézettábla létrehozása és törlése, táblák tartalmának módosítása nézettáblákon keresztül (Példatár 6.fej.)

http://people.inf.elte.hu/sila/eduAB/Feladatok.pdf