

Több táblára vonatkozó lekérdezések

Relációs algebra és SQL SELECT

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

4.fej. Tervezés: E/K-modell elemei

2.3.+7.1. Megvalósítás: Több tábla
és megszorítások megadása (SQL)

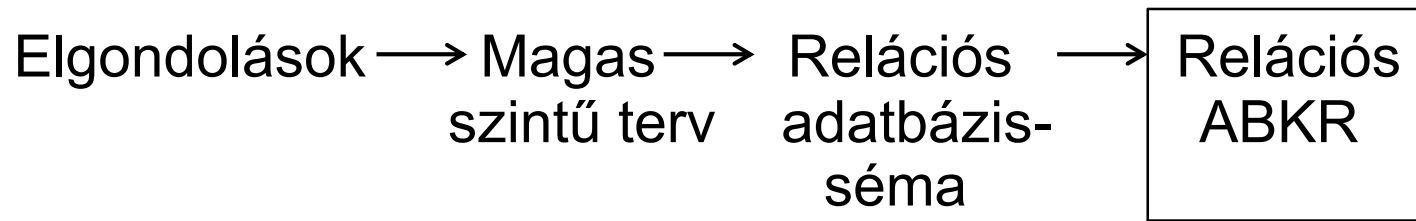
2.4. Lekérdezések: Relációs algebra

6.2. Több tábla lekérdezése SQL-ben



Magas szintű adatbázismodellek

- Vizsgáljuk meg azt a folyamatot, amikor egy új adatbázist létrehozunk, vegyünk 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...

Egyed-kapcsolat modell elemei

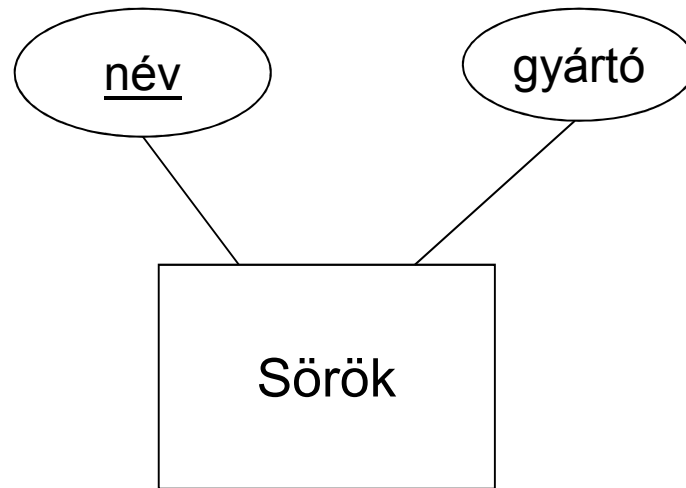
- **Egyed-kapcsolat modell: E/K modell**
(Entity-relationship ER) alapfogalmak:
- **Egyedhalmazok** (absztrakt objektumok osztálya)
 - Miről gyűjtünk adatokat?
 - Mit tegyünk egy gyűjteménybe? - hasonlóság
 - Hasonló egyedek összessége
- **Attribútumok**
 - Megfigyelhető tulajdonságok, megfigyelt értékek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

E/K modell elemei: Egyedhalmazok

- $E(A_1, \dots, A_n)$ egyedhalmaz **séma**:
 - E az egyedhalmaz neve,
 - A_1, \dots, A_n tulajdonságok,
 - $\text{DOM}(A_i)$ – lehetséges értékek halmaza.
 - például: tanár(név, tanszék).
- $E(A_1, \dots, A_n)$ sémájú egyedhalmaz **előfordulása**:
 - A konkrét egyedekből áll
 - $E = \{e_1, \dots, e_m\}$ egyedek (entitások) halmaza, ahol
 - $e_i(k) \in \text{DOM}(A_k)$,
 - semelyik két egyed sem egyezik meg minden attribútumban (léteznek és megkülönböztethetők)

E/K-diagram: Egyedhalmazok

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

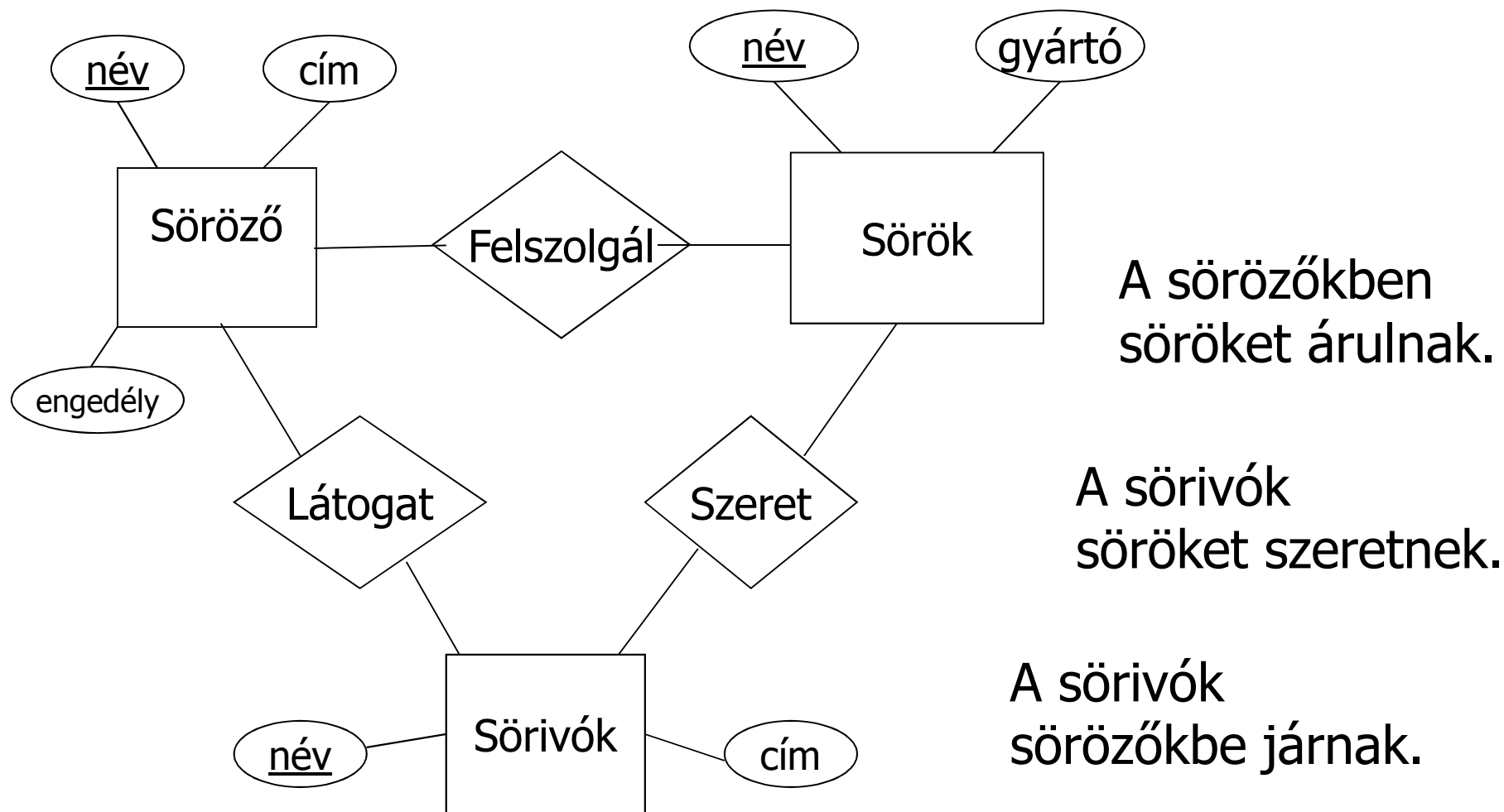


E/K modell elemei: Kapcsolatok

- $K(E_1, \dots, E_p)$ a kapcsolat sémája,
 - K a kapcsolat neve,
 - E_1, \dots, E_p egyedhalmazok sémái,
 - $p=2$ bináris kapcsolat, $p>2$ többágú kapcsolat,
 - például: tanít(tanár, tárgy).
- $K(E_1, \dots, E_p)$ sémájú kapcsolat előfordulása:
 - $K = \{(e_1, \dots, e_p)\}$ egyed p -esek halmaza, ahol
 - $e_i \in E_i$,
 - a kapcsolat előfordulásaira tett megszorítások határozzák meg a kapcsolat típusát.

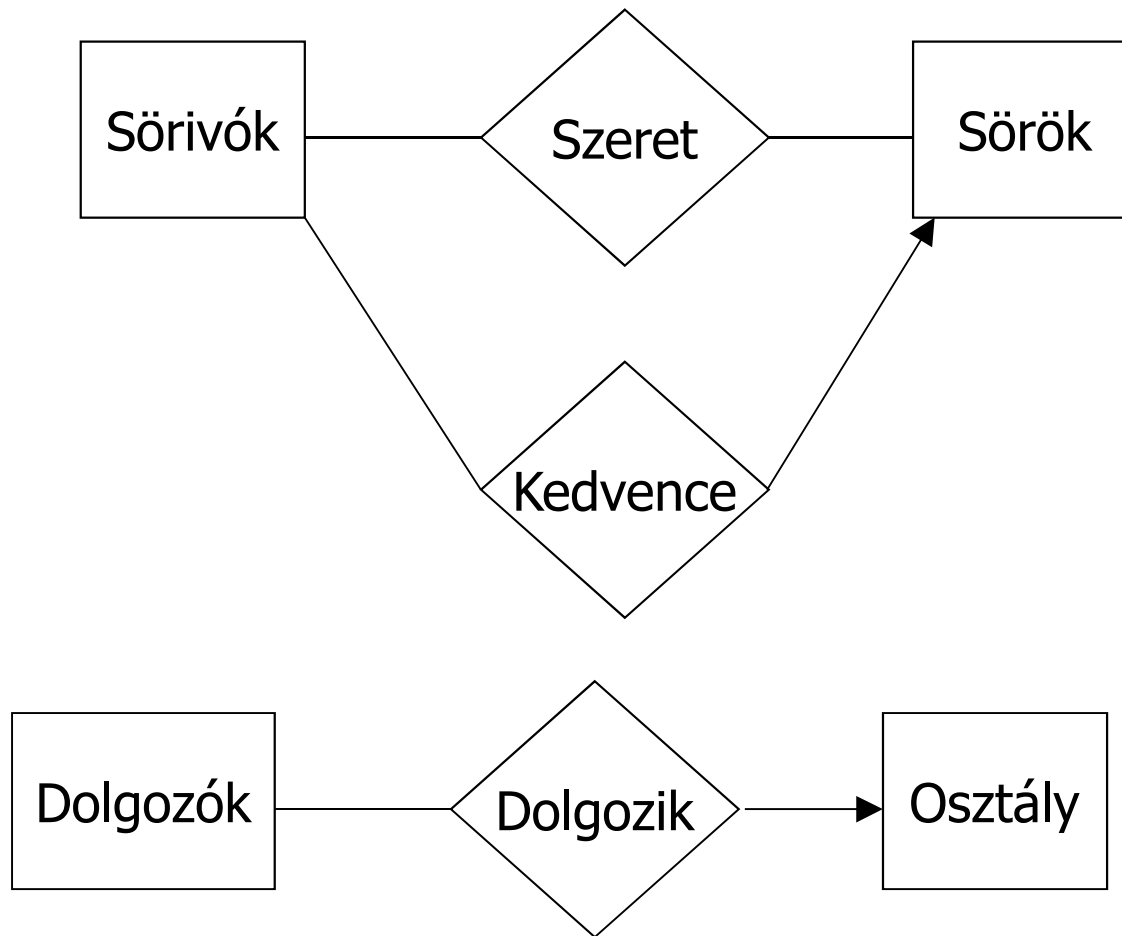
E/K-diagram: Kapcsolatok

➤ A kapcsolatok jele: **rombusz**

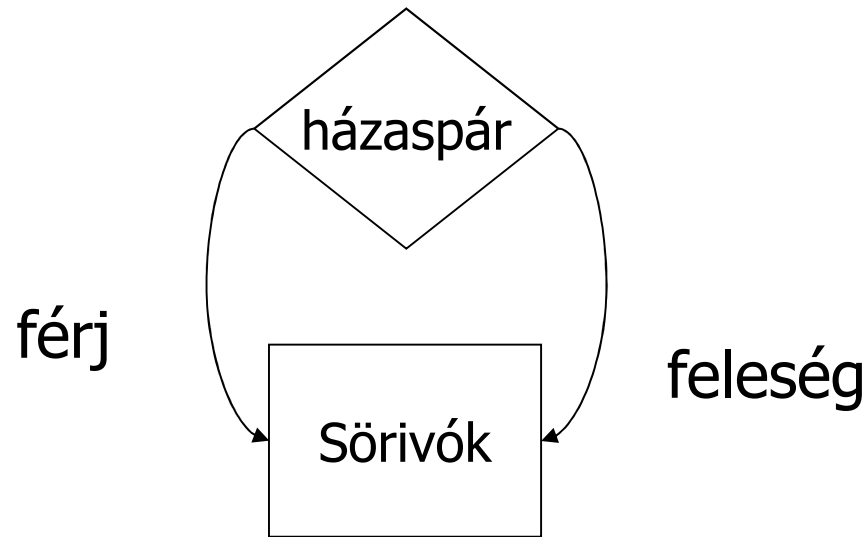


Kapcsolatok típusai: sok-egy, sok-sok

(két egyedhalmaz között több kapcsolat is lehet)



Egy egyedhalmaz önmagával is kapcsolódhat: Szerepek (Roles)

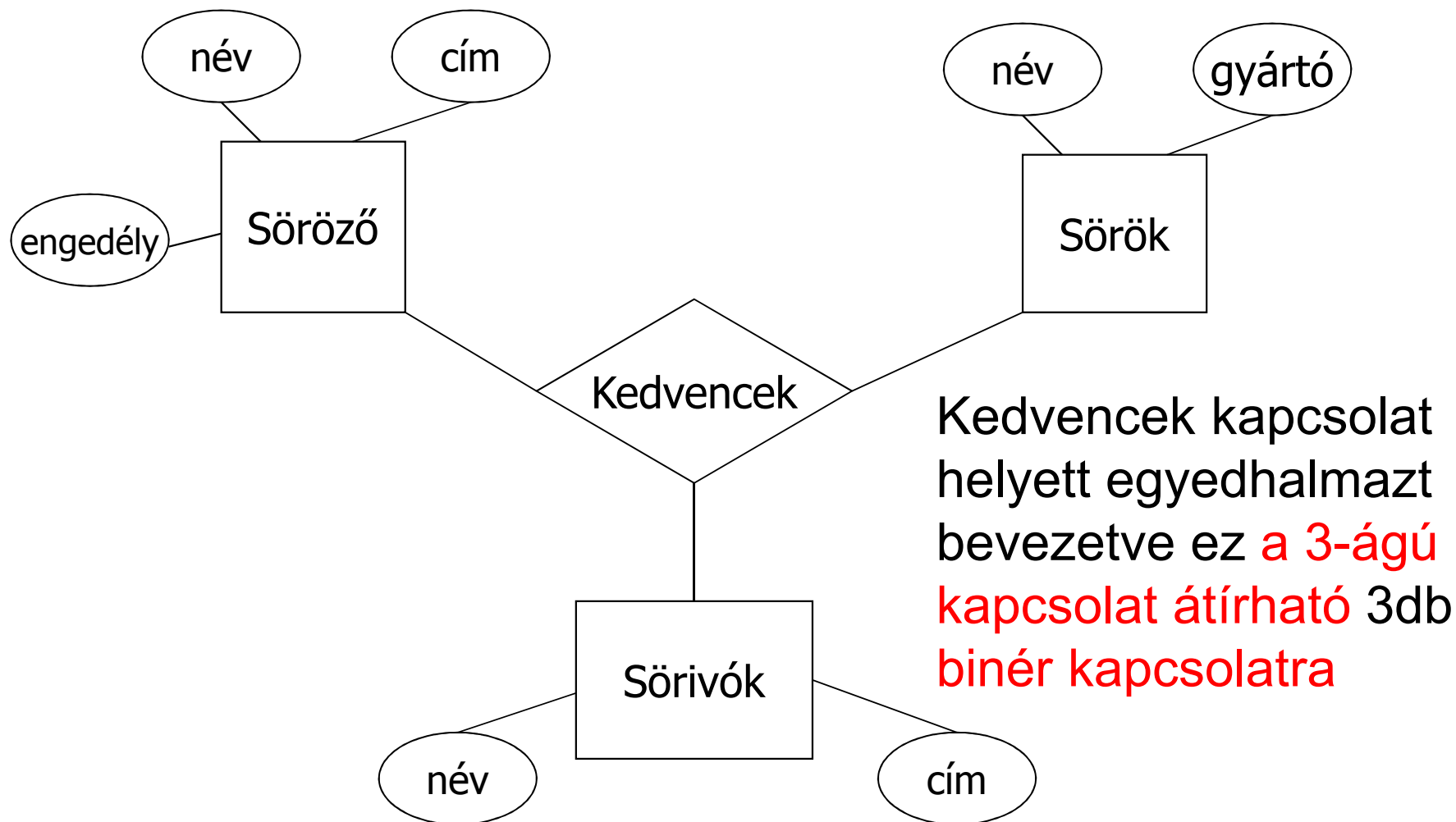


A kapcsolat előfordulása

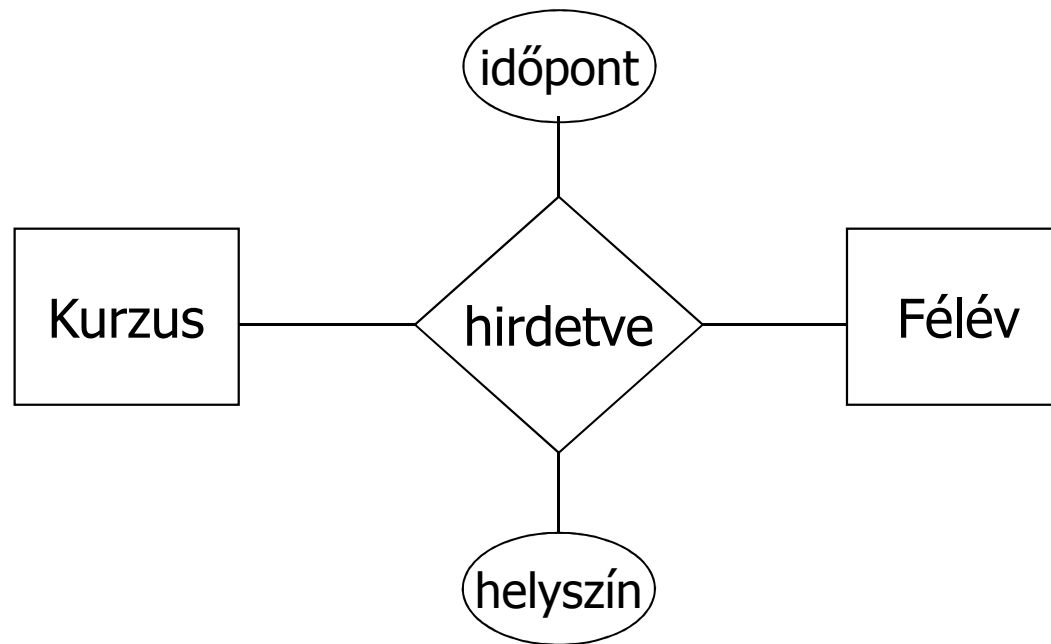
Férj	Feleség
Bob	Ann
Joe	Sue
...	...

Dolgozók egyedhalmaz is kapcsolódik önmagával:
ki kinek a főnöke (sok-egy kapcsolat)

Példa: Többágú (3-ágú) kapcsolatra

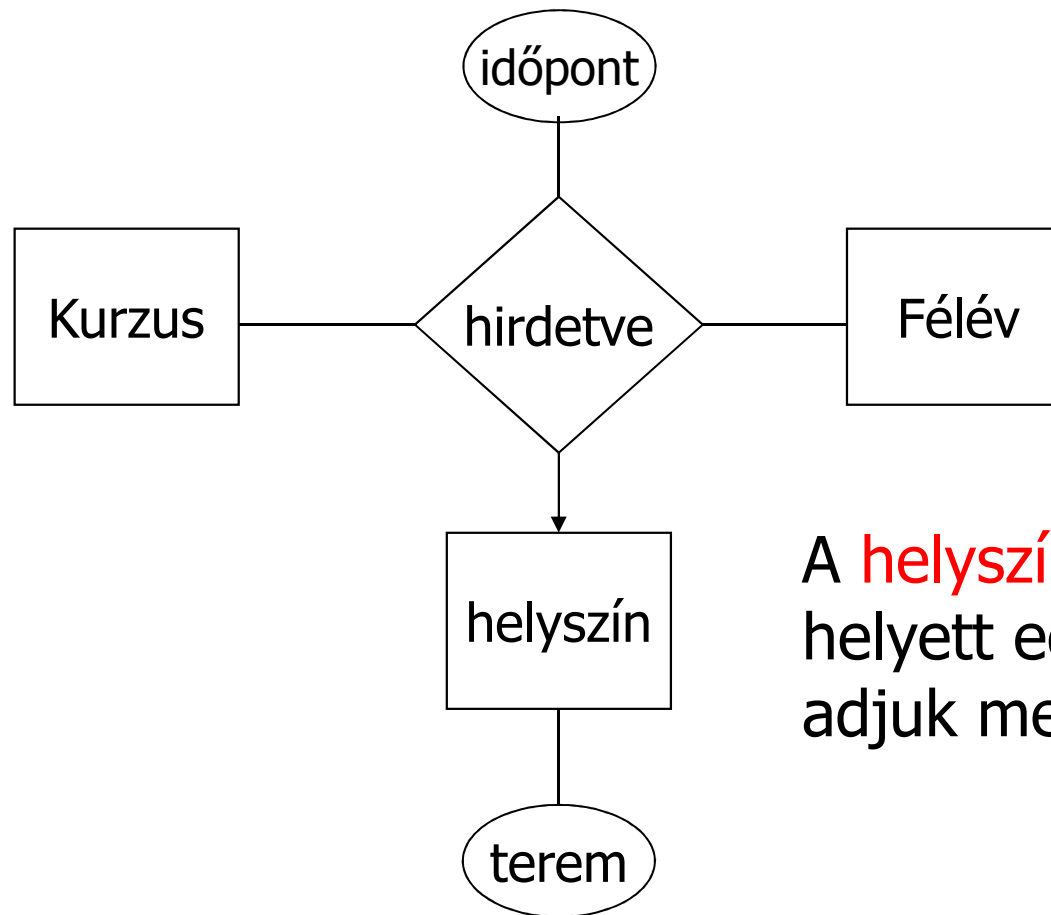


Kapcsolatnak is lehet attribútuma



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

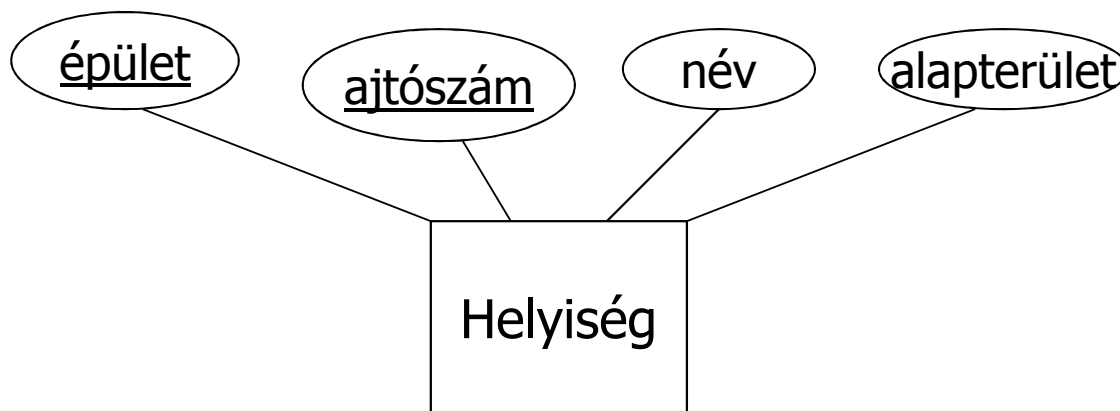
Kulcs megszorítás

jele: aláhúzás

Példa egyszerű kulcsra: név a Sörök elsődleges kulcsa:



Példa összetett kulcsra: épület, ajtószám két-attribútumos elsődleges kulcsa a Helyiség-nek:



E/K-diagram átírása

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

- Egyedhalmazok átírása relációkká
 - E/K-kapcsolatok átírása relációkká
 - Egyszerűsítés, összevonások
- folyt.köv. későbbi előadáson:
- Gyenge egyedhalmazok kezelése
 - Osztályhierarchia átalakítása relációkká

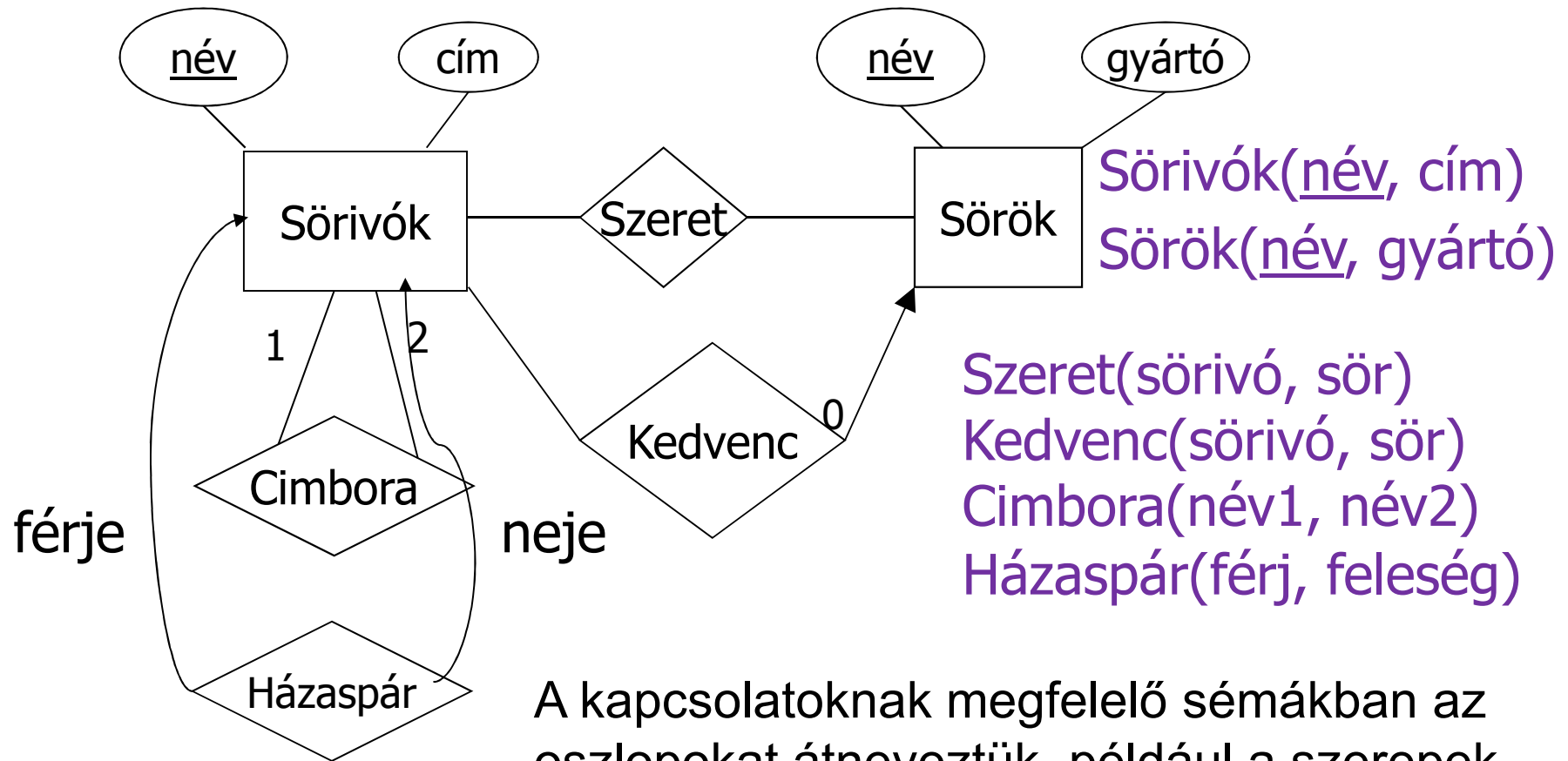
E/K diagram átírása relációs adatbázistervre

Mi minek felel meg:

➤ egyedhalmaz séma	↔	relációséma
$E(A_1, \dots, A_n)$		$E(A_1, \dots, A_n)$
➤ tulajdonságok	↔	attribútumok
➤ (szuper)kulcs	↔	(szuper)kulcs
➤ egyedhalmaz előfordulása	↔	reláció
➤ e egyed	↔	$(e(A_1), \dots, e(A_n))$ sor
➤ $R(E_1, \dots, E_p, A_1, \dots, A_q)$	↔	$R(K_1, \dots, K_p, A_1, \dots, A_q)$
kapcsolati séma, ahol		relációséma, ahol
E_i egyedhalmaz,	↔	K_i az E_i (szuper)kulcsa
A_j saját tulajdonság		
E/K modell	↔	Relációs adatmodell

Példa: E/K diagram átírása relációkká

Az egyedek és a kapcsolatok átírása:



A kapcsolatoknak megfelelő sémákban az oszlopokat átneveztük, például a szerepek alapján, (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.
- Hasonlóan a Dolgozók táblába összevonható az Osztállyal illetve önmagával való kapcsolat

Példa – Sörivók adatbázisséma

- Az előadások SQL lekérdezései az alábbi Sörivók adatbázissémán alapulnak
(aláhúzás jelöli a kulcs attribútumokat)

Sörök(név, gyártó)

Sörözők(név, város, tulaj, engedély)

Sörivók(név, város, tel)

Szeret(név, sör)

Felszolgál(söröző, sör, ár)

Látogat(név, söröző)

Táblák és megszorítások megadása

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

2.3.(folyt) és 7.1. Több táblára
vonatkozó megszorítások
megadása (SQL DDL)



Ismétlés: relációsémák definiálása

- Az SQL tartalmaz **adateleíró 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
 - Nézet táblák [CREATE [OR REPLACE] | DROP] VIEW
 - Átmeneti munkatáblák (WITH záradéka a SELECT-nek)
- **Alaptáblák** megadása: **CREATE TABLE**

Tankönyv példa: Filmek séma

Filmek(

cím:string,
év:integer,
hossz:integer,
műfaj:string,
stúdióNév:string,
producerAzon:integer)

FilmSzínész(

név:string,
cím:string,
nem:char,
születésiDátum:date)

Stúdió(

név:string,
cím:string,
elnökAzon:integer)

Mit jelentenek az aláhúzások?

Tankönyv példája, hibás fordítás:

title=(film)cím és address=(lak)cím

Tervezéssel később foglalkozunk, ez a példa hibás, az elnevezések, de így jó lesz, hogy a lekérdezéseknél megnézzük hogyan kezeljük.

SzerepelBenne(

filmCím:string,
filmÉv:integer,
színészNév:string)

GyártásIrányító(

név:string,
cím:string,
azonosító:integer,
nettóBevétel:integer)

Példa megszorításokra: Kulcs

- Előző példában: attribútumok aláhúzása mit jelent?
- Filmek: elvárjuk, hogy ne legyen a megengedett előfordulásokban két különböző sor, amelyek megegyeznek cím, év attribútumokon.
- Egyszerű kulcs egy attribútumból áll, de egy kulcs nem feltétlenül áll egy attribútumból, ez az összetett kulcs. Például a **Filmek** táblában a cím és év együtt alkotják a kulcsot, nem elég a cím, ugyanis van például (King Kong, 1933), (King Kong, 1976) és (King Kong, 2005).
- A kulcsot aláhúzás jelöli: **Filmek (cím, év, hossz, ...)**

Kulcsra vonatkozó megszorítások

- Az attribútumok egy halmaza egy **kulcsot** alkot egy relációra nézve, ha a reláció **bármely előfordulásában** nincs két olyan sor, amelyek a kulcs összes attribútumának értékein megegyeznének.

- Formális megadása:

$$R(U), X \subseteq U, U = \{A_1, \dots, A_n\}, X = \{A_{j_1}, \dots, A_{j_k}\}$$

$$t \in R, t[X] = \langle A_{j_1} : t(A_{j_1}), \dots, A_{j_k} : t(A_{j_k}) \rangle$$

ezzel a jelöléssel mit jelent, hogy X kulcs elvárás?

ha $t_1 \in R, t_2 \in R$ és $t_1[X] = t_2[X]$ akkor $t_1 = t_2$

Kulcs megadása

- **PRIMARY KEY** vagy **UNIQUE**
- Nincs a relációnak két olyan sora, amely a lista minden attribútumán megegyezne.
- Kulcs esetén nincs értelme a DEFAULT értéknek.
- Kulcsok megadásának két változata van:
 - Egyszerű kulcs (egy attribútum) vagy
 - Összetett kulcs (attribútumok listája)

Kulcs megadása

- **PRIMARY KEY** vagy **UNIQUE**
- Nincs a relációnak két olyan sora, amely a lista minden attribútumán megegyezne.
- Kulcs esetén nincs értelme a DEFAULT értéknek.
- Kulcsok megadásának két változata van:
 - Egyszerű kulcs (egy attribútum) vagy
 - Összetett kulcs (attribútumok listája)

Egyszerű kulcs megadása

- Ha a kulcs egyetlen attribútum, akkor ez az attribútum deklarációban megadható

<attribútumnév> <típus> **PRIMARY KEY**

vagy <attribútumnév> <típus> **UNIQUE**

- Példa:

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

Összetett kulcs megadása

- Ha a kulcs több attribútumból áll, akkor a CREATE TABLE utasításban az attribútum deklaráció után a **kiegészítő részben** meg lehet adni további tábla elemeket: **PRIMARY KEY (attrnév₁, ... attrnév_k)**
- Példa:

```
CREATE TABLE Felszolgal (
    söröző      CHAR(20) ,
    sör         VARCHAR2(20) ,
    ár          NUMBER(10,2) ,
    PRIMARY KEY (söröző, sör)
);
```

PRIMARY KEY vs. UNIQUE

- Csak egyetlen **PRIMARY KEY** lehet a relációban, viszont **UNIQUE** több is lehet.
- PRIMARY KEY egyik attribútuma sem lehet **NULL érték** egyik sorban sem. Viszont UNIQUE-nak deklarált attribútum lehet NULL értékű, vagyis a táblának lehet olyan sora, ahol a UNIQUE attribútum értéke **NULL** vagyis **hiányzó érték**.
- az SQL lekérdezésnél adjuk meg hogyan kell ezzel a speciális értékkel gazdálkodni, hogyan lehet NULL-t kifejezésekben és hogyan lehet feltételekben használni
- Következő héten visszatérünk a megszorítások és a hivatkozási épség megadására.

Egyszerű kulcs megadása

- Ha a kulcs egyetlen attribútum, akkor ez az attribútum deklarációban megadható

<attribútumnév> <típus> **PRIMARY KEY**

vagy <attribútumnév> <típus> **UNIQUE**

- Példa:

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

Összetett kulcs megadása

- Ha a kulcs több attribútumból áll, akkor a CREATE TABLE utasításban az attribútum deklaráció után a kiegészítő részben meg lehet adni további tábla elemeket: **PRIMARY KEY (attrnév₁, ... attrnév_k)**
- Példa:

```
CREATE TABLE Felszolgal (
    söröző      CHAR(20) ,
    sör         VARCHAR2(20) ,
    ár          NUMBER(10,2) ,
    PRIMARY KEY (söröző, sör)
);
```

PRIMARY KEY vs. UNIQUE

- Csak egyetlen **PRIMARY KEY** lehet a relációban, viszont **UNIQUE** több is lehet.
- PRIMARY KEY egyik attribútuma sem lehet **NULL érték** egyik sorban sem. Viszont UNIQUE-nak deklarált attribútum lehet NULL értékű, vagyis a táblának lehet olyan sora, ahol a UNIQUE attribútum értéke **NULL** vagyis **hiányzó érték**.
- az SQL lekérdezésnél adjuk meg hogyan kell ezzel a speciális értékkel gazdálkodni, hogyan lehet NULL-t kifejezésekben és hogyan lehet feltételekben használni
- Következő héten visszatérünk a megszorítások és a hivatkozási épség megadására.

Idegen kulcsok megadása

- Az első előadáson a táblák létrehozásához veszünk 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 Felszolgal (  
    söröző       CHAR(20) ,  
    sör          CHAR(20) ,  
    ár           REAL ,  
    FOREIGN KEY (sör) REFERENCES Sörök(név) ) ;
```

Lekérdezések: Relációs algebra

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



2.4. Relációs algebra, mint
lekérdező nyelv (később
ebben tudjuk értelmezni
az SQL SELECT utasítást)

Lekérdezések: Mi az algebra?

- Nyelv: a kérdés szintaktikai alakja és a kérdés kiértékelése (algorithmus) kiértékelési szemantika
- Algebra **műveleteket** és **atomi operandusokat** tartalmaz.
- **Relációs algebra**: az atomi operandusokon és az algebrai kifejezéseken végzett műveletek alkalmazásával kapott relációkon műveleteket adunk meg, kifejezéseket építünk (a kifejezés felel meg a kérdés szintaktikai alakjának).
- Fontos tehát, hogy **minden művelet végeredménye reláció**, amelyen további műveletek adhatók meg.
- A relációs algebra atomi operandusai a következők:
 - a relációkhoz tartozó **változók**,
 - **konstansok**, amelyek véges relációt fejeznek ki.

Relációs algebrai lekérdező nyelv ---1

Relációs algebrai kifejezés, mint lekérdező nyelv

Lekérdező nyelv: L -nyelv

Adott az adatbázis sémája: $\mathbb{R} = \{R_1, \dots, R_k\}$

$q \in L$ $q: R_1, \dots, R_k \rightarrow V$ (eredmény-reláció)

E - relációs algebrai kifejezés: $E(R_1, \dots, R_k) = V$ (output)

Relációs algebrai kifejezések formális felépítése

➤ Elemi kifejezések (alapkifejezések)

(i) $R_i \in \mathbb{R}$ (az adatbázis-sémában levő relációnevek)

R_i kiértékelése: az aktuális előfordulása

(ii) konstans reláció (véges sok, konstansból álló sor)

➤ Összetett kifejezések (folyt. köv.oldalon)

Relációs algebrai lekérdező nyelv ---2

(folyt.) Relációs algebrai kifejezések felépítése

- Összetett kifejezések
- Ha E_1, E_2 kifejezések, akkor a következő E is kifejezés
 - $E := \Pi_{\text{lista}} (E_1)$ vetítés (típus a lista szerint)
 - $E := \sigma_{\text{Feltétel}} (E_1)$ kiválasztás (típus nem változik)
 - $E := E_1 \cup E_2$ unió, ha azonos típusúak (és ez a típusa)
 - $E := E_1 - E_2$ különbség, ha E_1, E_2 azonos típusúak (típus)
 - $E := E_1 \bowtie E_2$ term. összekapcsolás (típus attr-ok uniója)
 - $E := \rho_{S(B_1, \dots, B_k)} (E_1 (A_1, \dots, A_k))$ átnevezés (típ.új attr.nevek)
 - $E := (E_1)$ kifejezést zárójelezve is kifejezést kapunk
- Ezek és csak ezek a kifejezések, amit így meg tudunk adni

Vetítés (project, jelölése pí: Π)

- **Vetítés** (projekció). Adott relációt vetít le az alsó indexben szereplő attribútumokra (attribútumok számát csökkentik)
- $\Pi_{\text{lista}}(R)$ ahol lista: $\{A_{i_1}, \dots, A_{i_k}\}$ R-sémájában levő attribútumok egy részhalmazának felsorolása
eredmény típusa $\langle A_{i_1} : \text{értéktípus}_{i_1}, \dots, A_{i_k} : \text{értéktípus}_{i_k} \rangle$
 $\Pi_{\text{lista}}(R) := \{ t.A_{i_1}, t.A_{i_2}, \dots, t.A_{i_k} \mid t \in R \} = \{ t[\text{lista}] \mid t \in R \}$
- Reláció soraiból kiválasztja az attribútumoknak megfelelő A_{i_1}, \dots, A_{i_k} -n előforduló értékeket, ha többször előfordul akkor a duplikátumokat kiszűrjük (hogy halmazt kapjunk)

➤ **Példa:**

A	B	C
a	b	c
c	d	e
c	d	d

$\Pi_{A, B}(R)$



A	B
a	b
c	d

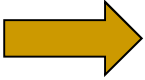
Kiválasztás (select, jelölése szigma: σ)

- **Kiválasztás** (szűrés). Kiválasztja az argumentumban szereplő reláció azon sorait, amelyek eleget tesznek az alsó indexben szereplő feltételnek.
- $\sigma_{\text{Feltétel}}(R)$ és R sémája megegyezik
- $\sigma_{\text{Feltétel}}(R) := \{ t \mid t \in R \text{ és } t \text{ kielégíti az } F \text{ feltételt} \}$
- $R(A_1, \dots, A_n)$ séma feletti reláció esetén a σ_F kiválasztás F feltétele a következőképpen épül fel:
 - **elemi feltétel**: $A_i \theta A_j$, $A_i \theta c$, ahol c konstans, θ pedig $=, \neq, <, >, \leq, \geq$
 - **összetett feltétel**: ha B_1, B_2 feltételek, akkor $\neg B_1$, $B_1 \wedge B_2$, $B_1 \vee B_2$ és zárójelezésekkel is feltételek

➤ **Példa:**

A	B	C
a	b	c
c	d	e
g	a	d

$\sigma_{A=a \vee C=d}(R)$



A	B	C
a	b	c
g	a	d

Halmazműveletek (jelölése a szokásos)

- Reláció előfordulás véges sok sorból álló halmaz. Így értelmezhetők a szokásos halmazműveletek: az **unió** (az eredmény halmaz, csak egyszer szerepel egy sor) értelmezhető a **metszet** és a **különbség**. Milyen művelet van még halmazokon? Értelmezhető-e relációkon?
- R, S és azonos típusú, $R \cup S$ és $R - S$ típusa ugyanez
 $R \cup S := \{t \mid t \in R \vee t \in S\}$, $R - S := \{t \mid t \in R \wedge t \notin S\}$
- Az alpműveletekhez az **unió** és **különbség** tartozik, **metszet** műveletet származtatjuk $R \cap S = R - (R - S)$

➤

A	B	C
a	b	c
c	d	e
g	a	d

A	B	C
a	b	c
c	d	e
g	d	f

Példa: különbségre

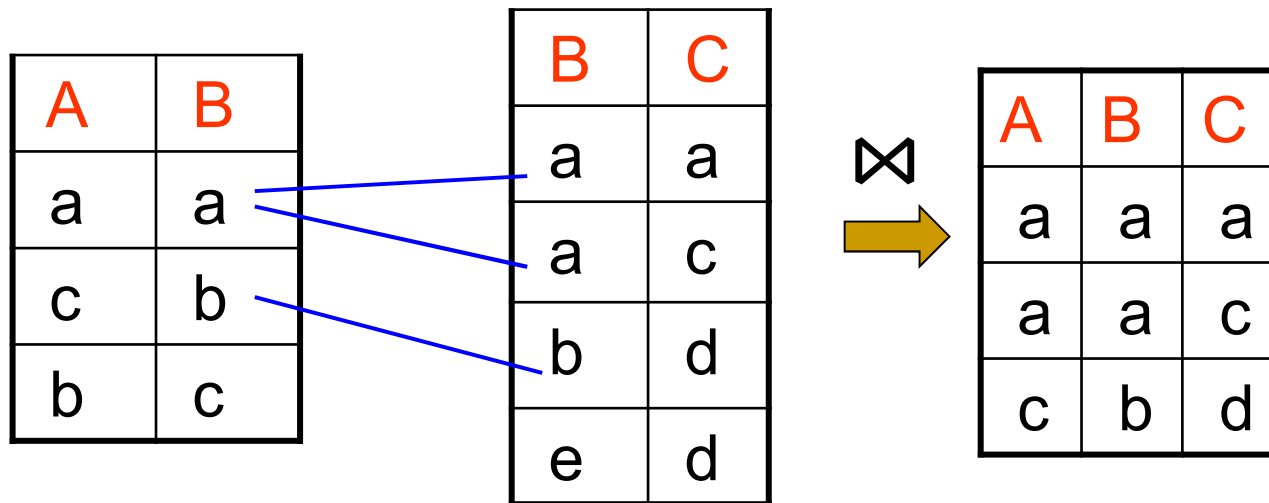
$R - S$

➔

A	B	C
g	a	d

Természetes összekapcsolás ---1

- Szorzás jellegű műveletek (attribútumok számát növeli) többféle lehetőség, amelyekből csak egyik alpművelet:
- Angolul: Natural Join (jelölése: „csokornyakkendő”)
- **Természetes összekapcsolás**: közös attribútum-nevekre épül. $R \bowtie S$ azon sorpárokat tartalmazza R-ből illetve S-ből, amelyek R és S azonos attribútumain megegyeznek.



Természetes összekapcsolás ---2

- **Természetes összekapcsolás:**
- Legyen $R(A_1, \dots, A_k, B_1, \dots, B_n)$, illetve $S(B_1, \dots, B_n, C_1, \dots, C_m)$
- $R \bowtie S$ típusa $(A_1, \dots, A_k, B_1, \dots, B_n, C_1, \dots, C_m)$ vagyis a két attribútum-halmaz uniója
- $R \bowtie S = \{ \langle A_1: t(A_1), \dots, A_k: t(A_k), B_1: t(B_1), \dots, B_n: t(B_n), C_1: s(C_1), \dots, C_m: s(C_m) \rangle \mid t \in R, s \in S, t(B_i) = s(B_i) \ i=1, \dots, n \}$
- $R \bowtie S$ elemei $v \in R \bowtie S$
$$R \bowtie S = \{ v \mid \exists t \in R, \exists s \in S: t[B_1, \dots, B_n] = s[B_1, \dots, B_n] \wedge v[A_1, \dots, A_k] = t[A_1, \dots, A_k] \wedge v[B_1, \dots, B_n] = t[B_1, \dots, B_n] \wedge v[C_1, \dots, C_m] = s[C_1, \dots, C_m] \}$$

Természetes összekapcsolás ---3

- **Példákban:** két azonos nevű attribútumot úgy tekintünk, hogy ugyanazt jelenti és a közös érték alapján fűzzük össze a sorokat.
- **Milyen problémák lehetnek?**
- Filmek adatbázisban ugyanarra a tulajdonságra más névvel hivatkozunk: Filmek.év és SzerepelBenne.filmÉv, illetve FilmSzínész.név és SzerepelBenne.színészNév
- Termékek adatbázisban pedig ugyanaz az azonosító mást jelent: Termék.típus más, mint Nyomtató.típus
- Emiatt a Filmek és a Termékek adatbázisokban ahhoz, hogy jól működjön az összekapcsolás **szükségünk van** egy technikai műveletre, és ez: **az átnevezés (rename)**

Átnevezés (rename, jelölése ró: ρ)

- Miért van erre szükség? Nem tudjuk a reláció saját magával való szorzatát kifejezni, $R \bowtie R = R$ lesz.
- Láttuk, hogy egyes esetekben szükség lehet relációnak vagy a reláció attribútumainak **átnevezésére**:

$$\rho_{T(B_1, \dots, B_k)}(R(A_1, \dots, A_k))$$

- Ha az attribútumokat nem szeretnénk átnevezni, csak a relációt, ezt $\rho_T(R)$ -rel jelöljük. Ha ugyanazt a táblát használjuk többször, akkor a táblának adunk másik hivatkozási (alias) nevet.
- Az attribútumok átnevezése helyett alternatíva: $R.A$ (vagyis relációnév.attribútumnév hivatkozás) amivel meg tudjuk különböztetni a különböző táblákból származó azonos nevű attribútumokat.

Szorzás jellegű műveletek ---1

- Szorzás jellegű műveletek többféle lehetősége közül csak az egyiket vesszük alpműveletnek: **join vagy természetes összekapcsolást**, amely közös attribútumnevekre épül.
 $R \bowtie S$ azon sorpárokat tartalmazza R-ből illetve S-ből, amelyek R és S azonos attribútumain megegyeznek.
- Egy másik lehetőség: **direkt-szorzat (Descartes-szorzat)**
Ez is tekinthető alpműveletnek (és bizonyos esetekben egyszerűbb ezt venni alpműveletnek) az ennél sokkal gyakrabban használt **természetes összekapcsolás** helyett.
- $R \times S$: az R és S minden sora párban összefűződik, az első tábla minden sorához hozzáfűzzük a második tábla minden sorát

$$R \times S := \{ t \mid t[R] \in R \text{ és } t[S] \in S \}$$

Szorzás jellegű műveletek ---2

- A **direkt-szorzat** (vagy szorzat, **Descartes-szorzat**) esetén természetesen nem fontos az attribútumok egyenlősége. A két vagy több reláció azonos nevű attribútumait azonban meg kell különböztetni egymástól. Hivatkozás séma: oszlopok átnevezése illetve azonos nevű oszlop esetén: $R.A_1, \dots, R.A_k, S.A_1, \dots, S.A_k$

- **Példa:**

A	B	C
a	b	c
c	d	e
g	a	d

B	D
b	r
q	s



$R \times S$

A	R.B	C	S.B	D
a	b	c	b	r
a	b	c	q	s
c	d	e	b	r
c	d	e	q	s
g	a	d	b	r
g	a	d	q	s

Szorzás jellegű műveletek ---3

- Ha R, S sémái megegyeznek, akkor $R \bowtie S = R \cap S$.
- Ha R, S sémáiban nincs közös attribútum, akkor $R \bowtie S = R \times S$.
- Később nézünk még további szorzás jellegű műveletet:
Théta összekapcsolás \bowtie_{θ} , félig összekapcsolás \ltimes , és a rel.algebra kiterjesztésénél külső összekapcsolásokat.
- Hogyan fejezhető ki az $R \times S$ **direkt szorzat** relációs algebrában? (ha a **természetes összekapcsolást** tekintjük alpműveletnek, ebből és az átnevezés segítségével felírható a direkt szorzat).
- Hogyan fejezhető ki a **természetes összekapcsolás**, ha a **direkt szorzatot** soroljuk az alpműveletek közé?

Lekérdezések kifejezése algebrában

- Következő előadáson folytatjuk (példák)

Több táblára vonatkozó lekérdezések

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

6.2. Több táblára vonatkozó
lekérdezések az SQL-ben,
lekérdezések értelmezése



Select-From-Where (SFW) utasítás

- Gyakran előforduló relációs algebrai kifejezés
 $\Pi_{\text{Lista}} (\sigma_{\text{Felt}} (R_1 \times \dots \times R_n))$ típusú kifejezések
- Szorzat és összekapcsolás az SQL-ben
- SELECT s-lista -- milyen típusú sort szeretnénk az eredményben látni?
FROM f-lista -- relációk (táblák) összekapcsolása, illetve szorzata
WHERE felt -- milyen feltételeknek eleget tevő sorokat kell kiválasztani?
- FROM f-lista elemei (ezek ismétlődhetnek)
táblanév [[AS] sorváltozó, ...]
Itt: a from lista elemei a táblák direkt szorzatát jelenti, az összekapcsolási feltételt where-ben adjuk meg, később bevezetünk majd tovább lehetőségeket a különböző összekapcsolásokra az SQL from záradékában.

Attribútumok megkülönböztetése ---1

- Milyen problémák merülnek fel?
- (1) Ha egy attribútumnév több sémában is előfordul, akkor nem elég az attribútumnév használata, mert ekkor nem tudjuk, hogy melyik sémához tartozik.
- Ezt a problémát az SQL úgy oldja meg, hogy megengedi egy relációnévnek és egy pontnak a használatát egy attribútum előtt: **R.A** (az R reláció A attribútumát jelenti).
- **Természetes összekapcsolás** legyen $R(A, B), S(B, C)$
SELECT A, R.B B, C
FROM R, S
WHERE R.B=S.B;

Attribútumok megkülönböztetése ---2

- Milyen problémák merülnek még fel?
- (2) Ugyanaz a reláció többször is szerepelhet, vagyis szükség lehet arra, hogy ugyanaz a relációnév többször is előforduljon a FROM listában.
- Ekkor a FROM listában a táblához másodnevet kell megadni, erre **sorváltóként** is szoktak hivatkozni, megadjuk, h. melyik sorváltó melyik relációt képviseli:
FROM $R_1 [t_1], \dots, R_n [t_n]$
Ekkor a SELECT és WHERE záradékok kifejezésekben a hivatkozás: **$t_i.A$** (vagyis sorváltó.attribútumnév)

SFW szabvány alapértelmezése ---1

- Kiindulunk a **FROM záradékból**: a FROM lista minden eleméhez **egy beágyazott ciklus**, végigfut az adott tábla sorain a ciklus minden lépésénél az n darab sorváltozónak lesz egy-egy értéke
- ehhez kiértékeljük a WHERE feltételt, vagyis elvégezzük a **WHERE záradékban** szereplő feltételnek eleget tevő sorok kiválasztását (csak a helyesek, ahol TRUE=igaz választ kapunk), azok a sorok kerülnek az eredménybe.
- Alkalmazzuk a **SELECT záradékban** jelölt kiterjesztett projekciót. Az **SQL-ben az eredmény alapértelmezés szerint** itt sem halmaz, hanem **multihalmaz**.

Ahhoz, hogy halmazt kapjunk, azt külön kérni kell:
SELECT DISTINCT Lista

SFW szabvány alapértelmezése ---2

FOR t_1 sorra az R_1 relációban DO

FOR t_2 sorra az R_2 relációban DO

...

FOR t_n sorra az R_n relációban DO

IF a where záradék igaz, amikor az attribútumokban
 t_1, t_2, \dots, t_n megfelelő értékei találhatóak

THEN

t_1, t_2, \dots, t_n -nek megfelelően kiértékeljük a
select záradék attribútumait
és az értékekből alkotott sort
az eredményhez adjuk

SFW szabvány alapértelmezése ---3

SELECT [DISTINCT] kif₁ [[AS] onév₁], ..., kif_n [[AS] onév_n]
FROM R₁ [t₁], ..., R_n [t_n]
WHERE feltétel (vagyis logikai kifejezés)

Alapértelmezés (a műveletek szemantikája -- általában)

- A FROM záradékban levő relációkhoz tekintünk egy-egy **sorváltozót**, amelyek a megfelelő reláció minden sorát bejárják (beágyazott ciklusban)
- Minden egyes „aktuális” sorhoz kiértékeljük a WHERE záradékot
- Ha helyes (vagyis igaz) választ kaptunk, akkor képezünk egy sort a SELECT záradékban szereplő kifejezéseknek megfelelően.

Megj.: konverzió relációs algebrába

SELECT [DISTINCT] kif₁ [[AS] onév₁], ..., kif_n [[AS] onév_n]
FROM R₁ [t₁], ..., R_n [t_n]

WHERE feltétel (vagyis logikai kifejezés)

- 1.) A FROM záradék sorváltozóiból indulunk ki, és tekintjük a hozzájuk tartozó relációk Descartes-szorzatát. Átnevezéssel valamint R.A jelöléssel elérjük, hogy minden attribútumnak egyedi neve legyen.
- 2.) A WHERE záradékot átalakítjuk egy kiválasztási feltétellé, melyet alkalmazunk az elkészített szorzatra.
- 3.) Végül a SELECT záradék alapján létrehozuk a kifejezések listáját, a (kiterjesztett) vetítési művelethez.

$$\Pi_{\text{onév1}, \dots, \text{onévn}} (\sigma_{\text{feltétel}} (R_1 \times \dots \times R_n))$$

Példa: Két tábla összekapcsolása ---1

- Mely söröket szeretik a Joe's Bárba járó sörivók?

SELECT sör

FROM Szeret, Látogat

WHERE bár = 'Joe' 's Bar'

AND Látogat.név = Szeret.név;

- Kiválasztási feltétel: **bár = 'Joe' 's Bar'**
- Összekapcsolási feltétel: **Látogat.név = Szeret.név**
- Alapértelmezése a következő oldalon a mai órán
- Összekapcsolások szintaxisát később nézzük majd

Példa: Két tábla összekapcsolása ---2

Látogat

t1

név	bár
Sally	Joe's

Szeret

t2

név	sör
Sally	Bud

Ellenőrzés
Joe's bárja

Ellenőrizzük, hogy
megegyeznek-e

output

Tábla önmagával való szorzata ---1

- Bizonyos lekérdezéseknél arra van szükségünk, hogy ugyanannak a relációnak több példányát vegyük.
- Ahhoz, hogy meg tudjuk különböztetni a példányokat a relációkat átnevezzük, másodnevet adunk, vagyis **sorváltozókat** írunk mellé a FROM záradékban.
- A relációkat mindig átnevezhetjük ily módon, akkor is, ha egyébként nincs rá szükség (csak kényelmesebb).
- **Példa: R(Szülő, Gyerek)** séma feletti relációban adott szülő-gyerek adatpárokból állítsuk elő a megállapítható Nagyszülő-Unoka párokat!

```
SELECT t1.Szülő NagySzülő, t2.Gyerek Unoka  
FROM R t1, R t2  
WHERE t1.Gyerek = t2.Szülő;
```

Tábla önmagával való szorzata ---2

- **Példa: Sörök(név, gyártó)** tábla felhasználásával keressük meg az összes olyan sörpárt, amelyeknek ugyanaz a gyártója.
- Ne állítsunk elő (Bud, Bud) sörpárokat.
- A sörpárokat ábécé sorrendben képezzük, például ha (Bud, Miller) szerepel az eredményben, akkor (Miller, Bud) ne szerepeljen.

```
SELECT s1.név, s2.név
FROM Sörök s1, Sörök s2
WHERE s1.gyártó = s2.gyártó
      AND s1.név < s2.név;
```

Halmazműveletek az SQL-ben

- Mi hiányzik még, hogy a relációs algebra alpműveleteit mindet az SQL-ben vissza tudjuk adni?
- A relációs algebrai halmazműveletek: **unió, különbség** mellett az **SQL-ben ide soroljuk a metszetet is** (ugyanis fontos a metszet és az SQL-ben is implementálva van).
- Az SQL-ben a halmazműveleteket úgy vezették be, hogy azt mindig két lekérdezés között lehet értelmezni, vagyis nem relációk között, mint $R \cup S$, hanem lekérdezem az egyiket is és a másikat is, majd a lekérdezések unióját veszem.

(SFW-lekérdezés1)

[**UNION** | **INTERSECT** | {**EXCEPT** | **MINUS**}]

(SFW-lekérdezés2);

Példa: Intersect (metszet)

- Szeret(név, sör), Felszolgál(bár, sör, ár) és Látogat(név, bár) táblák felhasználásával keressük

Trükk: itt ez az alkérdés valójában az adatbázisban tárolt tábla azokat a sörivőket és söröket, amelyekre a sörivő szereti az adott sört **és** a sörivő látogat olyan bárt, ahol felszolgálják a sört.

(SELECT * FROM Szeret)

INTERSECT

(SELECT név, sör

FROM Felszolgál, Látogat

WHERE Látogat.bár = Felszolgál.bár);

(név, sör) párok, ahol a sörivő látogat olyan bárt, ahol ezt a sört felszolgálják

Kérdés/Válasz

- Köszönöm a figyelmet! Kérdés/Válasz?
- A 2.gyakorlat az 1.előadáshoz kapcsolódik, egy táblára vonatkozó lekérdezések: az Oracle SQL leggyakrabban használt sorfüggvényeit nézzük meg az egyszerű egy táblára vonatkozó lekérdezések SELECT és WHERE záradékaiban, lásd Oracle 3.lecke sorfüggvények
- Numerikus, karakteres, dátum, konverziós, általános, például NULL értéket megadott értékkel helyettesítő NVL és COALESCE sorfüggvényeket