

## Adatbázisok elmélete 4. előadás

Katona Gyula Y.  
Budapesti Műszaki és Gazdaságtudományi Egyetem  
Számítástudományi Tsz.  
I. B. 137/b  
kiskat@cs.bme.hu  
<http://www.cs.bme.hu/~kiskat>

2005

### 3. Hivatkozási épség:

lehet a rajzon jelezni, ha egy kapcsolatnál azt szeretnénk, hogy pontosan egy egyed tartozzon egy kiválasztott egyedhez. Ilyenkor kerek nyilat használunk:



Ebben az esetben minden filmhez pontosan egy stúdiónak kell tartoznia.

### 4. Értelmezési tartományra vonatkozó megkötések és egyéb megszorítások:

Értelmezési tartomány: típusal.

Egyéb: kapcsolat fokát lehet itt is korlátozni, pl:

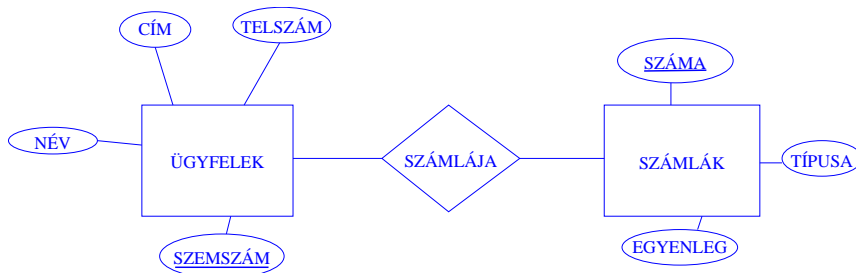


Ekkor egy filmhez 10-nél kevesebb színészt rendelünk.

## Megszorítások E/K modellben

### 1. Kulcsok:

egy kulcsot aláhúzással jelölünk (a kulcsba tartozó attribútumokat aláhúzzuk), a többi kulcsot az ábrán nem lehet jelölni, ezeket szövegesen mellékeljük.



### 2. Egyértékűség:

- egyszerű attribútumok használata  $\implies$  minden attribútum egyértékű az E/K modellben (általában lehet NULL-érték is, ha mégsem, akkor írásban jelezhető)
- kapcsolatnál: nyilakkal jelezhető, ha valamerre "egy" a kapcsolat

## Gyenge egyedhalmazok

Az E/K modell sajátossága. Egy egyedhalmaz akkor gyenge egyedhalmaz, ha az egyedeit nem azonosítják az attribútumai, csak a kapcsolatokkal együtt. (ODL-nél nincs ez a dolog, mert ott az egyedi OID mindig azonosít.)

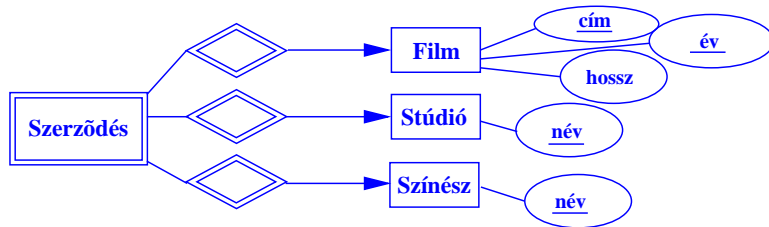
Jelölés: dupla téglalap az egyedhalmaznak és dupla rombusz azoknak a kapcsolatoknak, amiken keresztül megy az azonosítás.

A gyenge egyedhalmaznál az aláhúzott attribútumok belekerülnek a gyenge egyedhalmaz kulcsába, de még más attribútumok is hozzájönnek ehhez: azok, amik a duplarombuszos kapcsolat(ok) végén álló egyedhalmaz(ok) kulcsai.

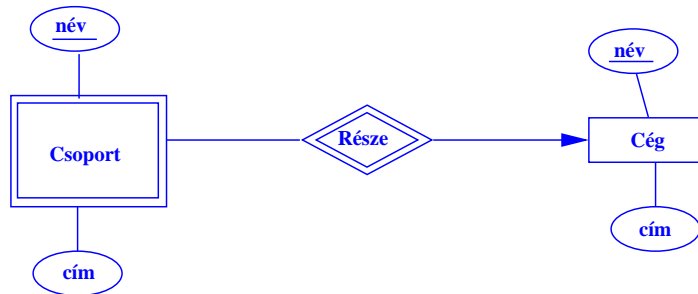
## Példák:

- Amikor a többágú kapcsolatot binárisra írtuk át, akkor olyan egyedhalmaz keletkezik (a kapcsolatból), aminek általában nincs is attribútuma, ezért ennek az egyedhalmaznak az egyedeit csak a kapcsolatokon át lehet azonosítani.

A filmes példa esetén a Szerződés egyedhalmaz egyedeit a kapcsolódó egyedhalmazok (Film, Színész, Stúdió) kulcsattribútumai azonosítják: film címe, gyártási éve, színész neve, stúdió neve. Ha ezek adottak, akkor már csak egy szerződés lehet, ami ezekre vonatkozik.



- Ebben a példában a csoport neve még önmagában nem kulcs (sok cégnél lehet pl. HR csoport), sőt a címmel együtt sem feltétlenül azonosít egy csoportot, de ha a kapcsolaton keresztül a céget is bevesszük az azonosításba, úgy már egyértelmű lesz, hogy melyik csoportról beszélünk.



## Követelmények az azonosító kapcsolatra

A gyenge egyedhalmaz kulcsában benne lehetnek saját attribútumai (mint az előbb a Csoport neve) és biztosan vannak benne olyan attribútumok, amiket duplarombuszos kapcsolatokon keresztül szerez.

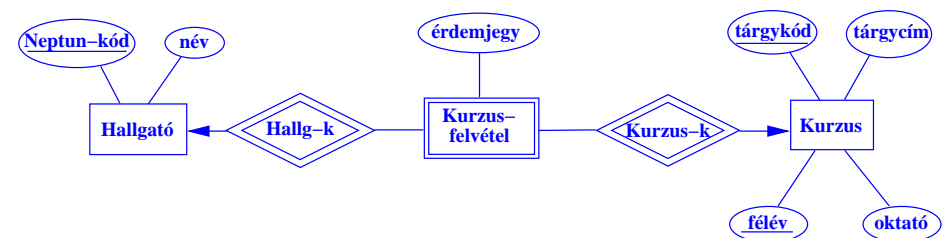
Követelmények ezekre a kapcsolatokra:

- Ha az  $E$  gyenge egyedhalmaz kulcsattribútumot szerez egy  $F$  egyedhalmaztól az  $R$  kapcsolaton át, akkor  $R$  legyen több-egy  $E$ -ből  $F$ -be. (Így egy  $E$ -belihez egyértelműen tartozik egy  $F$ -beli).
- Egy attribútum pontosan akkor kerül bele az  $E$  gyenge egyedhalmaz kulcsába, ha benne van az  $F$  egyedhalmaz kulcsában is.

Megjegyzés: természetesen  $F$  is lehet gyenge egyedhalmaz.

## Példa

Tervezzék E/K diagrammot egy egyetemi nyilvántartáshoz, ahol hallgatókat és az általuk szerzett jegyeket tartjuk nyilván. Vegyünk három egyedhalmazt: hallgató, kurzus, kurzusfelvétel (ez utóbbi kapcsoló egyedhalmaz a hallgatók és kurzusok között, ennél reprezentáljuk a kapott érdemjegyet is). Adjuk meg ezt E/K diagrammal, jelöljük a gyenge egyedhalmazokat és a kulcsokat is.



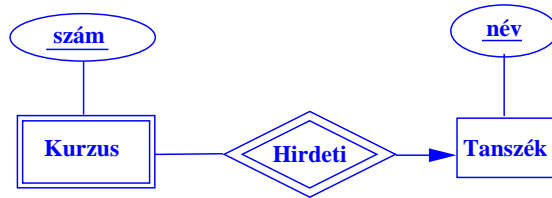
Döntsük el, hogy az érdemjegy része-e a kurzusfelvételt reprezentáló egyedhalmaz kulcsának?

Az érdemjegy nem része a kurzusfelvétel egyedhalmaz kulcsának, ezen egyedhalmaz kulcsa a két kapcsolaton keresztül jön: a hallgatótól a neptun-kód, a tárgytól meg a tárgykód és a félév.

## Példa

Tervezzon E/K diagrammot a következőre és jelölje a rajzon a kulcsokat és a gyenge egyedhalmazokat:

Egyedhalmazok: **Kurzusok**, **Tanszékek**. Egy kurzust egy tanszék hirdet meg, de azt csak egy számmal azonosítja. Különböző tanszékek adhatják ugyanazt a számot a kurzusoknak, de egy tanszék tárgyai mind különböző számot kapnak.



## Miért vannak gyenge egyedhalmazok?

- Maguktól keletkeznek, amikor többágú kapcsolatot írunk át binárisá.
- A redundancia elkerülése céljából. (Minek a cég nevét minden csoportnál külön felvenni, elég ha egyszer felírjuk és a kapcsolatból derítjük ki.)

A redundancia elkerülése nem csak az E/K modellben fontos, ez minden megközelítésben lényeges, hisz a redundancia bajok forrása.

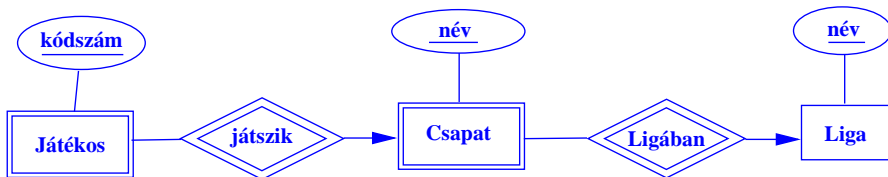
- Nehéz konzisztens állapotban tartani a DB-t, ha ugyanaz az infó ezer helyen van beírva.
- Nem lesz elég egyszerű a séma, nehéz lesz átlátni, hogy mi az ami ugyanaz, csak sokszor tároljuk és mi valóban más infó.
- Helyprobléma (ez egyre kevésbé van).

Ezek miatt törekszünk a redundancia kiküszöbölésére, de persze nem kell mindent kiirtani, hisz a világ is redundáns.

## Példa

Tervezzon E/K diagrammot a következőre és jelölje a rajzon a kulcsokat és a gyenge egyedhalmazokat:

Egyedhalmazok: **Ligák**, **Csapatok**, **Játékosok**. A Ligák nevei egyediek, a Csapatoké egy ligan belül különbözik, de különböző ligan belül lehetnek azonos nevű csapatok. Egy csapaton belül nincsenek azonos kódszámú játékosok, de különböző csapatokban lehetnek ilyenek.



## Tervezési alapelvek

1. **Valóságghű modellezés:** megragadni a lényeget, megfelelő adatelemeket választani, megfelelő kapcsolatokat (természetesek legyenek, de néha kellene mesterséges, technikai egyedhalmazok, osztályok is).
2. **Redundancia kerülése:** egyszerű mértékben. Ezt majd a relációs modell nagyon jól megoldja, de azért már a tervezéskor is jó erre figyelni.
3. **Egyszerűség:** csak az legyen a sémában, aminek lennie kell, minél egyszerűbb szerkezetben.
4. **Megfelelő (típusú, összetettségű) adatelemek választása:** jól döntsünk, hogy mi legyen attribútum, mi inkább kapcsolat, illetve esetleg külön osztály/egyedhalmaz. Az attribútumot egyszerűbb implementálni, de néha átláthatóbb egy külön egyedhalmaz.

Általános elvek:

- ha egy egyedhalmaznak csak egy attribútuma lenne  $\Rightarrow$  nem érdemes külön venni, ha összetettebb, akkor legyen külön.
- ha egy infót magában nem akarunk megőrizni, csak valamihez kapcsoltn  $\Rightarrow$  lehet csak attribútum (pl. ha a stúdiók csak annyiban érdekelnek minket, hogy melyik filmet ki gyártja, akkor nem kell külön Stúdió egyedhalmaz)

Ez mind a modellezéskor dől el, aszerint, hogy milyen sémát akarunk.

## Régebbi adatmodellek

- **Hálós adatmodell:** szemléletében hasonlít az objektumosra, de itt sokkal jobban közelíti a terv a fizikai megvalósítást (pl. az attribútumok megadásánál rögtön rendelkezünk a tárolás módjáról is). Lekérdezés, módosítás csak a tárolás pontos ismeretében lehetséges  $\Rightarrow$  nehezkesebb mint a relációs modell használata.
- **Hierarchikus adatmodell:** az első, korai rendszerek hierarchikussága miatt szervesen alakult ki. Akkor jó, ha az adatok, vagy a tárolás hierarchikus szerkezetű. Itt is ismerni kell a fizikai megvalósítást a kérdéshez/módosításhoz.

## Relációs adatmodell

Mit fogunk róla tanulni?

1. **elvi keret** (alapfogalmak, alapműveletek)
2. **konkrét nyelvek** (ISBL, QBE, QUELL, SQL, sémadefinícióra, adatmódosításra és lekérdezésre)
3. **tervezés** (minél jobb séma kialakítása, matematikai elmélet)

Egyetlen alapfogalom (nincs külön egyedhalmaz és kapcsolat): **reláció**.

## Relációs adatmodell

Jelenleg ez a legelterjedtebb, szinte minden DBMS ezen az elven működik.

Ennek okai:

- jól lehet benne modellezni, a modell után pedig könnyű a konkrét sémát megvalósítani
- nem kell ismerni a fizikai megvalósítást a lekérdezéshez, módosításhoz
- a logikai tervezésnek nagy, szép matematikai eszköztára van, ami segíti az egyszerű séma létrehozását

## A reláció definíciója

1. Gondolhatunk rá úgy, mint egy síkbeli táblázatra:

$R_1$	$A_1$	$A_2$
	1	y
	1	z
	3	y

$R_2$	$A_1$	$A_2$
	2	y
	1	z

Itt  $R_1$  a reláció neve,  $A_1$  és  $A_2$  az attribútumok nevei, a sorok pedig a reláció elemei. Az oszlopokban levő értékek az attribútumokhoz tartozó értékészletről kerülnek ki.

2. Tekinthejtük egy Descartes-szorzat részhalmazának is a relációt:

$A_1, A_2, \dots, A_n$  tetszőleges halmazok (attribútumok)

$$R \subseteq A_1 \times \dots \times A_n$$

⇒ Minden sor csak egyszer szerepel

⇒ a sorok sorrendje lényegtelen.

Példa:  $A_1 = \{1, 2, 3\}$ ,  $A_2 = \{x, y, z\}$

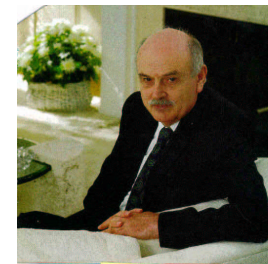
$R_1 = \{(1, y), (1, z), (3, z)\}$

$R_2 = \{(2, y), (1, z)\}$

De  $R$  elemeit tekinthejtük halmazoknak is, nem rendezett  $n$ -eseknek.

Ekkor az attribútumok sorrendje is mindegy.

## Relációs modell



Edgar F. Codd, (1932–)

1970-es cikk: *A Relational Model of Data for Large Shared Data Banks*

Teljes adatmodell: nem csak azt mondja meg hogyan írok le, hanem vannak műveletek is.

Ezeket a műveleteket relációkra alkalmazhatom és így újabb relációkat kapok majd.

3. Gondolhatunk egy relációra úgy is, mint függvények halmazára:

**Definíció.** Egy sor = egy függvény:  $s : \{\text{attribútumok}\} \rightarrow \{\text{attr. értékészlete}\}$

Egy  $R$  reláció ilyen függvények halmaza.

Így tényleg nem számít a sorrend, se a sorok között, se az attribútumok között.

Nincs két azonos sor.

Például:

$R_1$ -ben: 1. sor:  $A_1 \rightarrow 1$ ;  $A_2 \rightarrow y$ ;

Jelölés:

**Definíció.** Relációs séma:  $R(A_1, \dots, A_n)$ , ahol  $R$  a reláció neve, az  $A_i$ -k pedig az attribútumok nevei.

Például: Személy(Vezetéknév, Keresztnév, Neme, Végzettsége)

Gyakorlatban azért mégis rögzítünk egy sorrendet, azt, amelyikben felsoroljuk az attribútumokat.

## A relációs algebra alapműveletei

- Halmazműveletek (bármilyen halmazra mennének)
  - ★ unió:  $\cup$
  - ★ különbség:  $\setminus$
  - ★ szorzat:  $\times$
- Relációs műveletek (ezek már kihasználják, hogy itt relációról van szó)
  - ★ vetítés, projekció:  $\pi$
  - ★ kiválasztás, szelekció:  $\sigma$

Ezek mind tiszta műveletek: reláció  $\rightarrow$  reláció

⇒ gond nélkül egymásba ágyazhatók

## Műveletek

## Unió

- $R, S$  relációk  $\Rightarrow R \cup S =$  sorai vagy  $R$  vagy  $S$  sorai  
Azonos sorok csak egyszer szerepeljenek. (Gyakorlatban néha lehetnek azonos sorok.)
- csak akkor alkalmazható, ha  $R$  és  $S$  oszlopszáma egyenlő
- nem feltétlenül örököl típusokat vagy attribútum neveket
- Példa:

$R$	$A$	$B$	$S$	$A$	$C$	$R \cup S$	$A$	$(R \cup S)_2$
	$a$	$a$		$a$	$a$		$a$	$a$
	$a$	$c$		$a$	$d$		$a$	$c$
	$b$	$a$		$a$	$c$		$b$	$a$
				$b$	$b$		$a$	$d$
				$b$	$b$		$b$	$b$

## Műveletek

## Szorzat (direkt szorzat, Descartes szorzat)

- $R(A_1, \dots, A_k), S(B_1, \dots, B_l)$   $k$  ill.  $l$  attribútumos relációk  $\Rightarrow R \times S =$  egy  $k + l$  attribútumos reláció,  $R$  minden sora mögé odatesszük  $S$  minden sorát, minden lehetséges módon.  
Ha  $R$ -nek  $n$  sora van  $S$ -nek  $m$  sora  $\Rightarrow R \times S$ -nek  $nm$  sora van
- nincs kompatibilitási követelmény
- Az eredmény lényegében örökli  $R$  és  $S$  típusait és attribútum neveit, esetleg át kell nevezni.

## Műveletek

## Különbség

- $R, S$  relációk  $\Rightarrow R \setminus S = R$  azon sorai, amelyek  $S$ -ben nem szerepelnek
- nincs kompatibilitási követelmény (Ha pl. különböző az oszlopszám, nem szerepelhetnek azonos sorok úgysem. Ekkor  $R \setminus S = R$ )
- Az eredmény örökli  $R$  típusait és attribútum neveit (mert  $R \setminus S \subseteq R$ )
- Példa:

$R$	$A$	$B$	$S$	$A$	$C$	$R \setminus S$	$A$	$B$
	$a$	$a$		$a$	$a$		$b$	$a$
	$a$	$c$		$a$	$d$			
	$b$	$a$		$a$	$c$			
				$b$	$b$			

- Példa:

$R$	$A$	$B$	$S$	$A$	$C$	$R \times S$	$A$	$B$	$A'$	$C$
	$a$	$a$		$a$	$a$		$a$	$a$	$a$	$a$
	$a$	$c$		$a$	$d$		$a$	$a$	$a$	$d$
	$b$	$a$		$a$	$c$		$a$	$a$	$a$	$c$
				$b$	$b$		$a$	$a$	$b$	$b$
							$a$	$c$	$a$	$a$
							$a$	$c$	$a$	$d$
							$a$	$c$	$a$	$c$
							$a$	$c$	$b$	$b$
							$b$	$a$	$a$	$a$
							$b$	$a$	$a$	$d$
							$b$	$a$	$a$	$c$
							$b$	$a$	$b$	$b$

Az unió és különbség könnyű művelet, a szorzat nehezebb. Vigyázni kell mennyit használjuk.

## Műveletek

## Vetítés

- $R(A_1, \dots, A_l)$  alakú reláció  $\Rightarrow \pi_{A_{i_1}, \dots, A_{i_n}}(R)$   
 $R$  vetítése  $A_{i_1}, \dots, A_{i_n}$ -re (fontos a sorrend)  $\Rightarrow$   
 Veszem az oszlopokat ebben a sorrendben, a többbit eldobom és a többszörös sorokat is eldobom.  
 Egy oszlop akár többször is szerepelhet.  $\Rightarrow$  átnevezés
- nincs kompatibilitási követelmény (persze amire vetítünk az  $R$ -nek attribútuma kell, hogy legyen)
- Az eredmény öröklí  $R$  típusait és attribútum neveit
- Példa:

$R$	$A$	$B$	$C$
	$a$	$b$	$2$
	$a$	$c$	$3$
	$b$	$c$	$4$

$\pi_A(R)$	$A$
	$a$
	$b$

$\pi_{C,B,B}(R)$	$C$	$B$	$B$
	$2$	$b$	$b$
	$3$	$c$	$c$
	$4$	$c$	$c$

## Műveletek