ZH-Q&D*Mintamegoldás

Orosz József

2001. május 1.

1. Feladat

Az ominózus modell az 1. ábrán látható, az attribútumokat mindenki vegye fel maga. A relációk nálam kb. így néznek ki.

```
Páciens (TAJ, M_ID, nev, szul_ido, ...)
Betegség (B_ID, nev, latin_nev, leiras)
Mentoallomas (M_ID, nev, cim, K_ID)
Korhaz (K_ID, nev, cim, ..., O_ID)
Orvos (O_ID, nev, szakterulet, kor, fokozat, ...)
dolgozik (O_ID,K_ID)
szenved (TAJ, B_ID)
kezel (TAJ, O_ID, datum)
```

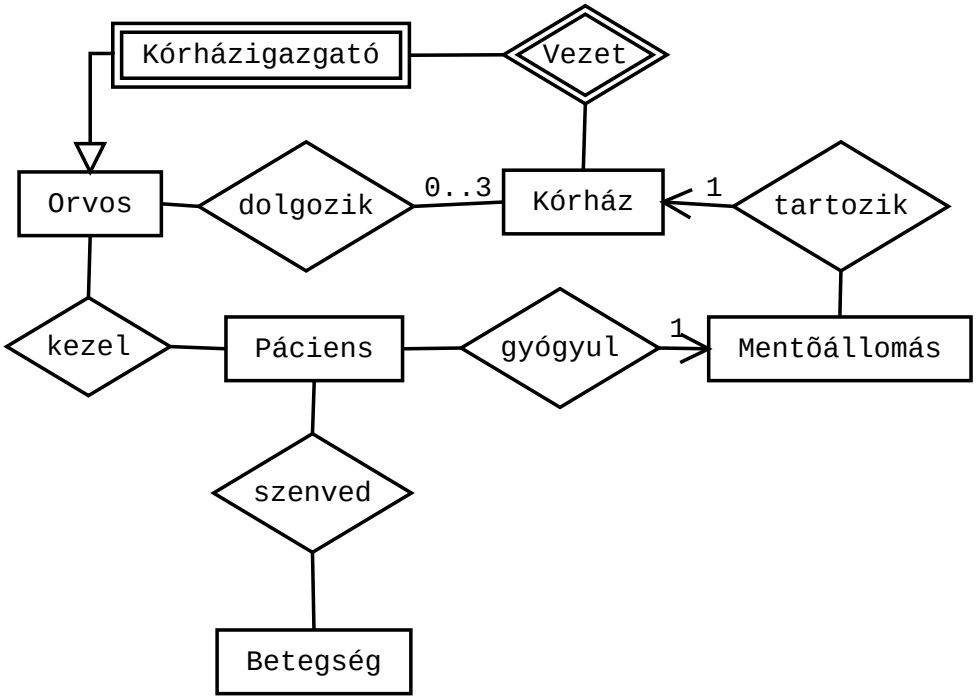
A kórházigazgatós vacakolásra triggert kell írni, az a legtisztább.

Egy másik megoldás (*KisG*). Az orvosos dolgot egy trigger helyett a modellben oldjuk meg. (2. ábra) Ez elegánsabb és kifejezőbb.

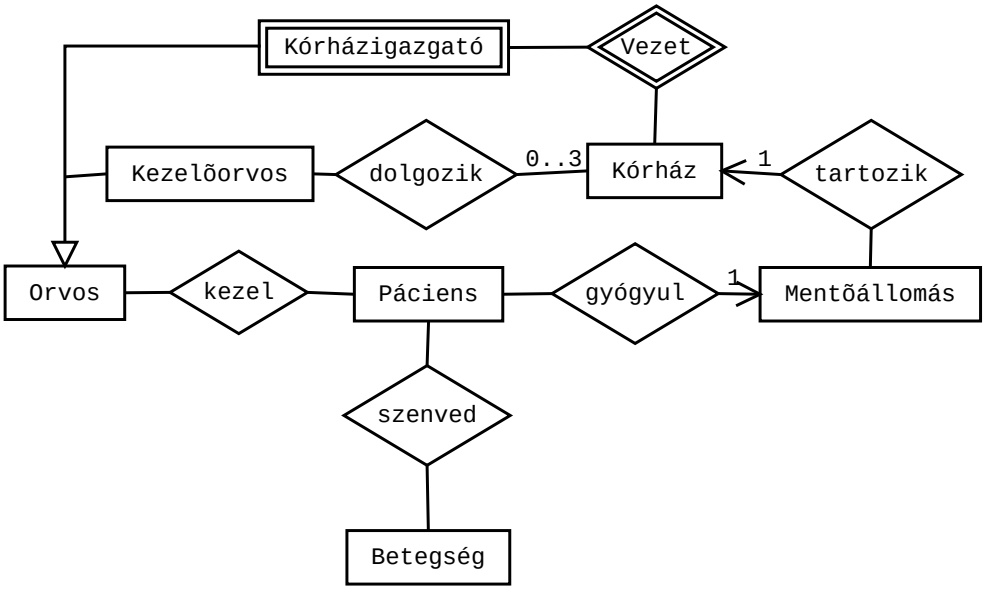
Szívás warning. A Gajdos-könyv nem a Lockemann-féle min-max jelölést használja, és nem beszél öröklésről! Ezért elvileg egyik megoldás sem tökéletes.

Megjegyzés. A gyakorlatban a fenti példát úgy oldanám meg, hogy kikötjük, hogy az O_ID a főorvosoknál mondjuk F-fel, a többi orvosnál O-val kezdődjön. Ekkor a sima orvos reláció felvételét megspóroljuk, és ugyanakkor biztos, hogy a két maradó orvos relációban nem fordul elő ugyanaz a kulcs. A kezel relációban ezek után pedig nyugodtan használhatjuk az O_ID-t.

*Quick & Dirty



1. ábra. Az 1. feladat modellje



2. ábra. KisG féle modell

2. Feladat

A vak is látja, hogy nem. (Meg nyilván azért kérdezik a bizonyítást, mert nem az.) Erre így lehet rájönni:

Favágó módszer. Nosza, táblázatos módszer. Azok kedvéért, akiknek nincs könyvük gyorsan a módszer: az oszlopokba felírjuk az összes attribútumot, a sorokba pedig magukat a darabolt relációkat. Ezután kiinduláskor a-t írunk az I oszlopba, ha I szerepel az adott sorban található részrelációban.

Kiindulási állapot tehát:

	A	B	C	D	E	F	G
ACEFG	a	b1	a	b1	a	a	a
BCDE	b2	a	a	a	a	b2	b2

A módszer ezután a következő lépésekből áll:

- Válasszunk ki egy tetszőleges $X \to A$, függőséget!
- Ha van a táblázatban két olyan sor, amik X-en megegyeznek, és valamelyik sorban A helyén a van, akkor a másik sorba is írhatunk kis a-t az A oszlopába.
- Ha van egy olyan sor, ahol csupa a áll, akkor nyertünk, ellenkező esetben nem.

Rángassuk elő a $C \to F$, és az $E \to G$ függőségeket! Ezt kapjuk:

	A	B	C	D	E	F	G
ACEFG	a	b1	a	b1	a	a	a
BCDE	b2	a	a	a	a	a	a

És akkor itt jól elvagyunk, mert nincsen egyetlen olyan függőségünk se, amit alkalmazni tudunk. Lehetne-e vajon minimálni a függőségi halmazt valahogy úgy, hogy valamelyik függőség bal oldaláról elhagyunk valamit? Nem igazán, A-t ugyanis semmiképpen nem hagyhatjuk el, hiszen sehol sincs a baloldalon, legfeljebb C és B közül tudnánk valamelyiket kitenni. Tehát elakadtunk, mert D-n nincs egyenlőség, és bizony A-n sincs, ezért a maradék három függőségből már egyiket sem tudjuk felhasználni! Vége.

Okosan. Nézzük meg, hogy mi is az R kulcsainak minimális metszete, ergo mik az elsődleges attribútumok: A és E mindig csak a baloldalon szerepel, tehát ezek azok. *Akárhogyan is szedjük szét a relációkat, ahhoz, hogy veszteségmentesen össze lehessen őket*

kapcsolni mindkettőjüknek tartalmaznia kell A-t és E-t. Ennek az az oka, hogy ezeket az elemeket nem lehet másképpen előállítani! Ha akár egyetlen részrelációból is kihagyjuk őket azon a helyen nem fog működni a természetes illesztés! Mivel ez nem teljesül, reszelhetjük a felbontást.

Plusz még azt is láthatjuk, hogy a szétdarabolásnál a vetített függőségekben szépen kigolyózzuk az AB és AC függőségeit, így A elsődleges voltát is!

3. Feladat

Vegyük észre, hogy a feladat nagyon alulspecifikált, így kérdezzük meg, hogy most mi is van?

Ritka index. Nálam a ritka index az au naturel az egyszintű index, tehát akkor ezzel számolunk.

- 4000 bájton 850-es rekordhosszal 4 rekord fér el. Ez azt jelenti, hogy összesen 3882 blokk kell az adatoknak.
- Ehhez jön még az egyszintű index. 4000 bájton 58 db 68 bájtos kulcs-mutató pár fér el. Így a 3882 blokk indexeléséhez 67 lapra van szükség.
- A maximális elérési idő. 5000 bájtba egy lap fér be egyszerre. Ehhez azt kell tudni, hogy hogyan is keresünk egy ilyen ritka indexben. Hát mondjuk ugye binárisan nem, mert az indexlapok nincsenek szorosan egymás után, csak láncolva. Akkor szekvenciálisan. Az viszont azért cikis, mert rossz esetben végig kell olvasni az egész indexet (mind a 67 lapot). Utána még egy lapelérés, tehát max. 68 lap kell.

Ha van elég sok memória, akkor az indexet a memóriában tarthatjuk, és ilyenkor már lehet binárisan keresni, meg minden. Úgyhogy ilyenkor ha beolvastuk az indexlapokat, akkor csak egy-egy lapművelet kell.

B fa. B fában majdnem ugyanaz a helyzet, de itt az indexlapokon eggyel kevesebb kulcsot kell tárolnunk, mint mutatót. Ezt kihasználva arra jutunk, hogy így már 59 bejegyzés is elfér, ami 66 lapot igényel. Ha 59 bejegyzés fér el egy csomópontban, akkor a 66 indexlap még éppen nem fér el két szinten, különösen, ha nem 100%-os csomóponti telítettséggel számolunk. Tehát a fa 3 szintű. és 4 lapművelettel mindig megvan a keresett rekord.

4. Feladat

Triviális. Mit jelent az, hogy R nem BCNF? Azt, hogy van egy olyan nemtriviális $X \to A$ függőség, amire X nem szuperkulcs. A 'nemtriviális" pontosan azt jelenti, hogy $A \notin X$. Ugyanakkor X nem is szuperkulcs, tehát van legalább egy olyan B, amit nem határoz meg. Ebből azt kaptuk, hogy $X \subseteq R \setminus (A \cup B)$, hiszen sem A, sem B nem lehet X eleme.

Most vegyük sorra bele $R \setminus (A \cup B)$ -ből az összes olyan elemet X-be, ami még nincs benne, és nézzük meg mit kaptunk. Az így kapott függőség éppen az amit keresünk, hiszen a bal oldalt addig tuningoltuk, amíg nem lett $R \setminus (A \cup B)$-vel egyenlő. A függőség pedig ugyanakkor igaz, hiszen X meghatározta A-t, és ezt nem tudjuk elrontani azzal, hogy új elemeket veszünk hozzá (axióma). Örülhetünk.

5. Feladat

Ingyen öt pont.

$\{k \mid \exists o \, \exists s \ (LATOGAT(k, o) \wedge ARUL(o, s) \wedge KEDVEL(k, s))\}$

Vagy ha nem tetszik ez a Lockemannos jelölés, akkor a Gajdos-könyv szerinti:

$\Big\{k \mid \exists o \, \exists s \, \exists l^{(2)} \, \exists a^{(2)} \, \exists v^{(2)} \, \Big(LATOGAT(l^{(2)}) \wedge ARUL(a^{(2)}) \wedge KEDVEL(v^{(2)}) \wedge l[1] = k \wedge l[2] = o \wedge a[1$

6. Feladat

Sajnos a jegyzet ezen részéből nem igazán lehetkihámozni sok érdekes dolgot. A lényeg az, hogy OO-nál vannak olyan típuskonstruktorok, hogy SET (halmaz), LIST(lista) és TUPLE (rekord).

Na most ezekből valami ilyet lehet összerakni, bár a fene se érti, hogy ez miért jó így.

```
Newspaper : TUPLE OF (
          title : String,
          editor : Person,
          asseditor : LIST OF (ed: Person),
          sections : SET OF ( sec: Section)
       )
Person : TUPLE OF (
          name: String
         )
Section : TUPLE OF (
          leader: Person,
          authors: SET OF (aut: Person)
         )
```

Gyakorlatilag ezt jobb helyeken úgy írják le, hogy ODL, és az így néz ki:

```
CLASS Newspaper {
   attribute string title;
   relationship Person editor;
   relationship List<Person> asseditors;
   relationship Set<Section> sections;
}
CLASS Person {
   attribute string name;
}
CLASS Section {
   relationship Person leader;
   relationship Set<Person> authors;
}
```