

# **ADATMODELLEZÉS**

**Elmélet és gyakorlat**

**Halassy Béla**

**Budapest, 2000**

© *Dr. Halassy Béla, 2000.*

*A jóban bízó, lélekben fiatal kollegáknak*

# TARTALOMJEGYZÉK

<b>BEVEZETÉS .....</b>	<b>7</b>
Szerkezet és tartalom .....	7
Egyéb tudnivalók .....	8
<b>1. ADAT ÉS INFORMÁCIÓ.....</b>	<b>9</b>
1.1 Az alapfogalmak szemléletmódja .....	9
1.2 Adat.....	9
1.3 Információ .....	11
1.4 „Adatszerű” ismeret.....	11
1.5 Adatdimenziók .....	12
1.6 Adatszerkezeti tényezők.....	14
1.6.1 Egyed.....	15
1.6.2 Tulajdonság.....	16
1.6.3 Azonosító.....	17
1.6.4 Kapcsolat .....	18
1.7 Ellenőrző kérdések.....	20
<b>2. ADATBÁZIS ÉS ADATMODELL .....</b>	<b>22</b>
2.1 Valóság és tükörkép .....	22
2.2 Absztrakciós szintek .....	23
2.2.1 Fizikai szint .....	23
2.2.2 Logikai szint .....	24
2.2.3 Fogalmi szint .....	26
2.3 Absztrakciós vetületek .....	26
2.4 Adatbázis.....	28
2.5 Adatmodell .....	29
2.6 Adatmodell dokumentáció.....	30
2.7 Adatmodell diagram .....	31
2.8 Ellenőrző kérdések.....	33
<b>3. A MODELL ALAPVETŐ SZERKEZETE .....</b>	<b>35</b>
3.1 Az adatmodell mint rendszer .....	35
3.2 Egyedszerkezet .....	35
3.3 Relatív szerep .....	36
3.4 Abszolút szerep.....	37
3.5 Kombinált szerepek .....	38
3.6 Kapcsolótulajdonság .....	39
3.7 Kapcsolatjellemzők .....	40
3.8 Ismétlődő csoport.....	41
3.9 Hálós viszony.....	42
3.10 Mellérendelt viszony .....	44
3.11 Kölcsönös viszony.....	45
3.12 Ellenőrző kérdések.....	45
<b>4. SZERKEZETI FINOMSÁGOK .....</b>	<b>47</b>
4.1 Az adatmodell sokszínűsége.....	47
4.2 Értéktartomány .....	47
4.3 Szerepnév .....	48

4.4	Ál-szerepnév.....	49
4.5	Visszamatató kapcsolat.....	50
4.6	Családfa.....	52
4.7	Üres érték.....	54
4.8	Egyedaltípus.....	54
4.9	Korrekt és inkorrekt kapcsolatok.....	56
4.10	Ellenőrző kérdések.....	57
<b>5.</b>	<b>MODELL, TERV, SZÓTÁR.....</b>	<b>59</b>
5.1	Az adatbázis szemléleti síkjai.....	59
5.2	Metaadatbázis.....	59
5.3	Metastruktúra.....	61
5.4	Adatszótár.....	63
5.5	Adatmodell és adatbázisterv.....	64
5.5.1	Hatókör.....	64
5.5.2	Szint és nézet.....	65
5.5.3	Alapegység.....	66
5.5.4	Tartalmi kör.....	67
5.6	Álegyedek.....	68
5.7	Optimumkritériumok.....	69
<b>6.</b>	<b>AZ ADATMODELLEK HIBÁI.....</b>	<b>72</b>
6.1	Az adatbázis hibáinak a forrásai.....	72
6.2	Tipikus adatmodellezési hibák.....	72
6.2.1	Nyílt logikai átfedés.....	73
6.2.2	Látszólagos logikai átfedés.....	74
6.2.3	Rejtett logikai átfedés.....	75
6.2.4	Technikai homonimák és szinonimák.....	76
6.2.5	A logikai átfedés hiánya.....	77
6.2.6	Fizikai átfedés.....	78
6.2.7	Kiegyensúlyozatlanság.....	79
6.2.8	Inkorrekt nevek.....	80
6.2.9	A példa megoldása.....	81
6.3	A modellezési hibák gyökerei.....	81
6.3.1	Nézetvakság.....	82
6.3.2	Szintkeverés.....	82
6.3.3	Szerepkörtévesztés.....	83
6.3.4	Bemenet/kimenet szemüveg.....	84
6.4	Ellenőrző kérdések.....	84
<b>7.</b>	<b>FÜGGÉSEK ÉS ELŐNORMALIZÁLÁS.....</b>	<b>86</b>
7.1	Normalizálás.....	86
7.2	Relációs adatbázis.....	86
7.3	Funkcionális függés.....	88
7.4	Tartományfüggés.....	91
7.5	Nem-normalizált egyed.....	92
7.6	Többféle ismétlődés.....	95
7.7	A normalizálás első lépése.....	97
7.8	Ellenőrző kérdések.....	98

<b>8. ALAPVETŐ NORMÁLFORMÁK.....</b>	<b>100</b>
8.1 A normalizálás alapjai .....	100
8.2 Második normálforma.....	100
8.3 A normalizálás második lépése .....	102
8.4 Normalizálás és modellstruktúra .....	104
8.5 Harmadik normálforma.....	105
8.6 A normalizálás természete .....	106
8.7 Egy teljes példa.....	108
8.8 A dekompozíció sajátosságai.....	110
8.9 Alternáló kulcs.....	112
8.10 Ellenőrző kérdések.....	112
<b>9. MAGASABB NORMÁLFORMÁK.....</b>	<b>114</b>
9.1 Hány, melyik és milyen a kulcs?.....	114
9.2 Külső kulcstörő függés .....	115
9.3 Boyce-Codd normálforma .....	116
9.4 A normalizálás negyedik lépése .....	118
9.5 Többértékű függés.....	119
9.6 Negyedik normálforma .....	121
9.7 Kapcsolásfüggés.....	123
9.8 Ötödik normálforma .....	124
9.9 Ellenőrző kérdések.....	126
<b>10. SZEMANTIKUS NORMALIZÁLÁS .....</b>	<b>127</b>
10.1 A csoportok csapdái .....	127
10.2 Függéstáblázat .....	128
10.3 Speciális függések.....	129
10.4 Belső kulcstörő függés.....	130
10.5 Metszetfüggés .....	132
10.6 Csoportfüggés .....	134
10.7 Új szemlélet .....	135
10.8 Pszeudo-tranzitivitás .....	136
10.9 Ellenőrző kérdések.....	137
<b>11. NORMALIZÁLÁSI ELJÁRÁSOK.....</b>	<b>139</b>
11.1 A normalizálás mint feldolgozás .....	139
11.2 Normálforma analízis .....	139
11.3 Kapcsolathiány .....	140
11.4 Normálforma szintézis.....	141
11.5 Ellentmondó függések.....	142
11.6 Evidens függések .....	143
11.7 Kulcsmátrix .....	144
11.8 Egyedek közötti függés.....	144
11.9 Lineáris szerkezet .....	145
11.10 Ciklikus függés.....	147
11.11 Közvetett tranzitivitás .....	148
11.12 Fonalak.....	149
<b>12. SAJÁTOS SZERKEZETI TÉNYEZŐK.....</b>	<b>151</b>
12.1 Minőségi adatmodellezés .....	151
12.2 Többszörös inhomogén kapcsolat .....	151

12.3 Szerepnevek függései .....	153
12.4 Homogén viszonyok .....	154
12.5 Feltételes függés.....	157
12.6 Generalizáció és specializáció .....	158
12.7 Unáris egyed .....	159
12.8 Származtatott tulajdonság .....	160
12.9 Szinguláris egyed .....	162
<b>13. AZ IDŐ MODELLEZÉSE .....</b>	<b>164</b>
13.1 Dinamika az adatmodellben .....	164
13.2 Extenzió és intenzió .....	164
13.3 Az idő mint modelltényező.....	166
13.4 Állapot és esemény.....	169
13.5 Állapotmodellezés.....	171
13.6 Eseménymodellezés .....	173
13.7 Inverzió .....	174
13.8 Idősor.....	176
<b>14. KRITIKUS TERVEZÉSI TÉNYEZŐK .....</b>	<b>177</b>
14.1 Hibák - más szemszögből .....	177
14.2 Fogalomalkotás .....	177
14.2.1 Nevek.....	177
14.2.2 Absztrakció.....	179
14.2.3 Leírások .....	181
14.3 „CSAKiS”.....	182
14.3.1 Csoportok.....	182
14.3.2 Azonosítók.....	183
14.3.3 Kódok .....	184
14.3.4 Indexek .....	185
14.3.5 Szerepnevek.....	185
14.4 Sablonok .....	186
<b>15. MODELLEZÉS ÉS RENDSZER.....</b>	<b>188</b>
15.1 Az integráció hiánya .....	188
15.2 Rendszerfolyamat.....	188
15.3 Tájékozódás a modellben.....	191
15.4 Modellkeret.....	194
15.5 Egyedszerepek .....	195
15.6 Sajátos megoldások .....	198
<b>16. TERVEZÉSI ESETTANULMÁNY .....</b>	<b>201</b>
16.1 A példa kerete .....	201
16.2 A kiinduló terv.....	201
16.3 A megoldás elé.....	208
16.4 Ismerkedés a tervvel .....	208
16.5 Az egyértelműség elemzése.....	209
16.6 Azonosság- és azonosítóelemzés .....	210
16.7 Egyéb elemzések .....	213
16.8 Kapcsolatelemzés .....	214

<b>17. KÉT PÉLDA .....</b>	<b>217</b>
17.1 Készletezés .....	217
17.1.1 A mintapéllda tartalmi hibái .....	218
17.1.2 Az adatmodell dokumentálása .....	220
17.2 Egészségbiztosítás.....	223
17.2.1 Papírazonosító.....	223
17.2.2 Háziorvos .....	224
17.2.3 Finanszírozás .....	225
17.2.4 Válaszok .....	226
<b>18. TULAJDONSÁGMODELLEZÉS .....</b>	<b>229</b>
18.1 Egy átfogó példa .....	229
18.2 A tulajdonság lényege .....	232
18.3 Tulajdonságnevek .....	235
18.4 Ami a nevek mögött van .....	238
<b>19. EGYEDMODELLEZÉS.....</b>	<b>242</b>
19.1 Csoportok és rejtett egyedek .....	242
19.2 Az egyed lényege .....	244
19.3 Rossz egyedmegadások .....	245
19.4 Azonosítók.....	248
19.5 Teljesség és minimalitás.....	250
<b>20. KAPCSOLATMODELLEZÉS.....</b>	<b>254</b>
20.1 Rossz kapcsolatmegadások.....	254
20.2 Rossz egyedmegadások .....	257
<b>21. TULAJDONSÁGMODELLEZÉS - MEGOLDÁSOK .....</b>	<b>260</b>
21.1 Egy átfogó példa .....	260
21.2 A tulajdonság lényege .....	262
21.3 Tulajdonságnevek .....	266
21.4 Ami a nevek mögött van .....	269
<b>22. EGYEDMODELLEZÉS - MEGOLDÁSOK .....</b>	<b>272</b>
22.1 Csoportok és rejtett egyedek .....	272
22.2 Az egyed lényege .....	273
22.3 Rossz egyedmegadások .....	275
22.4 Azonosítók.....	279
22.5 Teljesség és minimalitás.....	281
<b>23. KAPCSOLATMODELLEZÉS - MEGOLDÁSOK .....</b>	<b>285</b>
23.1 Rossz kapcsolatmegadások.....	285
23.2 Rossz egyedmegadások .....	288
<b>A. FELADATMEGOLDÁSOK.....</b>	<b>291</b>
<b>B. FOGALOMJEGYZÉK.....</b>	<b>295</b>
<b>C. IRODALOMJEGYZÉK.....</b>	<b>297</b>

# BEVEZETÉS

## Szerkezet és tartalom

Két legalapvetőbb fogalmunk - az adat és az információ - meghatározását követően a könyv mondanivalóját négy részre tagolhatjuk.

Az első részben (2-6. fejezetek) rámutatunk a felhasználói adatoknak (adatbázis) és az azok felépítését leíró szerkezeti képnak (adatmodell) az összefüggéseire. Felvázoljuk az adatmodell alapvető tényezőit és azok összefüggéseit. Majd ismertetjük az ezeken alapuló összetettebb illetve mélyebb struktúrákat. Általában véve az informatikusoknak gondokat szokott okozni az adatmodell és az adatbázisinterv hatókörének a szétválasztása. Ezért el kell mondanunk, hogy miképpen függenek össze ezek egymással is és az adatszótárral is, ami az ismeretekről szóló ismeretek (metaadatok) tárolására és kezelésére szolgál. Ez a rész az adatmodellekre általánosan jellemző hibáknak a kifejtésével zárul.

A második tömb (7-11. fejezetek) célja a matematikai és a szemantikai normalizálási technikák bemutatása. A függések és az előnormalizálás ismertetése után térünk rá előbb a szokványos (1-3NF), majd a magasabb (BCNF, 4-5NF) normálalakok részletezésére. A kilencedik fejezetért előre is elnézést kell kérnünk az olvasótól. Az elméleti teljességre való törekedés jegyében kötelességünk volt a gyakorlatilag nemigen használható és eléggé nehezen érthető normálalakok prezentálása is. A továbbiakban kifejtjük, hogy a közismert matematikai normálformák és normalizálási módszerek korántsem vezetnek a kívánatos eredményre. Ezért szükséges a szemantikai alapú megközelítés.

A harmadik rész (12-15. fejezetek) némileg átfedésben van a korábbiakkal. Például újra bemutatja a sajátos szerkezeti tényezőket, de most már a függési viszonyokra figyelve. Érdekes témakör az idő modellezése. Annak kifejtése után ismét a modellezési bajokról lesz szó. Ezúttal úgy, hogy a kritikus tényezők terén elkövethető hibákat mutatjuk be. Ez a rész egy eddig elhanyagolt tárgykörrel zárul. Mindaddig mikrotechnikaként szemléltük a modellezést, vagyis a részletekkel törődtünk. Most megmutatjuk azt is, hogy miképpen lehet és kell modellezési szemszögből megközelíteni a teljes integrált rendszereket.

A negyedik blokk (16-23. fejezetek) kimondottan gyakorlati jellegű. Előbb egy nagy esettanulmányt vázolunk fel. Majd egy közös fejezetben két kisebb, de igen érdekes példát fogunk megoldani. A további részekben a tulajdonságok, egyedek és kapcsolatok sajátos modellezési problémáit az olvasó által megválaszolandó kérdéssorozatokon át világítjuk meg. Mivel a tulajdonságok és az egyedek modellezésénél impliciten a kapcsolatokról is szó esik, az azokról szóló rész a többinél szükségszerűen rövidebb. Ezt a három példasort követi a megoldások három fejezete.

Művünk legvégén az A. függelékben adjuk közre a fejezeteken belüli kisebb feladatok megoldásait. A könyv fogalom- és irodalomjegyzékkel (B. és C. függelék) zárul.



## Egyéb tudnivalók

Az elméleti fejezetek mindig egy bevezetőnek szánt ponttal kezdődnek. Ezek írják le az adott rész céljait és itt ismertetjük az adott rész tartalmát. Ezt zárójelbe tett dőltbetűs kiemelésekkel tesszük, például: „(1.5 *Adatdimenziók*)”. A további pontok a megjelölt témakörök kifejtésére szolgálnak. A fejezetek közül ellenőrzőkérdésekkel zárulnak azok, amiknél a gyakorlás lényeges és lehetséges.

Előző műveinktől eltérően most tömörebb, kevésbé elbeszélő jellegű a megfogalmazás. A mondanivalót általában egy meghatározás, szabály vagy példa kapcsán fejtjük ki. Külön kiemeltük a lényeges definíciókat illetve szabályokat. Az előbbiekre „+”, az utóbbiakra „-” jel utal. (Vannak kevésbé lényeges, nem szó szerint elsajátítandó meghatározások illetve szabályok is. Ezeket nem emeltük ki az említett módon.) A szövegben magyarázatok, problémafelvetések, kiegészítő megjegyzések találhatók ilyen címkéjű bekezdésekben.

Külön keretekben itt-ott közreadunk figyelmeztetéseket is. Ezek most kivételesen nem a szerző szokásos morgolódásai, amiket a jelen műben igyekezett elkerülni. Csak arról van szó, hogy a bevett technikák - módszerek illetve eszközök - sokszor nem alkalmasak egy-egy összetettebb probléma helyes megoldására. Ezért figyelmeztetnünk kell az olvasót arra, hogy a technika hiányosságait magának kell pótolnia valamilyen módon.

A példákat és az ábrákat a fejezeteken belül számozzuk. Az irodalmi utalások szögletes zárójelben találhatók. Ugyancsak szögletes zárójelben adtuk meg az általános modellezési fogalmak angol nyelvű megfelelőit. Az egyedek tulajdonságsoraiban, a függésekben, az ábrákban, a táblákban számos, a könyvön belül egységes jelölést alkalmaztunk. Ezeket majd a megfelelő helyeken ismertetjük.

Az olvashatóság kedvéért a szöveget jobban tagoltuk, mint e könyv elődeiben. Kerültük a túlzottan sok kiemelést. Helyenként sajátos kereszthivatkozásokat alkalmaztunk. Ezek közül az (← *Egyedszerkezet*) jelölés olyan fogalomra mutat, amit már a könyv egy korábbi részében kifejtettünk. Az (→ *Egyedszerkezet*) jelölés később tárgyalandó fogalomra mutat. Ezzel a hivatkozással is csak akkor élünk, amikor lényegesnek tartottuk, hogy az olvasó világosan érzékelje az előzményeket illetve belássa, hogy a dolgok hálós összefüggései miatt valamilyen fontos kapcsolódó témát csak később áll módunkban kifejteni.

\*\*\*

Most pedig munkára fel! Kívánjuk, hogy az olvasó sikerrel gyarapítsa elméleti tudását és legyenek hasznosak számára példáink a gyakorlati feladatok megoldásánál is.

# 1. ADAT ÉS INFORMÁCIÓ

## 1.1 Az alapfogalmak szemléletmódja

A mindennapi életben az ismeret, tudás, adat és információ fogalmakat felcserélhető módon, vagyis mintegy szinonimákként használjuk. Tesszük ezt annak ellenére, hogy teljesen tisztában vagyunk eltérő lényegükkel. Például a szerző nem képes egyes újságok tőzsdei „információit” értelmezni, mert az amúgyis felesleges tudás lenne a számára. A televízió ma is megismétli a tegnapi hírt, aminek már nyilván nincs információtartalma. Tehát mindkét esetben olyan ismeretről van szó, amely adatként jelenik meg, de nem válik (mindenki számára) információvá.

Az informatikus sem azonos a számítástechnikussal. Az utóbbi a számítógépeivel és a programjaival foglalatossodik. Az előbbi pedig olyan ismereteket szeretne adatszerűen tárolni/kezelné az eszközökkel, amelyek a felhasználók számára információvá válhatnak. Ezért a vérbeli informatikus számára az adat és az információ messze nem azonos lényeg. Az előbbi egy jó vagy rossz tartalmú, jól vagy rosszul elrendezett jelsor. Viszont az utóbbi az a többlettudás, amit a jó tartalmú és jól elrendezett jelsorokból nyerhet a felhasználó.

*Ennek a fejezetnek az a célja, hogy feltárja az adat illetve az információ lényegét és e szavak köznap jelentésén túlmutató értelmet adjon nekik.*

Az ismerettől meglehetősen hosszú út vezet az adatbázison át az információig. Ebben a fejezetben két alapfogalmunk (1.1 *Adat* és 1.2 *Információ*) tisztázása után csak a legelső lépéseket fogjuk megtenni a szóbanforgó úton.

Először is redukálnunk kell az adat lényegét, mert bár technikai értelemben a szöveg, a hang, a kép stb. is adat, ezekkel az ismeretformákkal nem foglalkozunk ebben a könyvben (1.3 „*Adatszerű ismeret*”). Az adatszerű elrendezés lényege az, hogy az ismereteket meghatározott szerkezeti egységekbe kényszerítjük (1.6 *Adatszerkezeti tényezők*). Ez persze egyáltalán nem jelenti azt, hogy elrugaszkodunk az emberközeli valóságtól. Külön pontban (1.5 *Adatdimenziók*) fogjuk kifejteni, hogy a bennünket érdeklő adatbázist alkotó tényezők meglehetősen közeli viszonyban állnak a természetes nyelv elemeivel.

Természetes közléseink is a bennünket érdeklő jelenségek (→ *Egyed*) nekünk fontos sajátosságaira (→ *Tulajdonság*) és azok viszonyaira (→ *Kapcsolat*) vonatkoznak. Az is mindennapos dolog, hogy az ismeret tárgyát pontosan be kell határolni (→ *Azonosító*). Az viszont már az informatikai szakma speciális tudnivalója, hogy az előbb felsorolt négy tényező meghatározott módokon függ össze egymással.

## 1.2 Adat

Háttér: Mint annyi szavunk, az „adat” - és annak angol megfelelője [data] is - a latinból származik. A „datum” - amiből ered a dátum - adományt, ajándékot, adottat jelent. Tehát eredeti jelentésében még nem az ismereteket, hanem az „adott dolgokat” takarta. Később a datál, adatol formákon keresztül vette fel a hozzánk most közelebb eső jelentést.

Előzetes meghatározás:

*„Az adat tények, fogalmak, eligazítások olyan formalizált reprezentációja (tükröképe, megjelenítése), amely alkalmas az emberi vagy az automatikus eszközök által történő kommunikációra, értelmezésre vagy feldolgozásra [1].”*

Magyarázat: Az adat szónak több jelentése van. Itt három aspektusra térünk ki.

Technikai szempontból az adat valamilyen adathordozón tárolt/megjelenő önálló **jelsor**. Ebben az értelemben a számítógépen minden dolog adatok formáját ölti. A szövegeket, a képeket, a hangokat, sőt a programokat is adatoknak tekintett jelsorokban tároljuk és kezeljük. Ezen a síkon a hangsúly a tároláson és a megjelenítésen, végeredményben a fizikai **formán** (vö. „reprezentációja”) van.

Más meghatározások a tényszerűséget emelik ki. Az Idegen szavak szótára [2] szerint a „data ... ismert tényeket, adatokat, dolgokat” jelent. Ez már egy másik sík, mert itt nincs szó a formáról, inkább a **tartalmi** mondanivaló lényeges. Azonban ez nem lehet független az elrendezéstől (vö. „formalizált”). Az adott jelenségcsoportokra vonatkozó ismereteket már a számítógépek megjelenése előtt is irattartókban [file] gyűjtötték. A fájl összefüggő adatsora rögzítette a „feljegyzésre méltó jelenséget”, amit rekordnak [record] hívtak. A könyvelők az ismeretet névvel ellátott mezőkbe [field] kényszerítették. Ezt a név/tartalom párost - például: Típus/Lada - hívták később **adattételnek** [data item], röviden adatnak.

Ebben a könyvben az előzetes definíció értelmezés mozzanatára helyezzük a hangsúlyt.

+

**Az adat értelmezhető (észlelhető, érzékelhető, felfogható, megérthető), de nem értelmezett ismeret [3,4].**

**1.1 példa** „Az X rendszámú kocsi típusa Lada.”  
„Az Y szövetminta desszinszáma 12345678.”

Magyarázat: Mindkét mondat észlelhető (itt van), érzékelhető (írott formában van) és felfogható (latinbetűs) ismeretet közöl. Az első minden olvasó számára érthető, viszont a második csak annak mond valamit, aki tudja, hogy mi az a szövetminta és a desszin. Az értelmezés azt jelenti, hogy a fogalmakat (kocsi, típus stb.) az általunk már ismertekhez kapcsoljuk. Ezt nem mindenki tudja megtenni a második mondatnál. Ezért bármennyire meglepő, mégis igaz a következő kijelentés: A második kitételben lévő ismeret egyesek (az értők) számára adatot hordoz, mások (a nem értők) számára még adatot sem jelent.

A harmadik síkon nem a forma és a tartalmi elrendezés a lényeges, hanem a fogalmak által hordozott **jelentés** a fontos. Mi több, úgy gondoljuk, hogy az elrendezés és a forma csak másodlagos lényeg a jelentéshez képest. Az adat többsíkú szemléletével majd külön pontban is foglalkozunk (→ **Absztrakciós szintek**). Itt csak azt kell leszögeznünk, hogy a textíliákra vonatkozó adatbázist nem tudjuk megtervezni, ha „fogalmunk sincs” arról, hogy mi is az a desszin.

A fenti definícióhoz kapcsoltnak kell kitérnünk a minket szolgáló segédtudományokra. A **szintaktika** az ismeret közlésére szolgáló jelek illetve jelcsoportok olyan általános elrendezését jelenti, amely független az ismeret tartalmától, a megjelenítés módjától és használatától (lásd [1], 413. oldal). Tehát a szintaktika az érzékelés és a felfogás körébe tartozik. Hozzá kapcsolódik a **szemiotika**, a magukkal a jelekkel foglalkozó tudomány, ami a szintaktikával együtt az adatbázis alsó szintjein (vö. kódtervezés) segíti a tervezést.

A **szemantika** a jeleknek illetve jelcsoportoknak a közlendő tartalomhoz való viszonyát jelenti, függetlenül a megjelenítés módjától (lásd [1] 375. oldal). „A rendszám X típusa kocsi Lada.” mondat számunkra annak dacára is zagyvaság, hogy egyes szavait értjük és a tartalomról is

sejtünk valamit. A szemantika a megértés tartományába tartozik. Ez azt jelenti, hogy az ismeret-darabokat valamilyen rend szerint kell egymáshoz fűzni. Tehát a szemantika az adat második síkján játszik szerepet.

### 1.3 Információ

+

**Az információ új ismeretté értelmezett adat.**

**1.2 példa** „Az X rendszámú kocsitípusa Lada.”  
„Az Y gazdánál ezévből a törökbúza hozama 40 mázsa/hektár volt.”

Magyarázat: Az első mondat értelmezhető ismeretet takar, amit mihelyst elolvastunk, rögtön információvá értelmeztünk. Mivel ezt megtettük az előző pontban (← *Adat*), most már nem teljesül a meghatározás „új” kitétele. Akkor a mondat információvá érelődött bennünk, most viszont már csak megismételt tudás.

A második mondat érdekesebb. Nagyon sokan megértik, de egyáltalán nem biztos, hogy az egyik ember ugyanarra a következtetésre jut, mint a másik. Egyikük szerint ez a hozam átlagosan tűrhetőnek tekinthető. Másikuk úgy véli, hogy a termés nem volt rózsás. Aki pedig ismeri Y gazdát és tudja, hogy földjét elöntötte a víz, úgy találja, hogy ezek után az eredmény egészen kiemelkedő.

Az adat és az információ viszonyáról két dolgot kell megjegyezni. Egyrészt azt, hogy nem maga a közölt ismeret az információ, az még csak adat. A „tűrhető”, a „nem rózsás”, a „kiemelkedő” - az adatból levont - egyéni következtetés az információ. Az egyik ember más információvá értelmez(het)i ugyanazt az adatot, mint a másik. Másrészt figyelni kell az információ természetére. Volt egy időszak, amikor a tudósok az információ mérésére törekedtek. Tévedtek. Az adat mérhető, legalábbis sokszor *menyiségileg* is megfogható. A 60 mázsa több, mint a 40 mázsa. Viszont az információ lényege a *minőség*. Nincs sok értelme, hogy a „nem rózsás” információt mindenáron mennyiségi kalodába próbáljuk kényszeríteni. Már csak azért sincs, mert a Pisti „nem rózsás”-a teljesen mást jelenthet, mint a Béla „nem rózsás”-a.

Kiegészítés: A *pragmatika* a karaktereknek ill. jelsorozatoknak az azok értelmezéséhez és használatához való kapcsolata (lásd [1], 317. oldal). Az értelmezésnek az a feltétele, hogy a fogalmak és az azokhoz tapadó jelentések az adatot fogadó tudásával mintegy egybevágnak. A fenti példa első mondata nem értelmezhető az olyan ember számára, aki a „Lada”-t csak márkanévnek tartja, a „Lada 2107” mélyebb lényegét tartva típusnak. Más valaki szerint ez már a modellnév, és a személygépkocsi jelleg tekintendő típusnak.

Az adatbázisstervezőnek nem azt kell vitatnia, hogy melyik felhasználó fogja fel tévesen és melyik helyesen a „típus” fogalom jelentését. Neki az a dolga, hogy minden egyes felhasználó számára az általa információvá értelmezhető adatokat szolgáltatassa.

### 1.4 „Adatszerű” ismeret

Definíció: Adatszerű ismeretekről akkor beszélünk, ha azok adattételekben - mezőkben, rovatokban - elrendezettek (← *Adat* második sík). Ekkor az adatok név/tartalom párosok formáját öltik.

**1.3 példa** „Az X rendszámú kocsí típusa Lada.”  
„Rendszám = X, Típus = Lada”

Magyarázat: Ebben a könyvben csak a nyelvi formát öltő ismeretekkel foglalkozunk. A tárolási/alaki síkon a kép, a hang stb. is adatnak tekinthető, de a multimédia dolgai nem tartoznak rájuk. Az első mondatot *természetes* nyelvben fogalmaztuk meg. Írhattuk volna helyette ezt is: „Az X azonosítójú kocsí fajtája VAZ.” Vegyük észre, hogy a második sokkal formalizáltabb kitétel mögött is nyelv áll, csak éppen egy *mesterséges* nyelvezet.

Probléma: A természetes nyelvi megfogalmazás előnye és hátránya a rugalmasság. A *szövegszerű* ismeret kötetlen tartalmú. Ezért az inkonzisztens lehet (nincs mód a típusok értékeinek az ellenőrzésére); gondokat jelentenek a szinonimák (VAZ/Lada); és bár erre itt nem mutattunk példát, az előbbiek miatt nehéz megteremteni a kapcsolatokat, például egyértelműen megállapítani a kocsí tulajdonosát. Ezért a szövegszerű ismeretkezelés nem alkalmas a korlátos tartalmú, de egyértelmű és egymással kapcsolandó adatok vezetésére.

A köznapi életben adatbázisnak nevezhetjük a számunkra fontos ismeretek bármilyen, akár szövegszerűen is megfogalmazott halmazát. Azonban az informatikai szakmában az ilyen ismereteket tároló/kezelő rendszert nem adatbázisnak, hanem **adatbanknak** hívjuk. Az **adatbázis** megjelölést az adatszerűen tárolt és kezelt ismeretek szervezett együttesére tartjuk fenn. Azok közül sem mindegyikre, mert a „szervezett” jelző nagyon szigorúan meghatározott tartalmat takar. Ennek lényegét a következő pontban kezdjük el kifejteni.

## 1.5 Adatdimenziók

Háttér: Az adatbázisban olyan ismereteket őrzünk, amelyeket a természetes nyelvben is ki tudnánk fejteni. Az adatbázis felépítése nem áll távol a nyelvi szerkezetektől. A közlés legtermészetesebb egysége a (kijelentő) mondat. A mondat szavakból áll. Ezek különböző szerepeket betöltő mondatrészeket alkotnak. A teljes, de minimális közlésben egy alany és egy állítmány szerepel.

**1.4 példa** „Rózsa kocsija fehér.”  
„Gabi kocsija Lada.”

Magyarázat: A mondat *alanya* két részből áll. Ezek hivatalos neve „legközelebbi nem” [latinul: *genus proximum*] és „megkülönböztető jegy” [*differentia specifica*]. Példánkban a legközelebbi nem a KOCSI, a „Rózsa” és a „Gabi” a megkülönböztető jegy. Ugyanez a kettősség az *állítmányra* is vonatkozik. Az első mondat állítmányában a legközelebbi nem a Szín, a megkülönböztető jegy pedig a „fehér”. Míg az emberi közlésben a generikus tényező implicit is lehet (nem írjuk ki, hogy a szín fehér, mert tudjuk, hogy csak a színről lehet szó), addig a számítógéppel mindig tudatni kell, hogy mit takar a generikus elem.

Bachman már a hetvenes években az adat három **dimenziójának** nevezte <sup>[5]</sup> az ismeret specifikus alanyát („Rózsa” kocsija), generikus (Szín) és specifikus („fehér”) állítmányát. Abban a korban még nem voltak adatbázisok: egyszerre csak egyféle jelenséggel törődtek az ismeretkezelés során. Ezért nem utalhatott Bachman a negyedik tényezőre, a generikus alanyra (KOCSI), ami esetünkben már az első dimenzióvá lép elő.

Konvenciók: A négy dimenzió jelölésére a következő egyezményes írásmódokat fogjuk használni. Nagybetűvel írjuk az általános alanyt (KOCSI), félkövérrel a specifikusat (**Rózsa**). Nagybetűvel kezdjük az általános állítmányt (Szín), kiemelés nélküli kisbetűs a specifikus állítmány értéke (fehér).

	ALANY	ÁLLÍTMÁNY
Generikus	KOCSI	Szín
Specifikus	<b>Rózsa</b>	fehér

1.1 ábra: Az adat négy dimenziója

A tényeket nemcsak elemi mondatokban, hanem összetettekben is közölhetjük. Például „Rózsa kocsija fehér és Lada típusú.” Az összetett ismeret is elrendezhető a dimenziókba:

	ALANY	ÁLLÍTMÁNY1	ÁLLÍTMÁNY2
Generikus	KOCSI	Szín	Típus
Specifikus	<b>Rózsa</b>	fehér	Lada

1.2 ábra: Összetett dimenziók

Néha az állítmányt jelzőbe vagy határozóba rejtjük el. Ez az elrendezés szempontjából közömbös. A „Rózsa fehér kocsija Lada típusú.” közlés is az 1.2 ábrához vezet.

—

***Az ismeretközlésből nem maradhat el egyik dimenzió sem.***

Magyarázat: A szabály megsértése esetén az ismeret nem szolgálhat adatként, amint azt jól mutatják a következő ellenpéldák:

- A rózsza ára 50 Ft. Krumpliról vagy virágról van szó? Nincs generikus alany.
- A virág ára 50 Ft. Melyik virágé? A rózsáé, a liliomé? Nincs specifikus alany.
- A rózsza 50. Mije 50? Ennyi darab? Nincs generikus állítmány.
- A rózsza színe. Mi van vele? Hiányzik a specifikus állítmány.

**1.5 példa** „Gabi kocsija Lada típusú.”  
 „Gabi kocsija ötszemélyes.”  
 „Laci kocsija Lada típusú.”  
 „Laci kocsija ötszemélyes.”

	ALANY	ÁLLÍTMÁNY1	ÁLLÍTMÁNY2
Generikus	KOCSI	Típus	Férőhely
Specifikus	<b>Gabi</b>	Lada	5 személy
Specifikus	<b>Laci</b>	Lada	5 személy

1.3 ábra: A kocsik ismeretei sematikus formában

Magyarázat: A természetes nyelv redundáns. Az ábra jól mutatja, hogy ugyanazt a tényt táblázatunk kétszer is leírja. Ötezer Lada típusú kocsit esetében már ennyiszor rögzítenénk azt, hogy a férőhely 5 személy. Az adatbázisokban kerülni kell az efféle redundanciákat a később leírandó okok miatt (→ Az adatmodellek hibái). Ezt könnyen megtehetjük:

- 1.6 példa** „Gabi kocsija Lada típusú.”  
 „Laci kocsija Lada típusú.”  
 „A Lada típusú kocsik ötszemélyesek.”

Úgy takarítottunk meg egy mondatot, hogy nem veszítettünk ismeretet. Most is tudjuk, hogy Gabi és Laci kocsija öt férőhelyes: csak a megfelelő mondatokat kell összekötnünk.

- 1.7 példa** „Gabi kocsija fehér.”  
 „Gabi kocsija Lada típusú.”

Magyarázat: Természetes nyelvű mondataink kétféle jellegűek. Az egyik befejezett, a másik tovább folytatható: „Lada típusú - *tehát* ötszemélyes.” Az adatbázisok szerkezetét is úgy kell kialakítani, hogy abban az egyik ismerethez tudjuk kötni a másikat:

	ALANY1	ÁLLÍTMÁNY1	ALANY2	ÁLLÍTMÁNY2
Generikus	KOCSI	Típus	TÍPUS	Férőhely
Specifikus	<b>Gabi</b>	<i>Lada</i>	<b>Lada</b>	5 személy
Specifikus	<b>Laci</b>	<i>Lada</i>		

1.4 ábra: A kocsis-ismeretek átalakított sémája

Konvenció: Az ábrán dőlt nem-kövért szedetet mutatja az egyik mondatból a másikba átvett állítmányt. A „Laci kocsija Lada típusú és ötszemélyes.” ismeretre úgy teszünk szert, hogy az adatbázis egyik „mondatának” állítmányával összekötjük a másik alanyát. Lásd a sötétebben kiemelt tételeket.

Természetesen az adatbázis nem mondatokból épül fel. Az alkotóelemeket a következő pontban fogjuk ismertetni.

## 1.6 Adatszerkezeti tényezők

Az ismeretek generikus és specifikus tartalom szerint elrendezettek. Ennek megfelelően struktúrákat alkotnak. A szerkezetekben a négy abszolút dimenzió mellett felfedezhettünk két relatívát is. Az egyik mondat állítmánya a másik alanyában folytatódik specifikusan is („Laci kocsija Lada és ötszemélyes”) és általánosan is, hiszen minden kocsik rendelkezik valamilyen férőhellyel. Az adatbázis dimenzióit a következő címek alatt ismertetjük:

- 1.6.1 *Egyed*
- 1.6.2 *Tulajdonság*
- 1.6.3 *Azonosító*
- 1.6.4 *Kapcsolat*.

N.B.: Az azonosító nem külön dimenzió, hanem a kapcsolati tényező alapja.

### 1.6.1 Egyed

+

**Azt a dolgot-izét-valamit, amit ismeretekkel akarunk leírni, egyednek nevezzük.**

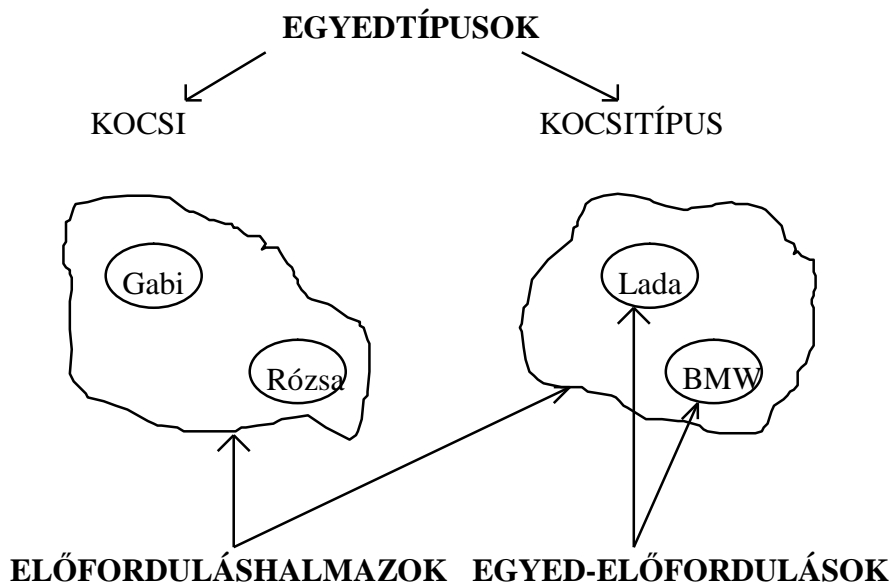
Magyarázat: A mondatokban az alany írja le azt a valamit, amire a közlés vonatkozik. A latinban „entitas”, annak alapján az angolban „entity” a neve annak az önálló lényeggel bíró dolognak, amiről ismeretet akarunk nyerni vagy közölni. Ez bármilyen jellegű lehet: személy, tárgy, fogalom, elképzelés stb. Azonban a fenti meghatározás nem egészen korrekt. A térkép nem azonos az ábrázolt földterülettel, a személy a róla közölt ismeretsorral. Az egyed az ismerettel leírni kívánt jelenség *informatikai tükröképe*.

+

**Az ismeretekkel leírandó jelenségek absztrakt osztálya az egyedtípus. A típusba sorolt konkrét jelenségeket egyedelőfordulásoknak hívjuk. A típus minden időpontban ezek adott halmazával rendelkezik.**

Magyarázat: A „Rózsa kocsija fehér.” és a „Gabi kocsija Lada.” mondatokban egyaránt gépjárművekről van szó. Ezeket ismeretekkel akarjuk leírni és ezért a közös és absztrakt KOCSI osztályba - vagyis egyedtípusba [entity type] - soroljuk őket. A Rózsa és a Gabi kocsija ennek az osztálynak egy-egy konkrét előfordulása [entity occurrence], amelyek együtt alkotják az itt szóbanforgó előfordulások halmazát [entity occurrence set]. Amint látjuk, az egyedtípus és az egyedelőfordulás megfelel az ismeret első két dimenziójának, az általános (típus) és a specifikus (előfordulás) alanynak (egyed).

Kiegészítés: A *konkrét* nem feltétlenül egy fizikai darabot jelent. Ha ezernyi M6-os csavarunk van, akkor nem lesz ennyi előfordulásunk, hanem csak egyetlen egy. Ugyanis nem magukat az egyes csavarokat, hanem az M6-os dolognak a lényegét kell tükröznünk.



1.5 ábra: Az egyedekkel kapcsolatos tényezők



Az *absztrakt* jelző azt mutatja, hogy a tervezőn múlik az általánosítás szintje. Például a kocsikat személy és teher alosztályokba is sorolhatná (→ Generalizáció és specializáció).

## 1.6.2 Tulajdonság

+

**Azt a dolgot-izét-valamit, amivel leírjuk a bennünket érdeklő dolgot, tulajdonságnak nevezzük.**

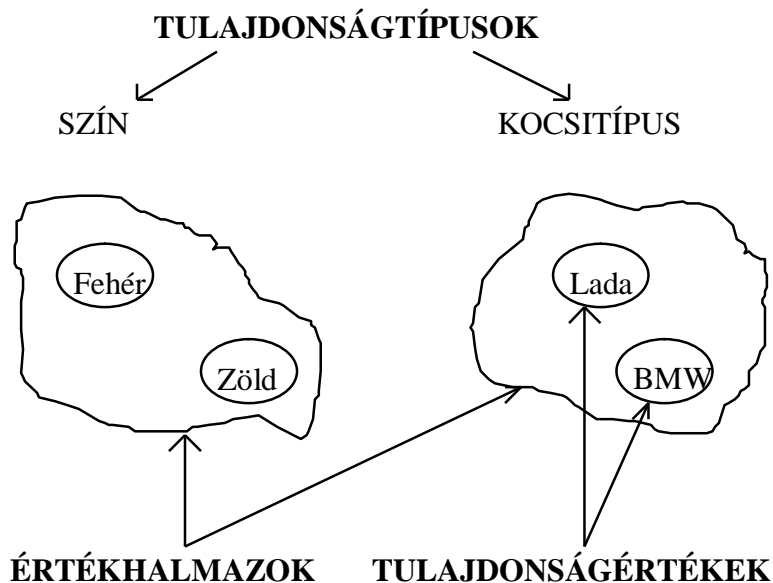
Magyarázat: A mondatokban az állítmány utal a leírt dolog tulajdonságára. Latinul „attributum”-nak, angolul ebből eredően „attribute”-nak hívják a dolgok elválaszthatatlan jellemzőit. Az „elválaszthatatlan” ezt jelenti: Mindegy, hogy fehér-e vagy piros-e Rózsa kocsija, de nincsen kocsí szín és annak konkrét megjelölése (fehér, piros stb.) nélkül.

+

**A jelenségeket leíró ismeretek absztrakt osztályait tulajdonságtípusokkal tükrözzük. A típusba sorolt konkrét ismeretet tulajdonság-előfordulásnak illetve -értéknek hívjuk. A tulajdonságtípus minden időpontban az értékek meghatározott halmazával bír.**

Magyarázat: Az egyedhez hasonlóan a tulajdonság is lehet absztrakt illetve konkrét. A szín absztrakt fogalmat a hozzá képest konkrét fehér, piros stb. színekből vonatkoztattuk el. A Szín tehát egy olyan tulajdonságtípus, amelynek előfordulása illetve értéke a fehér és a piros. Természetesen a szín általános értékhalmozába a kék, sárga stb. is beletartozik.

Probléma: A tulajdonság sem azonos a valósággal, hanem annak csak tükörképe. Ezért figyelni kell arra, hogy például a felhasználó a konkrét „Lada” és „Lada 2104” értékek közül melyiknek megfelelően értelmezi a Kocsitípus absztrakt fogalmát.



1.6 ábra: A tulajdonságokkal kapcsolatos tényezők

### 1.6.3 Azonosító

– **Az egyed és a tulajdonság viszonylagos modellezési tényezők.**

Magyarázat: Az egyed és a tulajdonság nem azonos a valósággal, hanem tükörkép. Ezért nem mondhatjuk eleve például azt, hogy a szín tulajdonság. Ha a színt a kocsik leírására használjuk, akkor tulajdonság. Ha a színt magát akarja valaki leírni - például a spektrumbeli helyével -, akkor a színt egyedként kell felfogni. A Lada kocsitípus lehet egy konkrét érték, de lehet egy - férőhellyel leírandó - konkrét egyedelőfordulás is.

+ **Az egyedelőfordulás egyértelmű hivatkozását azonosításnak hívjuk.**

Magyarázat: Az egyedelőfordulásokról nem tudunk úgy beszélni, hogy ne adnánk meg egy vagy több tulajdonságuknak az értékét. A „Rózsa” kocsiját úgy választottuk el „Gabi” kocsijától, hogy a tulajdonosok keresztnévére utaltunk. Ez a hivatkozás csak e két kocsi esetében egyértelmű, különben ezer kocsi tulajdonosa is viselheti a Gabi nevet. Ezért a keresztnév nem alkalmas az azonosításra. A célszerű azonosítók két csoportba tartoznak.

**Nominális** hivatkozásról beszélünk akkor, ha a dolgot névén nevezzük. Kovács Rózsa ilyen hivatkozása maga a személynév. Felhívjuk a figyelmet arra, hogy a logika szabályai szerint nem csak a „névszerű” név tekintendő annak. A kocsi rendszáma, a vásárlás számlaszáma is név. Mivel a természetes nevek - Kovács Rózsa - igen ritkán egyediek, vagyis egyértelműek (többen is viselhetik ezt a nevet), igen gyakran kénytelenek vagyunk mesterséges neveket (vö. számlaszám) alkalmazni az azonosításra.

A másik lehetséges megoldás a **deszkriptív** hivatkozás. Addig kapcsoljuk egymáshoz a jelenségek természetes sajátosságait, ameddig egyedi behatároláshoz nem jutunk. Például a személy nevét kiegészítjük a születési dátummal és hellyel, ha pedig még ez sem elég, akkor az édesanya nevével és/vagy a lakcímmel. Tehát egyetlen elemi tulajdonság helyett egy tulajdonságösszetételt választunk azonosítóknak.

+ **Az egyedtípus azon elemi tulajdonságtípusát vagy tulajdonságtípus-kombinációját, amely minden egyedelőfordulásra eltérő értéket vesz fel és azokkal kölcsönösen és tartósan egyértelmű viszonyban áll, az egyedtípus azonosítójának nevezzük.**

Magyarázat: A KOCSI egyedtípus Rendszám tulajdonságtípusa látszólag jól megfelel az azonosítás követelményeinek. Egy kocsinak nincs több rendszáma és egy rendszám nem tartozik több kocsihoz. Igaz ez a kitétel? Csak időlegesen! A holnapi Fordom felveheti a mai Ladám rendszámát és ha ellopják a rendszámtáblát, akkor új tartalmút kaphatok.

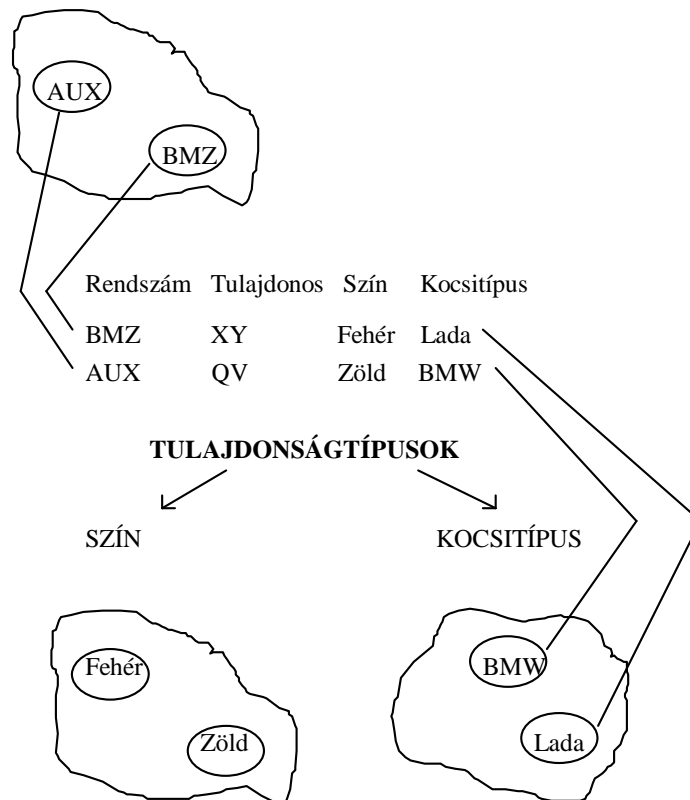
+ **Az egyedek és a tulajdonságok összefüggéseit egyed-tulajdonság-viszonyoknak nevezzük.**

Magyarázat: Egy egyedtípusnak (KOCSI) több tulajdonságtípusa (Rendszám, Szín stb.) van. Egy tulajdonságtípus (Szín) több egyedtípust jellemezhet (KOCSI, HÁZ stb.). Ezért az egyed és a tulajdonság nem fölé/alárendelt viszonyban áll egymással, amint azt a hagyományos adatkezelésben feltételezték, hanem olyan egymás mellé rendelt, vagyis egyenrangú tényezők, amelyeket egyed-tulajdonság-viszony párokban (KOCSI - Szín, HÁZ - Szín stb.) kell elképzelnünk. Ezen viszonyok között kiemelkedő szerepe van az azonosítóknak, amire nem csak a hivatkozás miatt van szükség. Arra épülnek az egyedek közötti összefüggések is (lásd a következő alpontot).

Rendszám	Tulajdonos	Szín	Kocsitípus
FGS 802	XY	Piros	Lada
AUI 999	QW	Zöld	BMW

## EGYEDTÍPUS

KOCSI



1.7 ábra: Az egyedek és a tulajdonságok összefüggései

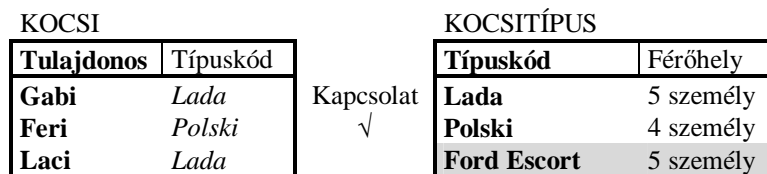
### 1.6.4 Kapcsolat

+

**A bennünket érdeklő jelenség-viszonyokat kapcsolatoknak hívjuk.**

**1.8 példa** „Gabi kocsija Lada típusú és ötszemélyes.”  
 „Laci kocsija Lada típusú és ötszemélyes.”  
 „A Lada típusú kocsik ötszemélyesek.”

Magyarázat: Ha az első mondatból elhagyjuk a befejező állításrészt, akkor is tudjuk a harmadik mondat alapján, hogy Gabi kocsija hány férőhelyes. Az adatbázisokban kerülni kell a redundanciákat, és ezért többféle „mondatot” alkotunk, vagyis többféle egyedtypust képzelünk el (→ *Adatdimenziók*). Erre nem csak az átfedés elkerülése miatt van szükség. A gépkocsitípus olyan jelenség, amely a konkrét gépkocsik nélkül is leírható önálló ismeretekkel. Például az adatbázisunkban nincs is még Ford Escort típusú konkrét autó, de attól még tudjuk, hogy az a típus is ötszemélyes.



1.8 ábra: A kocsiiismeretek ártértelmezett sémája

Ismeretet nemcsak úgy szerezhetünk az adatbázisból, hogy egy egyed (KOCSI) adatait keressük elő, hanem úgy is, hogy az egyikből a másikba (KOCSITÍPUS) lépünk át.

+

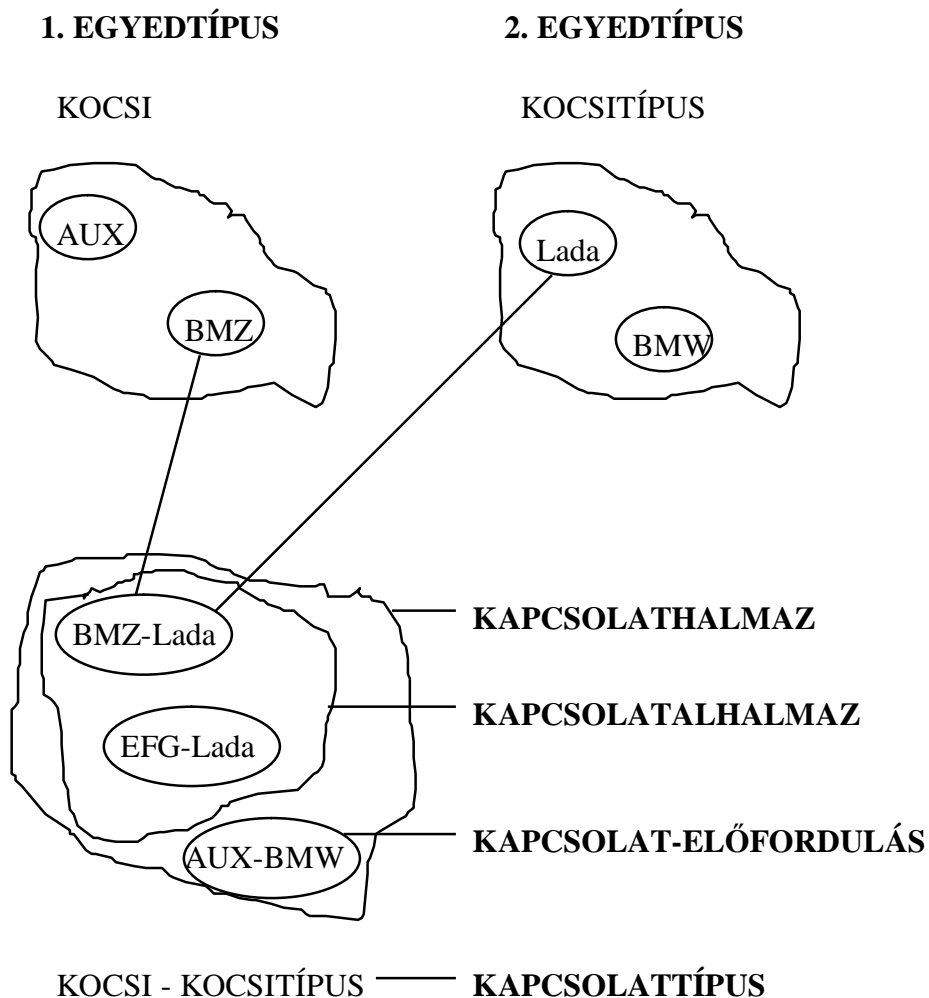
**A jelenségek viszonyainak absztrakt osztályait kapcsolatátípusokkal tükrözzük. A típusba sorolt konkrét viszonyt kapcsolatelőfordulásnak nevezzük. A kapcsolatátípus minden időpontban az előfordulások általános halmazával és egyedenkénti alhalmazával rendelkezik.**

Magyarázat: Minden kocsinak van típusa. Ezért a KOCSITÍPUS - KOCSI összefüggés általános kapcsolatátípus [relationship type]. A Gabi-Lada és a Laci-Lada viszony ennek egy-egy előfordulása. A KOCSITÍPUS Lada egyedéhez kettő, Polski egyedéhez egy, Ford Escort egyedéhez nulla elemű KOCSI alhalmaz tartozik. Általánosan a kapcsolatátípus halmazát három előfordulás alkotja (két Lada, egy Polski).

### 1.9 példa RENDELÉS (Rendelésszám, ...) RENDELÉSTÉTEL (Rendelésszám+Cikkszám, ...)

Magyarázat: Egyes esetekben az egyed kulcsa (← *Azonosító*) olyan konkatenáció, amely más egyedek azonosítóiból tevődik össze. Az ilyen összetételeket a „+” jellel mutatjuk. A rendeléstételek a Rendelésszám elemen keresztül kapcsolódnak a rendelésekhez.

Probléma: Egyes modellezési koncepciókban kiterjedten alkalmazzák az ún. *kapcsolat által való azonosítás* jónak vélt megoldását. Ha meghatározzuk a példa két egyede közötti RENDELÉS-RENDELÉSTÉTEL kapcsolatot, akkor az alapján el tudunk jutni az egyik egyedből a másikba és felesleges a második egyedben megismételni a Rendelésszámot. Ez a felvetés teljesen hibás. Eddig arról volt szó, hogy a kapcsolat a különböző egyedeket leíró tulajdonságok együttes előkeresésére alkalmas (lásd 1.8 ábra). Arról nem beszéltünk, hogy az ismereteket be is kell illeszteni az adatbázisba. Mármost ha a rendeléstételeknél nem adnánk meg a Rendelésszám értékét, akkor ugyan miképpen lehetne azokat betenni az adatbázisba és kapcsolatokkal a rendelésekhez kötni? A tervezőnek tudnia kell, hogy a kapcsolat általi azonosítás csak egy sajátos technikai megoldás, nem pedig olyan alapvető tényező, amely az adatszerkezetnek a természetes és nélkülözhetetlen eleme.



1.9 ábra: A kapcsolatokkal összefüggő tényezők

## 1.7 Ellenőrző kérdések

- 1/1 Információt (I) vagy adatot (A) takar-e a következő mondat: „Halassy Béla e mű szerzője”.
- 1/2 Információt (I) vagy adatot (A) rejt-e a következő leírt mondat: „A szerzőnek BMZ 873 rendszámú 1300-as Ladája van.”.
- 1/3 Információt (I) vagy adatot (A) hordoz-e a következő mondat, vagy netán egyiket sem (S): „Sári 6 éves.”.
- 1/4 Ez a jelsor: „navigadnoj” észlelhető-e (1), érzékelhető-e (2), felfogható-e (3) és értelmezhető-e (4) az Ön számára? Adja össze a válaszának megfelelő pontokat.

- 1/5 Az alábbi kijelentések közül melyik teljes (T) és melyikből hiányzik az általános (1) v. specifikus (2) alany, az általános (3) v. specifikus (4) állítmány?
- A kocsí típusa Lada.
  - A szerző kocsijának a típusa Lada.
  - Pityu másfél méter magas.
  - Sára Katié kilencvenes.
- 1/6 A következő mondatok közül melyik sejtet viszonyt (V) újabb közlés felé és melyik tovább nem vezető leíró (L) ismeret?
- Pista kocsija ötszemélyes.
  - A rendeléstételben a cikkszám 1234.
  - Gábor szervezői tanfolyamra jár.
  - A híd hossza 236 méter.
- 1/7 Az alábbi kitételek közül melyik igaz (I), melyik hamis (H)?
- A szövegszerű kezelés nem alkalmas több állomány kezelésére.
  - A szövegszerű kezelés nem alkalmas az ismeretkapcsolásra.
  - A készletkimutatást adatszerűen érdemes kezelni.
  - A múzeumi tárgyak leírására a kétféle kezelés együttese a célszerű.
- 1/8 Melyik mondat rejt egyedtypust (T), -előfordulást (E) és -halmazt (H)?
- A Fillér utcai lakók nem szeretik a karalábét.
  - Kíváncsiak vagyunk a Fillér utcai lakók ismereteire.
  - Gabi a Fillér utcában lakik.
- 1/9 Nyilvántartást kell vezetnünk a betegellátás finanszírozásáról. Adatokat kell tárolnunk az orvosokról, a betegekről és a társadalmi biztosítottakról, mint lehetséges betegekről. Ön hány (n) egyedtypust lát e feladat mögött?
- 1/10 Az egészségügyi dolgozók pótlékot kapnak. Ön miként tükrözné a pótlékra vonatkozó ismereteket? Egyedtypussal (E), tulajdonságtypussal (T) vagy netán mindkettővel (M)?
- 1/11 Lát-e különbséget a RENDELÉS egyed Rendelésdátum és Vevőkód tulajdonsága között? Fejtse ki a gondolatait.

## 2. ADATBÁZIS ÉS ADATMODELL

### 2.1 Valóság és tükörkép

A mindennapos szóhasználatban az **adatbázis** kifejezést teljesen kötetlenül használjuk a bennünket érdeklő ismeretek halmazának a megjelölésére. Az informatikai szakmában ez a laza szófordulat nem megengedhető, mert az adatbázis olyan technikai szakfogalom, aminek a tartalma nagyon pontosan meghatározott.

Az adatbázis legfontosabb két vonása az, hogy tudatosan szervezett és közösen használt ismeretegyüttes. A tudatos szervezés azt jelenti, hogy a valós jelenségeknek, azok fontos sajátosságainak és viszonyainak a tükörképeit (→ Egyed, Tulajdonság, Kapcsolat) egyre mélyebb ismervek szerint rendezzük el. A térkép nem azonos a földterülettel, annak csak tükörképe. A térkép mindig valamiről szól (első szint), vagyis van lényege. Ugyanarról a területről készülhet ilyen vagy olyan tartalmú térkép (második szint). Végül az egyikféle térképen így, a másikon úgy ábrázolják például a magasságokat (harmadik szint). Az adatbázis mélységeivel foglalkozik a fejezet első pontja (2.2 **Absztrakciós szint**).

A geológus és a hidrológus az adott földterületnek részben azonos, részben pedig eltérő ismereteiben érdekelt. Nem egyetlen egy térképet (vö. állomány), hanem térképekből álló atlaszt (vö. adatbázis) használnak eltérő módon, de együttesen. Az egyik így, a másik úgy szemléli ugyanazt a valóságot. Erről szól a második pont (2.3 **Absztrakciós vetület**). Azt akarjuk megértetni, hogy bár az atlasz valamennyi térképét senki sem akarja együtt látni (globális, mindent átfogó nézet), mert más területegységekre, más dimenziókra kíváncsi a geológus, mint a hidrológus (parciális, egyedi nézetek), az atlaszra mégis szükség van.

***Ennek a fejezetnek az a célja, hogy bemutassa az ismeretszervezés kettős lényegét: a vertikális szinteket és a horizontális nézeteket. Ezen túlmenően kitér az adatmodell dokumentációjának a legelemibb kérdéseire is.***

Az atlaszban lévő részletező térképek együttese jelenti a konkrét tudást, a szervezett ismeretegyüttest. Erről szól a 2.4 **Adatbázis** pont. A részletező térképek nem ad hoc módon kerülnek kialakításra. Egymáshoz azáltal illeszkednek, hogy az atlaszban létezik globális térkép is (különben nem beszélhetnénk részletezésről). A konkrétumok - például a Mecsek vagy a Mátra - mindig beilleszkednek egy átfogó képbe. Bennünket leginkább ez a fogalmi szintű és globális nézetű kép érdekel. Lásd a 2.5 **Adatmodell** pontot. A modell az az áttekintő térkép, amelyen alapulnak az atlasz lapjai.

Amint a térképeken is léteznek egyezményes jelek és azokhoz is kapcsolódnak leírások, úgy az adatmodell közreadására is megfelelő szabályok vonatkoznak. Ezeket fogják bemutatni a 2.6 **Adatmodell dokumentáció** és a 2.7 **Adatmodell diagram** pontok.

## 2.2 Absztrakciós szintek

Amint a világon nincs két egyforma falevél, úgy nem létezik két tökéletesen azonos ember vagy gépkocsi sem. A modellezőt nem az ilyen egyedi konkrét jelenségek érdeklik. Ő a dolgok közös, lényeges és tartós vonásai alapján osztályokat - SZEMÉLY, KOCSI stb. - alakít ki (←*Egyed*).

Az absztrakció az a gondolati művelet, amelynek során a különös, lényegtelen és időleges sajátosságoktól elvonatkoztatunk. Például a tehergépkocsinak más a funkciója, mint a személygépkocsinak, de nem alkotunk kétféle kocsi osztályt, hanem a funkciótól eltekintve egyetlen KOCSI egyed típusban gondolkodunk.

Az absztrakciót nem egy síkon hajtjuk végre. Minden ismeretnek van jelentése, tartalmi elrendezése és formája. A személynek más a lényege, mint a kocsinak (jelentés). Az egyik vállalatban más ismereteket óhajtanak vezetni a kocsiról, mint a másikban (tartalom). Ugyanazt a tartalmat pedig eltérő módokon jeleníthetjük meg az adathordozó eszközökön (forma). A jelentéshez, tartalomhoz és formához kapcsolatosan beszélünk az adatbázis három absztrakciós szintjéről. Ezeket a következő alpontokban ismertetjük:

- 2.2.1 *Fizikai szint*
- 2.2.2 *Logikai szint*
- 2.2.3 *Fogalmi szint*

### 2.2.1 Fizikai szint

A tudnivalót az alábbi mondatok világítják meg. Az első három ugyanazt a tényt tudatja velünk, vagyis **tartalma** azonos, szemben a negyedik mondatával. Ugyanakkor az első három mondat **formája** eltérő, miközben az első és a negyedik kitétel alakja azonos:

- 2.1 példa** „Béla születési dátuma 1946. május 14.”  
„Béla születési dátuma 1946.V.14.”  
„Béla születési dátuma 05/14/1946.”  
„Meli születési dátuma 1973. március 16.”

Az adatkezelő rendszerek számára a gépen tárolt adatok tartalma úgymond közömbös, mert ők az ismereteket azok megjelenési formája és helye szerint kezelik. Más formátumú a karakteres, mint a numerikus vagy a dátum típusú adat. Az ismeret alaki sajátosságait az ún. **adatábrázolás** [data representation] rögzíti. Ez két tényezőből áll: a *típusból* és a *méretből*. Például a Személy neve adat lehet karakteres típusú és 40 jel hosszú.

Az adatokat különböző elvek szerint lehet elrendezni a háttértárazon. Az **adatszervezés** [data organisation] - amit az állományokra redukáltan fájlservezésnek [file organisation] is neveznek - két egymással összefüggő tényezőre irányul. Az egyik az **elhelyezési mód** [location mode], amely az adattartalmak és a tárolóhelyek közötti viszonyt jelenti. Például a mód lehet soros (fizikai sorrendű, logikai rendezettség nélkül) úgy, hogy index biztosítja a szekvenciális (logikai sorrendű) visszakeresést. A másik tényező a **hozzáférési mód** [access mode]. Egy adott elhelyezés egy vagy több hozzáférési módszernek ad alapot. Esetünkben a rekordok vagy fizikailag sorosan, vagy index szerinti sorrendben érhetők el.



+

**Fizikai adatszerkezetnek hívjuk az adatok ábrázolásának és a tárolón való elhelyezésének illetve hozzáféréseinek a meghatározott rendjét.**

Magyarázat: A fizikai szint mesterséges absztrakció. Azért az, mert az ismeretnek nem természetes sajátossága az ábrázolás. Például a Születési dátum nevű ismeret karakteres, numerikus és dátum típusú adatként egyaránt megfogalmazható. Ennek a tulajdonságnak nem eleve elrendelt jellemzője a dátumszerű forma. Az ábrázolást a tervező választja ki különböző megfontolások alapján. Tehát egyazon tartalom (1) többféle (N) formában jelenhet meg. Vagyis a tartalmak (→ *logikai szint*) eltérő módokon képezhetők le a fizikai szintre.

+

**Fizikai adatfüggetlenségről beszélünk akkor, ha az adatok elhelyezési, hozzáférési és ábrázolási módjában bekövetkező változásokkor nincs szükség a programok módosítására.**

Magyarázat: A korszerű adatkezelők a tárolás fizikai részleteit többé-kevésbé elrejtik a (programozó) felhasználók elől. Ha tehát az adatábrázolás és/vagy az adatszerkezet megváltozik, akkor alig van szükség a programok átirására. Ez az „alig” viszont azt is sejteti, hogy ma még a fizikai függetlenség elve nem érvényesül tökéletesen, amint azt jól mutatja a kétezredik év körül felmerült problémakör is.

## 2.2.2 Logikai szint

Az adatbázis tervezését nem az adatábrázolás és -szervezés jellemzőinek a kijelölésével kezdjük (← *Fizikai szint*). Először tartalmilag rendezzük el az adatokat. Az elrendezési feladatot **adat-szerkesztésnek** [data structuring] hívjuk. Ez az állományok, az azok közötti viszonyok és az adatmezők meghatározását jelenti.

Egyazon (1) elvi tartalom (→ *Fogalmi szint*) számtalan (N) eltérő módon rendezhető el. Az alábbi ábra a kocsikra és tulajdonosaikra vonatkozó ismeretek háromféle különböző szerkesztését mutatja.

Az elrendezések nem adnak egyformán jó megoldásokat. A harmadik esetben azért nem látszik a két fájl kapcsolata, mert a tervező fizikai mutatóval akarja azt megteremteni. Ez rossz választás, mert sérti a fizikai adatfüggetlenség elvét (← *Fizikai szint*). A második változatban egy név alatt több érték jelenik meg, vagyis ún. **ismétlődő adatról** van szó. A probléma ekkor nemcsak az, hogy egyes kezelőrendszerek nem képesek az ismétlődések menedzselésére, hanem az is, hogy a szerkezet nehezen bővíthető újabb ismeretekkel, például a gépkocsi típusára vonatkozó részletező adatokkal.

Az első megoldás jónak látszik, de a másik kettővel összevetve abban is fellép egy gond. Egyes rendszerek nem képesek **adatcsoportok** kezelésére. Ezért azokat vagy részekre kell bontani (1. verzió), vagy elemi adatként kell megfogalmazni (2. és 3. verzió). Egyik megoldás sem jó, mert az előbbi esetben procedurálisan kell összeállítani, az utóbbiakban pedig ugyanígy kell megbontani a dátumot, ha az egészre illetve a részekre van szükség.

TULAJDONOS

Személy	Év	Hó	Nap
Béla	1946	május	14

KOCSI

Rendszám	Személy
BMZ 873	Béla
ZBM 378	Béla

## TULAJDONOS

Személy	Születési-dátum	Rendszámok
Béla	1946. 05. 14.	{BMZ 873, ZBM 378}

## TULAJDONOS

Személy	Születési-dátum	√	Rendszám
Béla	1946. V. 14.	?	BMZ 873 ZBM 378

## KOCSI

2.1 ábra: Az ismeretek különféle szerkesztési lehetőségei

+

**A technikai, hozzáférési és hatékonysági követelményeknek illetve korlátoknak megfelelően kialakított tartalmi adatstruktúrát nevezzük az adatbázis logikai szerkezetének.**

Magyarázat: A logikai tervezési szinten a tervezőnek számba kell vennie a kezelő *technikai* korlátait. Figyelnie kell a *biztonsági* aspektusokra. Például legbiztosabb az, ha a személyekre vonatkozó titkos adatokat külön tárolja azoktól, amelyek mindenki által kezelhetők. Ügyelni kell a *hatékonyságra* is. Néhány adatkezelő sokkal jobb teljesítményt nyújt több egymáshoz kapcsolt kevés mezőt tartalmazó állomány esetében, mint egyetlen sokmezősé. Azonban a legfontosabb tudnivaló az, hogy kerülni kell az álmegoldásokat.

Személy	...	Páratlan	Páros
Kovács	...	X forint	Y forint
Szabó	...	Z forint	Q forint

2.2 ábra: Egy „ravasz” megoldás

A példa a bérelszámolásról szól. Ebben a Páros és a Páratlan valójában nem egy adat, hanem a bérügyi adatoknak egy-egy sorozata. A tervező úgy képzelte el a dolgot, hogy januárban a páratlan, februárban a páros mezősorba írja a béradatokat. Márciusban már nincs szükség a januári adatokra, ezért felülírja a páratlan adatokat, áprilisban pedig a párosakat és így tovább. Világos, hogy ez a struktúra csak addig felel meg a céljának, ameddig nincs szükség egyszerre kettőnél több hónapra nézve is vezetni a béradatokat (a 13. havi fizetésről pedig feledkezzünk meg).

## SZEMÉLY

Személy	...
Kovács	...
Szabó	...
Kerekes	...

## BÉR

Személy	Hónap	Forint
Kovács	1	X
Kovács	2	Y
Szabó	1	Z
Szabó	2	Q
Kovács	3	W

2.3 ábra: Az okos megoldás

A valóság az, hogy egy személyhez 0, 1, ..., N bér kapcsolódhat. Az utóbbi verzió hűen tükrözi ezt a ténnyt. Az előbbi változatban az N értéket kettőre korlátozta a tervező. Ezért nem tartotta be az egyik igen fontos tervezési elvet:

- + **Logikai - más néven adat-program - függetlenségről beszélünk akkor, ha az adatok logikai szerkezetében bekövetkező változás nem hat ki a kezelőprogramokra.**

Magyarázat: A „ravasznak” hitt megoldás esetében ha kettőnél több hónapra is vezetni kell a bérek adatait, akkor nemcsak a logikai adatszerkezetet kell módosítani, hanem egyben az összes meglévő programot is át kell írni.

### 2.2.3 Fogalmi szint

Minden ismeretnek van jelentése, tartalma és formája. A jelentés nem azonos magával a tartalommal. A születési dátum egy lényeg, de megfogalmazható egyetlen csoport helyett három elemi adatban (év, hó, nap) is. A személy származási helye, neme és születésének évszázada három különböző lényeg, mégis annak idején - más ismeretekkel együtt - ezt a három dolgot az egyetlen Személyi szám adat tartalmában testesítették meg.

Technikai, biztonsági és hatékonysági okokból ugyanazt a lényegi jelentést többféle tartalmi elrendezésben tükrözhetjük (← *Logikai szint*). Ezért világos, hogy a jelentés meg kell, hogy előzze a tartalmat, vagyis az adatbázis tervezőjének először a lényeggel, tehát magukkal az alapfogalmakkal (származási hely, nem stb.) kell tisztában lennie.

- + **Fogalminak nevezzük a jelenségeket, azok sajátosságait és viszonyait a valóságnak megfelelő módon és természetes fogalmakban tükröző adatstruktúrát.**

Magyarázat: A tervezőnek két dologra kell figyelnie. Egyrészt arra, hogy a logikai terv szintjén a mesterséges megjelölések is megengedettek. A „rekordképben” a természetes „HÓNAP” név helyett írhatunk „HON”-t is. Sőt, egyes kezelők meg sem engedik a hosszú és/vagy ékezetes nevek használatát. Ezért a logikai szintű terv már nem természetes. Másrészt arra, hogy a kezelő nem zárja ki a valóságtól elütő elrendezések alkalmazását. Miközben a valóság az, hogy egy (1) személyhez tetszőleges számú (N) bér kapcsolódhat, a tervező a „tetszőlegeset” kettőre redukálhatja a logikai tervében. Lásd az idevágó példát (← *Logikai szint*). Ezért a logikai terv az adott esetben nem felel meg a valóságnak.

**Figyelmeztetés!** Az adatbázis terveket úgynevezett CASE [Computer Aided Systems Engineering - számítógéppel támogatott rendszertervezési] segédeszközökkel állítják össze. Az ilyen terveket sokszor minden alap nélkül nevezik fogalmi szintűeknek. A CASE alkalmazása nem zárja ki például azt, hogy egyetlen „tulajdonságban” teljesen eltérő fogalmakat (pl. származási hely és születési dátum) vegyítsenek.

## 2.3 Absztrakciós vetületek

Minden szervezetben léteznek olyan eltérő funkciók, amelyek ellátásához az emberek részben közös, részben különböző ismereteket igényelnek. Például a járművek esetében a biztosítási díj befizetésekor a tulajdonos adatai fontosak, a kockázat megállapításához a kocsitípus adatai

nélkülözhetetlenek, de mindkét esetben érdekesek a kocsi alapadatai. Ezért a típus illetve a tulajdonos ismereteinek a részletezésében nem érdekelt részlegek a következő egymástól szeparált nyilvántartásokat fogják vezetni:

KOCSI-1

Rendszám	Szín	Típusnév	Tulajnév	Tulajcím
X	F	Lada	Rózsa	C1
Y	Z	Lada	Gabi	C2
Z	F	Polski	Lajos	C3
Q	P	BMW	Rózsa	C1

KOCSI-2

Rendszám	Szín	Típusnév	Férőhely	Fogyasztás
X	Fehér	Lada	5	A
Y	Zöld	Lada	5	A
Z	Fehér	Polski	4	B
Q	Piros	Polski	5	C
W	Piros	BMW	4	B

2.4 ábra: Kétféle KOCSI állomány

Ez a kettős struktúra többféle baj forrása. A tervezők valamilyen módon elrendezték a tartalmakat (logikai szint), de nem gondolkodtak jelentésekben (fogalmi szint) és a mély megvalósítások is ellentmondások (fizikai szint). Még az utóbbi a legkisebb gond, bár kétségtelenül nem szerencsés, hogy a Szín adatot hol kódolják, hol pedig nem.

Ha egy szervezetben ugyanarra a lényegi jelenségre (KOCSI) vonatkozóan többféle nyilvántartást is vezetnek, akkor alapvető problémaként lép fel a **redundancia**. Ez nem a nagyobb tárigény, a többszörös kezelés és a többszörösen megírandó programok miatt igazán gond. Hanem azért, mert nagyon nagy az **inkonzisztencia** veszélye. A formai (lásd Szín) mellett felléphet tartalmi (lásd a Q kocsi típusát) eltérés is. A kiterjedési körben sem egybeváogók az állományok. Nem lehet tudni, hogy a W kocsit már törölték az elsőből és még nem a másodikból, vagy már bevitték a másodikba, de még nem az elsőbe.

Mindezeknél még nagyobb probléma az igények **változása**. Pl. egyik állományban sem szerepel a Díjtétel adat. Az arra is kíváncsi felhasználó ezért létre fog hozni egy harmadik KOCSI-3 állományt, hogy a hiányt pótolja. Végeredményben nagyon jellemző, hogy egy szervezetben az eltérő funkciókat ellátó egységek, csoportok egymástól eltérő struktúrájú, külön-külön kezelt állományokkal dolgoznak.

Az ismeretek korrekt és problémamentes tükrözését akkor lehetne jól megoldani, ha a felhasználók tisztában lennének a vetületek lényegével.

KOCSI

Rendszám	Szín	Típuskód	Tulajkód
X	Fehér	K1	T1
Y	Zöld	K1	T2
Z	Fehér	K2	T3
Q	Piros	K3	T1
W	Piros	K2	?

TULAJDONOS

Tulajkód	Tulajnév	Tulajcím
T1	Rózsa	C1
T2	Gabi	C2
T3	Lajos	C3

#### KOCSITÍPUS

Típuskód	Típusnév	Férőhely	Fogyasztás	Díjtétel
K1	Lada	5	A	II
K2	Polski	4	B	I
K3	BMW	5	C	IV

2.5 ábra: Egyetlen közös struktúra

A közös szerkezet előnyei világosak. Nem többszörösek a kocsiadatok (redundancia). Egyértelmű az adattartalom (inkonzisztencia), bár a W kocsi tulajdonosát még meg kell adni. Új ismeretek is felvehetők a maguk logikus helyére (változás - lásd Díjtétel) anélkül, hogy újabb állományra lenne szükség. Azonban a legnagyobb előny az, hogy ki lehet gyűjteni olyan ismeretkombinációkat is, amelyek több fájlból származnak. Például a KOCSI-X ismeret nem származtatható a kétféle kocsi állományból, mivel azok nem közös használatúak, viszont minden további nélkül levezethető az egyféle közös struktúrából.

#### KOCSI-X

Rendszám	Szín	Férőhely	Tulajcím	Díjtétel
X	Fehér	5	C1	II
Y	Zöld	5	C2	II
Z	Fehér	4	C3	I
W	Piros	4	?	I

2.6 ábra: Egy „vegyes” felhasználói szemlélet

Feltehetőleg a közös struktúrát egyetlen felhasználó sem látja a maga teljes egészében. Ezt a minden adatot felölelő látásmódot *globális szemléletnek* nevezzük. A csak a típus vagy csak a tulajdonos ismereteiben érdekelt felhasználó látásmódját *parciális nézetnek* [view] hívjuk, mert a teljes struktúrának csak az adott részletéről van szó. Mindez pedig azt jelenti, hogy az ismereteket nem csak a jelentés, tartalom, forma szintjei szerint, tehát mintegy *vertikálisan* kell tagolni (→ *Absztrakciós szintek*). Szükség van az adatok nézet szerinti *horizontális* tagolására is.

## 2.4 Adatbázis

*Az adatbázis nem adatbank.* Vagyis nem a korlátos azonosító ismeretekkel ellátott, de egyébként korlátlan tartalmú szöveges ismeretek összessége. Ha például az Interneten tallózunk, akkor bizonyos kulcsinformációk alapján (azonosító ismeretek) eljuthatunk egy szövegszerű (korlátlan tartalmú) információhoz. Ekkor pedig nem rovatokba kényszerített módon - vagyis adatszerűen - fogjuk látni az ismereteket.

*Az adatbázis nem adatállomány.* Egyetlen adatszerűen megfogalmazott állomány vagy ilyenek ad-hoc - szervezetlen - együttese nem tekinthető adatbázisnak. A legkorszerűbb fejlesztési eszközök segítségével is mód van arra, hogy meghatározzuk az alábbi tartalmú állományt, amely rengeteg adatátfedést tartalmaz:

Név	Cím	Rendszám	Típus	Férőhely	Szín
Rózsa	Forint utca	AAA 111	Lada	ötszemélyes	fehér
Gabi	Fillér utca	BBB 222	Lada	ötszemélyes	fehér
Rózsa	Forint utca	CCC 333	Polski	négyszemélyes	piros
Gabi	Fillér utca	DDD 444	Polski	négyszemélyes	zöld

2.7 ábra: Egy rosszul szerkesztett adatállomány

Az ábra nem adatbázist, hanem egy hibás tartalmi tervet mutat. A terv azért rossz, mert nem a valós fogalmak összefüggésein alapul. Ugyanis a célszerű, a fogalmakat helyesen tükröző elrendezés a következő:

TULAJDONOS

Név	Cím
Rózsa	Forint utca
Gabi	Fillér utca

KOCSI

Rendszám	Név	Típus	Szín
AAA 111	Rózsa	Lada	fehér
BBB 222	Gabi	Lada	fehér
CCC 333	Rózsa	Polski	piros
DDD 444	Gabi	Polski	zöld

KOCSITÍPUS

Típus	Férőhely
Lada	ötszemélyes
Polski	négyszemélyes

2.8 ábra: Egy jól szerkesztett adatbázis

+

**Az adatbázis véges számú egyedelőfordulásnak, azok egyenként is véges számú tulajdonságértékének és kapcsolatelőfordulásának az adatmodell szerint szervezett együttese.**

Magyarázat: Az egyed, a tulajdonság és a kapcsolat tényezőket a korábbiakban már meghatároztuk. Már csak az a kérdés, hogy mit jelent az „adatmodell szerint” kitétel.

## 2.5 Adatmodell

Az adatmodell szót az informatikában kétféle értelemben használják. Az egyik oldalon az ismereteknek az igen általános elrendezését jelenti. Például a később tárgyalandó ún. *relációs adatmodell* esetében az ismeretek adattáblákat alkotnak, amiket a *hierarchikus adatmodell* esetében szegmenseknek neveznek és részben más elvek alapján rendeznek el. Ebben a könyvben mi nem ezt az általános adatmodell fogalmat fogjuk használni. Itt az adatmodell mindig a konkrét, az adott szervezetben megvalósítandó adatbázis fogalmi szintű tervét jelenti.

Ha az X és az Y felhasználó teljesen *eltérő* ismeretekben érdekelt, ha valaki a saját - másokat nem érintő - nyilvántartásában egy *lényegtelen* adatot óhajt vezetni, ha az adat *ad-hoc*, vagyis nem tartósan jellemző, akkor nincs szükség modellezésre. A számítógépen úgy is lehet adatokat kezelni, hogy azok nem közösek, nem lényegesek, nem tartósak.

A modellezés a közös, lényeges és tartós jegyek kiemelését, absztrakcióját jelenti. Nem is akármelyik síkon (← *Absztrakciós szintek*). Az adatok összefüggéseit nem lehet pusztán a logikai vagy a fizikai szinten elrendezni. Az egyik felhasználó kocsitípusnak nevezi (logikai szint) és numerikusan kódolja (fizikai szint) ugyanazt az ismeretet, amit a másik kocsifajtának hív és másként ábrázol. Ezért az egyeztetést a fogalmi szinten kell kezdeni. Tisztázni kell azt, hogy a kocsitípus és a kocsifajta vajon azonos fogalmak-e?

+

**Az adatmodell véges számú egyed típusnak illetve azok egyenként is véges számú tulajdonság- és kapcsolattípusának a szervezett együttese.**

Magyarázat: Az adatmodellben nem állományokat, adatmezőket, adott fizikai módon megvalósított kapcsolatokat jelölünk ki. Az a feladat, hogy valós jelenségek (például tulajdonos és kocs) valós kapcsolatait és tulajdonságait tükrözzük. Ebből következően az adatmodell mindig a jelentéshez (← *Fogalmi szint*) kapcsolódik és mindig a teljes globális nézetnek felel meg (← *Absztrakciós vetületek*). Ezért kizárólag csak a *fogalmi szintű adatbázis tervet* hívjuk adatmodellnek, vagyis ez a két kifejezés valójában szinonima.

**Figyelmeztetés!** A mai tervezési segédletek arra szolgálnak, hogy a tervező egy-egy rendszerrész *logikai* adatszerkezetét könnyen meghatározhassa. Arra többségük nem képes, hogy a teljes rendszer *fogalmi* szerkezetét is elrendezze. Miközben az egyik tervező igen komolyan tekintett technikai támogatással kidolgozza pl. a partnerre vonatkozó adatok „adatmodelljét”, addig a másik tervező ugyanazzal az eszközzel felépíthet egy, az előzővel ellentmondásban álló másik partner „adatmodellt”.

Az adatmodell a teljes szervezet szintjén egyetlen elvi-fogalmi struktúra. Ennek felel meg az esetlegesen többszörös logikai/fizikai adatbázis-szerkezet. Például fogalmi szinten tisztáznunk kell, hogy mit jelent egy szolgáltató vállalatnál a „fogyasztási hely”. Az pedig már egy másik kérdés, hogy egy vagy több állományban (tartalom) és milyen módokon (forma) rögzítjük a konkrét fogyasztási helyekre vonatkozó ismereteket.

Az adatbázis tartalma strukturálisan az adatmodell felépítésének felel meg. Tehát a modell egyed-, tulajdonság- és kapcsolattípusai szerint rendezzük el az előfordulásokat. Azonban az adatmodell nem pusztán struktúra, hanem a fenti definícióhoz kapcsoltan a típusoknak és az azokra vonatkozó *korlátoknak* [constraints] a szervezett együttese. Minden kocsi adott kocsi-típusba tartozik. Ez egy kapcsolati korlát. Minden kocsinak van fogyasztása. Ez egy tartalmi korlát. A fogyasztás nem lehet nulla és ötezer liter/kilométer. Ez pedig egy értékhatár.

Az adatmodell mindig fogalmi szintű és nézetfüggetlen (globális vetületű) lényeg. Rá nem-szerkezeti fogalmi korlátok is vonatkoznak. Például a gépkocsi fogalomba nem fér bele az ötezer liter/kilométer fogyasztás. Ekkor már más fogalomról - netán rakétáról - van szó. Tehát a fogalom nemcsak szerkezeti, hanem tartalmi feltételek által is behatárolt.

## 2.6 Adatmodell dokumentáció

Ebben és a következő pontban az adatmodell dokumentálásáról lesz szó. Erről a fontos témáról akkor is beszélnünk kell, ha az adatmodellek egy részét ma már CASE-eszközök segítségével készítik és így úgy mondható a dokumentálás módja. Egyrészt hazánkban a CASE-ek még nem igazán elterjedtek. Másrészt nem mindig a célszerű módon használják őket. Harmadrészt még igen kevés olyan technikai segédeszköz létezik a világon, amely „emberi fogyasztásra” is alkalmas dokumentációt produkál. A legtöbb csak a mellékletbe való listák/táblák özöneivel árasztja el a felhasználót. A dokumentációkból szinte mindig hiányzik a kellő magyarázat, a mérlegelés - vagyis az adatmodell lelke.

A modelldokumentáció három egyaránt fontos tényezőből áll. Az egyed-, tulajdonság- és kapcsolattípusokat, ezek viszonyait és a rájuk vonatkozó korlátokat *táblázatokban* és *listákban* rögzítjük. Ezek szabványos felépítésűek, tartalmúak és formájúak.

+

**Az adatmodell formális leírását fogalmi sémának hívjuk.**

A fogalmi séma nem azonos a rekordképpel, amely logikai/fizikai szintű dolog és csak az állományok adatmezőit (vagy ami ugyanaz: a táblák oszlopait) sorolja fel. A fogalmi modell dokumentációjában - amint azt a név is mutatja - fogalmakat írunk le, ezért ennek a dokumentációnak nélkülözhetetlen része a fogalmak magyarázata, a logikai tervben már nem szereplő fogalomtár avagy szótár is (→ *Adatszótár*).

A magyarázatok kedvéért a formális részt kötetlen szöveges *leírásokkal* egészítjük ki a szótáron túlmenően is. Például ilyen módon indokoljuk meg a modell felépítését. Mivel a táblák/listák és leírások nem alkalmasak az összefüggések áttekintésére, a szemléltetésre *ábrákat* is használunk (→ *Adatmodell-diagram*).

Ez a dokumentáció a teljes adatmodellt tartalmazza, vagyis a globális nézet leírására szolgál. A parciális nézeteket nem ebben a dokumentációban, hanem a feldolgozások terveiben kell megadni.

+

**A globális adatmodellnek a feldolgozások által látott és kezelt részeit adatalmodelleknek, ezek formális leírásait pedig alsémáknak hívjuk.**

A „látott és kezelt” nem ugyanazt jelenti. Például ha valaki úgy akarja megjeleníteni a gépkocsi férőhelyét, hogy magát a típuskódot nem akarja kijeleztetni, akkor ez az utóbbi ismeret nem látott, de kezelt, mert nélküle nem tudnánk kikeresni a férőhelyet. Viszont az átlagos fogyasztás származtatott adat látott, de mivel nem tárolt, nem is kezelt.

A látott és kezelt adatok sokszor nem egyetlen egyedtípusból származnak. Viszont megjelenítésük olyan, hogy a felhasználó úgy érzékeli, mintha egy egyedet alkotnának. A több valós egyedből összeállított ismeretegyütteseket *virtuális egyedeknek* nevezzük. A 2.6 ábra mutatott egy ilyen virtuális egyedet.

## 2.7 Adatmodell diagram

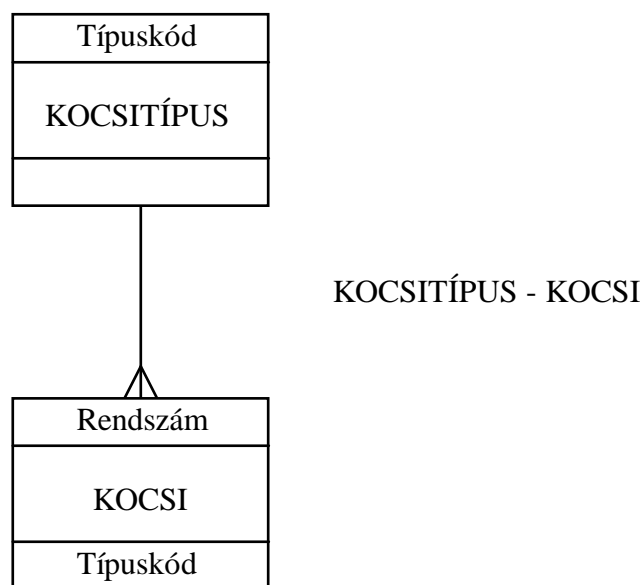
Az adatmodell szerkezetének az ábrázolására szolgáló rajzok az adatmodell diagramok. Felhasználjuk nyomán ezeket Bachman-diagramoknak [6] is nevezzük. Noha időközben a konvenciók több változáson mentek át, a lényeg változatlan maradt. Bár hasonló ábrákat használnak az adatbázisok logikai, sőt fizikai (← *Absztrakciós szintek*) struktúrájának a szemléltetésére is, természetesen itt csak a fogalmi szintű egyezményes jelek érdekesek.

Az ábrán téglalapok jelölik az *egyed típusokat*. A téglalap mérete tetszőleges. Az eredeti konvenció szerint a téglalap közepén található az egyedtípus neve. (Újabb divat, hogy az egyedtípusok jelölésére legömbölyített sarkú téglalapokat használnak. Ennek nincs semmi jelentősége. Az az érv, hogy itt a fogalmi szintről van szó és a logikai szinten használják a szögleletes dobozokat, nem állja meg a helyét. Ugyanis egy dokumentációban a két szint nem keveredhet...)

A téglalapon belül lehet feltüntetni a *tulajdonságtípusokat*. Azonban a hosszú listák eltorzítják a diagramot és megnehezítik az áttekintést. Ezért általában csak az azonosítót és az értelme-



zési/kapcsolati szempontból fontos tulajdonságok nevét szokták bejegyezni a dobozba. Konvenciónkban az **azonosító** - és annak részei - az egyed neve fölött, a többi esetleg érdekes tulajdonság az alatt helyezkedik el. A téglalapokat összekötő vonalak tükrözik az egyedek közötti **kapcsolattípusokat**. A vonal mellé írjuk a kapcsolattípus nevét. Sajátosságait több konvencióval fejezzük ki. A folytonos vonal az adott egyedtípus felől kötelező, a szaggatott vonal az onnan nézve opcionális összefüggést jelzi. A sima vonalvég „egyes”, a varjúlábás [crowsfoot] „ennes” fokot mutat az adott oldalon. Ha két kapcsolat kizáró - ilyen nem szerepel az ábrán -, úgy a kettőt átmetsző kapocs tükrözi ezt a tényt. Ha a kapcsolat visszamutató, akkor a vonal ugyanabba a dobozba tér vissza, mint ahonnan elindult. Mindezeket a sajátosságokat egy külön pontban részletesen is bemutatjuk (→ Kapcsolatjellemzők).



2.9 ábra: Az adatmodell-diagram alapvető konvenciói

Egyéb konvenciót mi nem alkalmazunk. Azért nem, mert azok már nem fogalmi szintű sajátosságokra utalnának. Például egyesek külön jelölik azt a tényt, hogy a kapcsolattípus részt vesz-e az azonosításban. Mivel fogalmi szinten ez nem megengedett (Azonosító  $\Leftarrow$ ), ezzel az egyezményes jelöléssel nem kell foglalkoznunk.

**Figyelmeztetés!** Egyes tervezési segédletek kézikönyveiben azt ajánlják, hogy a tervező a részletező egyedekkel kezdje a rajzolást, méghozzá felülről lefelé és balról-jobbra. Ez az ábrázolásmód teljesen ellentmond a modellezés szellemének, hiszen az a lényeg - és nem a részlet - kiemelésére tanít bennünket. Ezért az ilyen kifacsart megoldásokról jobb elfeledkezni. Sokkal célszerűbb, ha a fontos egyedet valahol az ábra közepén helyezzük el, felülre téve a minősítéseket, alulra pedig a részletezéseket. Azonban ez is csak tanács; amúgy mindenki hallgasson a józan eszére.

## 2.8 Ellenőrző kérdések

2/1 Ön szerint melyik kitétel igaz (I) és melyik hamis (H):

- Az adatbázis a számítógépen tárolt adatok együttese.
- Hanglemezeim adatait egy dBASE-állományban tartom. Ez adatbázis.
- Van egy HANGLEMEZ és egy KIADÓ fájlom. A kettő között a közös Kiadókód adattal tartok kapcsolatot. Ez adatbázis.
- Az adatbázis a felhasználó számára fontos ismeretek halmaza.

2/2 Az SQL nyelvben létezik az ún. „duplapontosságú” numerikus adattípus. Ön szerint ez fogalmi (F), logikai (L) vagy fizikai-tárolási (T) szintű tényező?

2/3 A SZEMÉLY állományban szerepel a nyelvtudásra vonatkozó ismétlődőcsoport (Nyelv, Vizsgadátum, Nyelvpótlék). A tervező három ilyen csoportot alkalmaz. Véleménye szerint ez fogalmi (F), logikai (L) vagy fizikai (T) szintű szerkezeti megfontolás?

2/4 Az előző feladatra vonatkozóan döntse el, hogy igazak (I) vagy hamisak (H) a következő megjegyzések:

- A baj az, hogy negyedik nyelvet nem lehet megadni.
- A fogalmi szintű tervben külön NYELV egyedet illene tervezni.
- Ha biztos, hogy további nyelvre soha nincs szükség, akkor a terv nem rossz.
- A kezelés mindenképpen nehézkes, mert melyik csoport az angol nyelv?

2/5 A tervező meghatároz egy külön Telefonszám adattípust. Ennek tartalmát az előre elképzelt felépítés szerint validálja, noha sima karakteres módon tárolja. Fogalmi (F), logikai (L) vagy fizikai (T) megfontolásról van-e itt szó?

2/6 Egy jó kezelő használata esetén Ön szerint az alábbi tényezők közül melyeket kell változtatni (V), melyeket esetleg (E) és melyeket nem (N), ha a Telefonszám adat méretét egy karakterrel megnöveljük?

- az adatot tartalmazó bemeneti képek
- az adatot tartalmazó kimeneti képek
- az adatot kezelőprogramok
- az adatbázis fizikai szerkezete
- az adatbázis logikai szerkezete

2/7 Van egy kárrendező és egy biztosítási-díj megállapító részlegünk. Az egyik arra kíváncsi, hogy egy káreseményben milyen bajok érték az abban résztvevő kocsikat. A másik arra, hogy egy kocsi milyen károk értek és emiatt miként számolja ki például a bonus-malus százalékát. Ön hányféle ismerettel leírandó jelenséget lát e feladat mögött? Adjon meg egy számot.

2/8 Mondja meg, hogy melyik megoldás a helyes (H) és melyik a rossz (R) az előző feladatok támogatására:

- A két részlegnek nincs köze egymáshoz. Itt két adatbázis szükséges.
- A feladatok részben közös ismeretekre támaszkodnak. Legjobb lenne két részben átfedő adatszerkezetet kreálni, ügyelve arra, hogy a közös KÁR egyed karbantartása egyszerre történjen meg a két részben.
- Itt csak egyetlen adatbázisnak a két nézetéről van szó.

2/9 Iskolai adatbázist tervezünk a tanárokról, a tanulókról és a bedolgozókról. Van aki a tanár által oktatott tárgyakra, más a tanuló által tanulandókra, ismét mások a tantárgyakhoz a segédeszközöket szállító személyekre kíváncsiak. A tanárt a képzettséghez, a diákot a jegyekhez, a bedolgozót az eszközökhöz kötődő ismeretek jellemzik. Ezek az ismeretek kölcsönösen kizáróak (például a tanár nem kap jegyet). A kérdés az, hogy milyen módon lehet az egyetlen SZEMÉLY egyedhez kapcsolódó különböző ismereteket mindenki részére a saját maga által igényelt módon megjeleníteni? Adja meg az alábbi változatok közül a helyesnek tartott válasz sorszámát:

- Három különböző SZEMÉLY rekordképet kell tervezni.
- A sajátos igénynek megfelelő szemléletet virtuális egyedben fejezzük ki.
- Saját programmal összebogarásszuk az ismereteket.

2/10 Az alábbi kitételek közül melyik igaz (I), melyik hamis (H):

- Az adatmodell nem más, mint egy adatmodell-diagram.
- Az adatbázis tervét rekordképekben kell megadni.
- Az adatbázis tervét jó bő szövegben le kell írni, hogy a felhasználó értse.
- Lista, szóközzel leírás és pár áttekinthető rajz a modell veleje.

### 3. A MODELL ALAPVETŐ SZERKEZETE

#### 3.1 Az adatmodell mint rendszer

Az adatbázis az egyed-, tulajdonság- és kapcsolatelőfordulásoknak az adatmodell szerint szervezett együttese. Az adatmodellt ugyanezen tényezők típusai alkotják. Ahhoz, hogy a „szervezett” jelzőt megértsük, tisztában kell lennünk azzal, hogy a típusok egy jól meghatározott rendszer szerint kötődnek egymáshoz. Az egyedekhez tulajdonságsorok tartoznak és az egyedek között kapcsolatok léteznek (3.2 *Egyedszerkezet*).

A tulajdonságok különböző feladatokat látnak el az egyedeken belül is (3.3 *Relatív szerep*) és az egyedek között is (3.4 *Abszolút szerep*) úgy, hogy adott esetben a funkciók összetettek is lehetnek (3.5 *Kombinált szerepek*). A modellt alkotó háromféle tényező igen szorosan kötődik egymáshoz ezeken a szerepeken át. Az egyedek egyes tulajdonságai adnak alapot a közöttük lévő kapcsolatoknak (3.6 *Kapcsolótulajdonság*).

*Ebben a fejezetben az adatmodell mint rendszer tényezőivel és az azok viszonyaival kapcsolatos alapvető tudnivalókat ismertetjük. Végeredményben azt fejtjük ki, hogy mit kell érteni az adatbázis szervezetségén.*

Az adatbázis célszerű szerkesztéséhez nem elég ismerni az általános összefüggéseket. Tisztában kell lenni a szerkezetek részleteivel is. Egyelőre - fontosságuk miatt - csak az átfogóbb szerkezeti egységekre térünk ki, a mélyebb részleteket majd csak a következő fejezetben ismertetjük (→ *Szerkezeti finomságok*).

A szerkezeti egységeket a kapcsolatok alapozzák meg. Ezért bemutatjuk azok alapvető sajátosságait (3.7 *Kapcsolatjellemzők*). A fejezet utolsó pontjai arra hívják fel a figyelmet, hogy milyen szerkezeti megoldásokat nem szabad alkalmazni a modellben. A további tudnivalókat a következő címek alatt lehet megtalálni:

- 3.8 *Ismétlődő csoport*
- 3.9 *Hálós viszony*
- 3.10 *Mellérendelt viszony*
- 3.11 *Kölcsönös viszony*

#### 3.2 Egyedszerkezet

Háttér: Az adatmodell szerkezetét egyed-, tulajdonság- és kapcsolattípusok alkotják. Ezek a tényezők egyenlő fontosságúak és pontosan meghatározható viszonyokban állnak egymással. Összefüggéseiket leginkább az egyedhez kapcsoltan érthetjük meg.

Megjegyzés: A modell mindig típusokból épül fel. A rövidség kedvéért a „típus” szót el fogjuk hagyni a megjelölésekből, amikor modellbeli összefüggésekről van szó. Tehát pl. egyedtípus helyett egyszerűen csak egyednek fogunk írni, azon természetesen típust értve.

+ **Az egyed tulajdonságainak a sorát az egyed belső szerkezetének nevezzük.**

**3.1 példa** „Az X rendszámú fehér, Lada típusú kocsí Rózsáé.”  
 „A Lada típusú kocsik ötszemélyesek.”  
 „A kis Polski típusú kocsik négyszemélyesek.”

Magyarázat: Mindhárom mondatban vannak közös *konkrét* szavak. Azonban az első mondat *általános* elrendezése eltér a másik kettőétől, amelyek belső rendje azonos. Az adatmodell mindig tipikus elrendezéseken alapul. Az egyedekhez (alany) rendeljük hozzá a tulajdonságokat (állítmány). A belső szerkezetet sablonosan így ábrázoljuk:

**3.2 példa** KOCSI (**Rendszám**, Tulajkód, *Típuskód*, Szín)  
 KOCSITÍPUS (*Típuskód*, Férőhely)

Konvenciók: Emlékeztetünk arra, hogy az azonosítót félkövér szedet mutatja. Ha egy tulajdonság az egyik egyedben azonosító, a másikban nem az, akkor ezt dőltbetű jelzi. A többi tulajdonság nem kerül kiemelésre.

+ **Az egyed kapcsolatainak az együttesét az egyed külső szerkezetének hívjuk.**

Magyarázat: A 3.1 példa első két mondatát a típus alapján összekapcsolva megtudjuk, hogy Rózsa kocsija ötszemélyes. A 3.2 példában ez úgy tükröződik, hogy az első egyedben lévő *Típuskód* a másodikban azonosítóként megismétlésre kerül. A két egyed ezért nem független egymástól: közöttük meghatározható a KOCSITÍPUS - KOCSI kapcsolat. A kapcsolatnevet pedig nem véletlenül írtuk ebben a sorrendben. A nevekben elől szerepel a fölérendelt, hátul pedig az alárendelt egyed (→ *Kapcsolatjellemzők*).

Az adatmodell a fentiek szerint *hiperstruktúra*, azaz szerkezetek szerkezete: az egyedek belső és külső felépítésének a szervezett együttese. A 3.2 példa kiemeléseiből látszik, hogy ebben az összetett struktúrában a tulajdonságok nem azonos feladatokat látnak el. Erről bővebben a következő két pontban szólnunk.

### 3.3 Relatív szerep

+ **Relatív szerepnek a tulajdonságnak az adott egyed belső szerkezetében ellátott szerkezeti funkcióját nevezzük.**

**3.3 példa** KOCSI (**Rendszám**, Tulajkód, *Típuskód*, Szín)  
 KOCSITÍPUS (*Típuskód*, Férőhely, Fogyasztás, Díjtétel)  
 TULAJDONOS (**Tulajkód**, Tulajnév, Tulajcím)

Magyarázat: A fogalmi modellben a tulajdonságoknak az egyeden belüli elhelyezése közömbös. (NB.: Viszont a fizikai tervben ez a sorrend befolyásolhatja a hatékonyságot!) A TULAJDONOS egyedben a Tulajcímet a Tulajnév elé is írhattuk volna. Általános szokás, hogy az azonosítót előre

szoktuk venni. Bár ennek nincs strukturális jelentősége, mégis azt mutatja, hogy azt a tételt a többiekénél fontosabbnak tartjuk (← *Egyedszerkezet*).

Az *azonosító* funkciója - szerepe - a hivatkozás, az egyedek egyértelmű behatárolása. A többi tulajdonság feladata egyszerűbb és azt leírásnak nevezzük. A KOCSI egyedben a Rendszám tulajdonság azonosító, a Típuskód *leíró* relatív szerepű. A „relatív” jelző „az adott egyedhez képest”-it jelent. A Típuskód a KOCSI egyedhez képest leíró, ugyanakkor a KOCSITÍPUS egyed belső szerkezetében azonosító szerepű.

A kettős feladatokból következik, hogy a szerepeket majd más aspektusból tovább kell vizsgálnunk (→ *Abszolút szerep*, *Kombinált szerep*).

Korlátok: Az egyed belső szerkezetére több általános strukturális megkötés vonatkozik:

- K1 Minden egyednek kell, hogy legyen azonosítója.
- K2 Az azonosító értéke egyetlen egyedelőfordulásban sem lehet üres/ismeretlen.
- K3 Minden egyednek csak egy azonosító tulajdonsága lehet.
- K4 Ugyanaz a tulajdonság csak egyetlen egyednek lehet az azonosítója.

Magyarázat: A négy korlát egyetlen egyben is megfogalmazható. Az egyedek és az azonosítók között típus és előfordulás síkon egyaránt szigorúan és egyértelműen kölcsönös viszonyoknak kell fennállnia. A leírókra - egyelőre - nem adunk meg korlátot. Egy egyedben tetszőleges számú és tartalmú leíró szerepű tulajdonságot alkalmazhatunk.

### 3.4 Abszolút szerep

+

**Abszolút szerepnek a tulajdonságnak az egyedek külső szerkezetében ellátott szerkezeti funkcióját nevezzük.**

**3.4 példa** KOCSI (**Rendszám**, Tulajkód, Típuskód, Szín)  
 KOCSITÍPUS (**Típuskód**, Férőhely, Fogyasztás, Díjtétel)  
 TULAJDONOS (**Tulajkód**, Tulajnév, Tulajcím)

Magyarázat: Itt a 3.3 példát ismételtük meg. A Tulajkód tétel az első egyedben leíró, a harmadikban azonosító *relatív szerepű*. Ez a tulajdonság nem csak arra szolgál, hogy a KOCSI egyedből kiválogassuk a leíró értékét, hiszen az önmagában nem is mond semmit. Valódi funkciója az, hogy átléphessünk általa a TULAJDONOS egyedbe, hogy kikeressük onnan a tulajdonos nevét és/vagy címét. Az utóbbi egyedben az adat funkciója kettős. Az egyed *azonosítója* és egyben arra is szolgál, hogy kikereshessük vele a tulajdonosokhoz tartozó kocsikat. Ezt a kétirányú egyedek közötti közlekedést szemlélteti a következő ábra két kiemelt sora.

KOCSI				TULAJDONOS		
Rendszám	Szín	Típuskód	Tulajkód	Tulajkód	Tulajnév	Tulajcím
X	Fehér	T1	T1	T1	Rózsa	C1
Y	Zöld	T1	T2	T2	Gabi	C2
Z	Fehér	T2	T3	T3	Lajos	C3
Q	Piros	T3	T1			
W	Piros	T2	?			

3.1 ábra: Mintaadatbázisunk részlete

Amikor csak a KOCSI egyeddel törődünk, a Tulajkód közönséges leíró tulajdonságnak látszik. Nem ütközik ki az a feladata, amit az egyedek közötti struktúrában tölt be. Mivel a Szín alapján nem tudunk más egyed felé mozogni, a Tulajkóddal pedig ezt megtehetjük, a kétféle leíró tulajdonság között mégiscsak különbséget kellene tenni. Erre alkalmas az abszolút szerep.

Az abszolút szerep a tulajdonság legfontosabb relatív szerepe. Mivel a Tulajkód adott egyedben azonosító és ez fontosabb, mint a másik egyedben játszott relatív szerep, a tétel abszolút szerepe azonosító. Szemben a Színnel, amelynek abszolút szerepe csak leíró. Ez utóbbi pedig egyet jelent azzal, hogy az egyed külső szerkezetében nincs feladata.

A kettős feladatok miatt a leíró tulajdonságokat két csoportra kell osztanunk. Vannak olyanok, amelyeknek az abszolút szerepe is leíró (Szín). Vannak olyanok is, amelyek abszolút szerepe viszont azonosító (Tulajkód). Az utóbbiakat **kapcsoló** relatív szerepű tulajdonságoknak tekintjük.

### 3.5 Kombinált szerepek

Háttér: Az előző pont végén rámutattunk arra, hogy egy azonosító **abszolút szerepű** tulajdonság az egyik egyedben ugyanilyen, a másikban leíró **relatív szerepet** tölthet be. Ekkor valójában kapcsoló funkciót lát el. Hasonló kettős szerepek más összefüggésekben is előfordulhatnak.

**3.5 példa** KOCSI (**Rendszám**, Tulajkód, Típuskód, Szín)  
KÁRESEMÉNY (**Kárszám**, Kárdátum, Kárhely)  
KOCSI KÁRA (**Kárszám**+**Rendszám**, Kárösszeg)  
KÁRTÉTEL (**Kárszám**+**Tételsorszám**, *Rendszám*, Kárösszeg)

Magyarázat: Vannak olyan egyedek, amelyek azonosítója **elemi** [elementary key], azaz egy tételből áll (KOCSI). Máskor a kulcsot más egyedek kulcsainak az összekapcsolásával nyerjük (KOCSI KÁRA). Ekkor **konkatenált** kulcsról [composite key] beszélünk. Végül előfordul, hogy a többrészes azonosító valamelyik része nem azonosító másik egyedben (KÁRTÉTEL, ami nem külön egyed, hanem a KOCSI KÁRA másként szerkesztett változata). Ekkor a kulcs **összetett** [compound key]. Az összekapcsolt és az összetett azonosítók részeit a „+” jellel kötötten mutatjuk.

A Rendszám a KOCSI KÁRA egyedben nem azonosító relatív szerepű. Ezt a funkciót az összetétel látja el. Ugyanakkor nem is csak leíró a relatív szerepe, mert annál fontosabb a feladata: részt vesz az azonosításban. Az összetételt alkotó tulajdonságok **azonosítórész** relatív szerepűek. A Rendszám a KOCSI KÁRA, a Tételsorszám a KÁRTÉTEL egyedben ugyanezt a viszonylagos feladatot látja el, mégis látható közöttük egy fontos eltérés. Van olyan egyed, amiben a Rendszám kulcs, de nincs olyan, amiben a Tételsorszám azonosító funkciójú. Tehát az előbbi abszolút szerepe azonosító. Az utóbbié nem ez és nem is leíró. Mivel az abszolút a legfontosabb relatív szerep, a Tételsorszám olyan sajátos tulajdonság, amelynek az abszolút szerepe is azonosítórész.

Pontosítani kell a relatív szerepeket is. A KÁRTÉTEL-ben a Kárszám és a Rendszám nemcsak a belső, hanem a külső szerkezetben ( $\leftarrow$  *Egyedszerkezet*) is feladatot lát el. A két tétel egyaránt kapcsoló, de az első **kulcsrészként-kapcsoló**, a második **leíróként-kapcsoló**.

Korlát: Az ilyen kombinált szerepek alkalmazása nem öncélú. Fennáll az alábbi korlát:

*K5 Az összetett azonosító részei egyetlen egyedelőfordulásra nézve sem lehetnek üres vagy ismeretlen értékűek.*

Ez pedig egyet jelent azzal, hogy amennyiben egy tulajdonság csak leíróként-kapcsoló, úgy az lehet üres értékű is, vagyis a kapcsolat lehet opcionális is. Ha viszont a tulajdonság kulcsrészként-kapcsoló, akkor nem lehet üres értékű és a ráépülő kapcsolat mindenképpen kötelező jellegű. A jellegre nézve lásd a 3.7 pontot (→ Kapcsolatjellemzők). A kapcsoló szerep fontosságát a következő pontban világítjuk meg.

### 3.6 Kapcsolótulajdonság

+

**Két egyed akkor és csak akkor áll kapcsolatban egymással, ha az egyik kapcsoló szerepű tulajdonságként tartalmazza a másik azonosító szerepű tulajdonságát. Ezt a tétel kapcsolótulajdonságnak hívjuk.**

**3.6 példa** KOCSI (Rendszám, Tulajkód, ...)  
TULAJDONOS (Tulajkód, ..., Kárszám)  
KOCSI KÁRA (Kárszám+Rendszám, ...)

Magyarázat: A Rendszám és a Tulajkód **abszolút szerepe** azonosító. Az előbbi **relatív szerepe** a KOCSI KÁRA egyedben kulcsrészként-kapcsoló, az utóbbié a KOCSI egyedben leíróként-kapcsoló. Ezért a KOCSI - KOCSI KÁRA (Rendszám) és a TULAJDONOS - KOCSI (Tulajkód) összefüggések olyan kapcsolattípusok, amelyek a zárójelben mutatott kapcsolótulajdonságokon alapulnak.

A Kárszám tulajdonság nem való a TULAJDONOS egyedbe. Itt csak didaktikai okokból szerepeltetjük. Bár a két utolsó egyednek van közös tulajdonsága, a Kárszám egyikben sem azonosító szerepű, nem kapcsoló adat és a két egyed között nem létezik kapcsolat. Az ugyanis nem lenne egyértelmű, amint arra azonnal rámutatunk.

KOCSI		KOCSI KÁRA		TULAJDONOS	
Rendszám	Tulajkód	Kárszám	Rendszám	Tulajkód	Kárszám
X	T1	K1	X	T1	K1
Y	T2	K1	Y	T2	K1
Z	T3	K2	Y	T3	K2
Q	T1	K2	Z		
W	?	K3	Q		

3.2 ábra: Többértelmű viszonyok

Probléma: A Kárszám azért van rossz helyen a TULAJDONOS egyedben, mert csak egy érték jelölhető ki hozzá. Ezért a T1 tulajdonosnál nem tudunk hivatkozni a K3 számú kárra. Azonban a dolognak most nem ez az oldala a lényeges. Ha a K1 Kárszám alapján a T1 tulajdonost összekapcsoljuk a megfelelő kocsikárokkal, akkor belebotlunk a K1 - Y kulcspárosba is (!), márpedig a T1 tulajdonosnak semmi köze sincs az Y kocsihoz.

**Figyelmeztetés!** A relációs rendszerek nem zárják ki, hogy a táblákat a nem-kapcsoló adatokon is összekössék egymással. Ebből pedig inkorrekt ismeretek származnak.



Mellérendelt (TULAJDONOS és KOCSI KÁRA) egyedek nem állhatnak kapcsolatban. Az egymáshoz rendelési viszonyokat a következő pontokban tárgyaljuk.

### 3.7 Kapcsolatjellemzők

**3.7 példa** TULAJDONOS - KOCSI  
TULAJDONOS - TULAJDONOS  
TULAJDONOS - SZERVEZET

**Kapcsolatnem:** Az első esetben külön nemű tényezők viszonyát neveztük meg. A kocsi és a tulajdonos természete szerint két eltérő dolog. Az ilyen viszony „inhomogén”, amit *birtoklási* [has-a] kapcsolatnak hívunk. A kocsinak van tulajdonosa. A második esetben azonos nemű dolgokról van szó. Az ilyen „homogén” viszonyban egyazon egyedtypus előfordulásai kapcsolódnak egymáshoz, ezért azt *viszszamutató* [recursive v. envoluted] kapcsolatnak hívjuk. A harmadik páros kvázi-egynemű dolgokat takar. A tulajdonosokat magánszemélyekre és szervezetekre osztályozhatjuk. A szervezet is egy kocsitulajdonos, a lehetséges tulajdonosok egyik altípusa. Ez a kapcsolat osztályozási, másképpen szólva *is-egy* [is-a] természetű viszony.

**Kapcsolatfok:** Ez a számszerű viszony azt mutatja meg, hogy egy egyedtypusnak hány előfordulása kapcsolódhat a másik (az előbbivel esetleg azonos) egyedtypus egy tételéhez és megfordítva. Nem konkrét számokról van szó, hanem csak arról, hogy a kapcsolódás több elemet is érinthet-e, vagy csak egy elemre korlátozott.

Egy (1) tulajdonosnak lehet több (N) kocsija is, de egy kocsi csak egy tulajdonoshoz tartozhat. A TULAJDONOS - KOCSI kapcsolat foka  $I:N$ . Ebben az esetben a kapcsolatot hierarchikusnak hívjuk. Ekkor az első egyedet fölérendeltnek, a másodikat alárendeltnek tekintjük. A tulajdonosok a kocsik fölérendeltjei. A kapcsolat felülről-lefelé mutat. Ezért az irányt kifejezendő a kapcsolat nevében mindig előre írjuk a fölérendeltet.

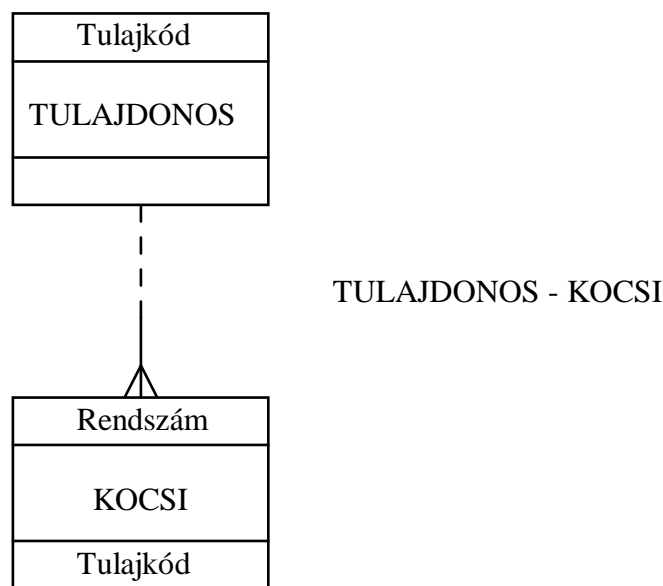
Egy (1) tulajdonos vagy egy (1) személy, vagy egy szervezet. A személy és a szervezet nem lehet több tulajdonos. A TULAJDONOS - SZERVEZET kapcsolat foka  $I:I$ . Ez a viszony altípus összefüggést fejez ki, ezért azt altípus-hierarchiának tekintjük, amelyben még mindig az első tag a fölérendelt, a második az alárendelt. Ha a tulajdonosok egymás közötti viszonya például az éppen érvényes házastársi kapcsolatokat mutatná, akkor a TULAJDONOS - TULAJDONOS kapcsolat foka is  $1:1$  lenne. Mivel azonban a házastársi kapcsolatnak nincs iránya, nem beszélhetünk hierarchiáról. Ilyen esetekben a kapcsolatot alkotó egyedek egymás mellérendeltjei.

Egy kocsi több (M) káresemény részese lehet és egy káreseményben több kocsi (N) vehet részt. Ha két jelenség egymással  $M:N$  fokú viszonyban áll, akkor nem hozhatók egymással közvetlen kapcsolatba, nincsenek alá/fölérendeltségi viszonyban. Az idevágó további tudnivalókat lásd a későbbiekben ( $\rightarrow$  *Hálós viszony*).

**Kapcsolaterő:** A kapcsolatfok csak azt árulja el, hogy az egyik egyedhez legfeljebb 1 vagy legfeljebb N másik egyedelőfordulás kapcsolódhat. Arra nem utal, hogy léteznie is kell-e a kapcsolatnak. Az „egy tulajdonoshoz több kocsi (N) tartozhat” kitétel nem mondja ki, hogy csak olyan tulajdonosokra vagyunk kíváncsiak, akiknek van ma ismert és nyilvántartott kocsija. Az adatbázisban vezethetünk olyan tulajdonosokat is, akiknek már vagy még nincs autója. Ezért az N értéke lehet nulla is. A dolog fordítva is igaz. Az „egy kocsi csak egy (1) tulajdonoshoz tartozhat” kitétel nem zárja ki, hogy nyilvántartunk pl. olyan megtalált autót, amelynek a tulajdonosa ismeretlen. Ezért az 1 érték úgy értendő, hogy 1 vagy 0.

A kapcsolat egyetlen lényeg. Ezért a függéserő - amit kapcsolatopcionálisnak is hívunk - négyféle értéket vehet fel. Vannak kétoldalról **kötelező** illetve kétoldalról **opcionális** viszonyok. Minden tulajdonosnak kell/nem kell hogy legyen kocsija és minden kocsi tulajdonoshoz kell/nem kell hogy tartozzon. A viszony vagy alulról (kocsi), vagy felülről (tulajdonos) opcionális is lehet, ami egyet jelent azzal, hogy ugyanakkor felülről (tulajdonos) illetve alulról (kocsi) ugyanaz a viszony egyszersmind kötelező.

**Figyelmeztetés!** Egyes tervezési segédletek használatakor a kapcsolat fokát és erejét kétszeresen, az egyik illetve a másik egyed oldaláról nézve kell megadni. Az sem ritka, hogy a függéserőt csak a fölérendelt oldaláról lehet minősíteni azt feltételezve, hogy az alárendelthez mindig tartozik fölérendelt. Ezek elvi hibák.



3.3 ábra: Hierarchikus inhomogén kapcsolat

Magyarázat: Az ábra az ismereteket összegzi. A TULAJDONOS - KOCSI kapcsolat inhomogén birtoklási viszony. A kapcsolat foka 1:N (lásd varjúláb). Csak olyan kocsit tartunk nyilván, amelynek ismert a tulajdonosa, de ez megfordítva nem igaz. Ezért a kapcsolat függésereje felülről-opcionális/alulról-kötelező.

### 3.8 Ismétlődő csoport

Háttér: Az adatmodellben a tulajdonságokat mindig **elemi** tényezőkként kell megadni. Ez azt jelenti, hogy azokban nem vonható össze több jelentés. Ha például a Számlaszám első pár jele a vevőre utaló vevőkód, a többi jele pedig relatív sorszámot alkot, akkor a nevezett tényező nem tulajdonság (fogalmi szint), hanem adatmező (logikai szint).

A fogalmi modellben lehetőség van **összetett** tulajdonságok meghatározására is, feltéve, hogy az összetételt explicit módon - az elemeket felsorolva - fejezzük ki. Az ilyen önálló névvel ellátott összetett tulajdonságokat **csoportoknak** nevezzük.

**3.8 példa** Dátum{Év, Hó, Nap}  
 Számlaszám{Vevőkód+Relatív számlasorszám}  
 {Rendszám,Kárszám}  
 Szállítási határidő

Magyarázat: A csoport nevét a csoporttagok „{ }” jelek közötti felsorolása követi. Az utolsó két tétel nem alkot csoportot, mert a harmadikból hiányzik annak explicit neve, a negyedikből pedig annak explicit felbontása. A csoportok háromfélék lehetnek. A Dátum *természetes* csoport, a Számlaszám *mesterséges* összetétel. A harmadik lehetőséget külön is ki kell fejtenünk.

+

Azon tulajdonságokat, amelyek egy egyedelőfordulásra több értéket is felvehetnek, ismétlődő adatoknak hívjuk. Ha az adat összetett, akkor ismétlődő csoportról [repeating group] beszélünk.

TULAJDONOS

Tulajkód	Tulajnév	Típuskód	Férőhely
T1	Rózsa	K1	5 fő
		K3	4 fő
T2	Gabi	K1	5 fő
T3	Lajos	K2	5 fő

3.4 ábra: Ismétlődő csoportot tartalmazó egyed

Korlát: Az ismétlődő adatok/csoportok számos probléma forrásai. Ezeket később fogjuk részletesen kifejezni (→ Kiegyensúlyozatlanság). Itt egyelőre csak meg kell tanulni, hogy:

*K6 Az egyedekben nem szerepelhetnek ismétlődő tulajdonságok illetve csoportok.*

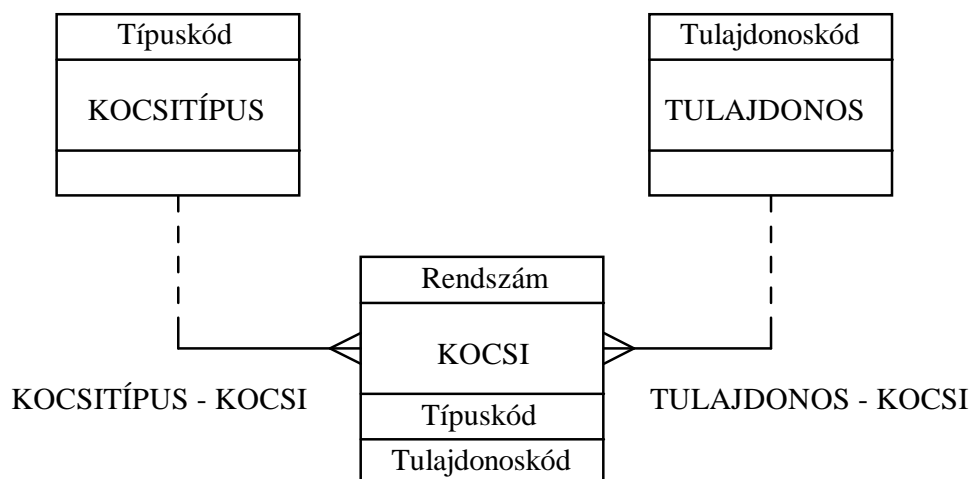
### 3.9 Hálós viszony

Definíció: Ha két egyedtípus előfordulásai 1:1 fokú viszonyban állnak egymással, akkor **lineáris**, ha 1:N fokú összefüggésben akkor **hierarchikus**, ha M:N fokúban, akkor pedig **hálós** szerkezetéről beszélünk. Ugyanígy szerkezeti elemekről van szó akkor is, ha egy egyedtípus eltérő előfordulásai között állnak fenn hasonló fokú kapcsolatok.

**3.9 példa** KOCSITÍPUS (Típuskód, {Tulajkód}, ...)  
 TULAJDONOS (Tulajkód, {Típuskód}, ...)

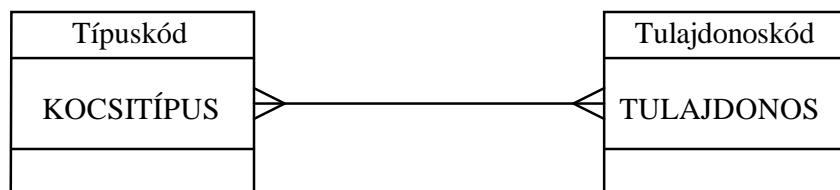
Magyarázat: Egy (1) tulajdonosnak eltérő típusú (N) kocsijai lehetnek és egy (1) típust több (M) tulajdonos is birtokolhat. Ezért az egyik egyedben a Tulajkód, a másikban a Típuskód tétel nem egyetlen, hanem több értéket vesz fel. Végeredményben a két egyed előfordulásai között az összefüggés M:N fokú és a megfelelő szerkezeti elemet a definíció szerint hálósnak tekintjük.

Probléma: A két leíró tulajdonság ismétlődő adatot ( $\leftarrow$  *Ismétlődő csoport*) alkot. Ez több baj forrása ( $\rightarrow$  *Kiegyensúlyozatlanság*), amiből itt csak kettőt említettünk. A 3.9 példa minden ismeretet kétszeresen tartalmaz, mert az egyik egyedben a tulajdonoshoz kötött típus a másikban típushoz kötött tulajdonosként jelentkezik. Problémát okoznak a számok is. Van akinek csak egy, a nagyvállalatnak pedig több tucat típusú kocsija lehet. Ezt a két szélsőséget nem lehet jól egyetlen egyedben tükrözni.



3.5 ábra: Hálós egyedviszony feloldása harmadik egyeddel

Magyarázat: Az M:N fokú viszonyok nem tükrözhetők korrekten egy adott tulajdonság értékeivel, ezért ezeket nem is hívjuk kapcsolatoknak. Az ilyen összefüggéseket mindig felbontjuk egy harmadik egyed segítségével. Erre az ad lehetőséget, hogy az M:N-es fok felfogható két 1:N-es (1:N és 1:M) együttesének: egy kocsitípushoz és egy tulajdonoshoz egyaránt több kocsi tartozik. Ezzel elkerülünk egy igen sajátos redundanciát is. A KOCSI egyed mindenképpen szükséges, még hozzá a mutatott módon. Ha a tulajdonosoknál vezetnénk a típuskódot és/vagy a típusoknál a tulajdonoskódot, akkor a típus-tulajdonos párt megdupláznánk, hiszen az a KOCSI egyedben amúgy is rendelkezésünkre áll.



3.6 ábra: Hálós viszony

Magyarázat: A végső adatmodellben ilyen „em-az-ennes” megoldás nem szerepelhet. Azonban a modellezés elején készíthetünk olyan *előzetes* globális modellt, amelyben még nem fejtünk ki pontosan minden részletet. Abban a 3.6 ábra megoldása még megengedett úgy, hogy azt a végső modellben a 3.5 ábrának megfelelően oldjuk fel.

**Figyelmeztetés!** Egyes módszerek a végső modellből sem zárják ki a hálós viszonyt.

Kiegészítés: Ez az engedékenység nemcsak gyakorlati, hanem elméleti okok miatt is hibás. Az egyed definíciója szerint az ismeretekkel leírandó jelenségeket ilyen szerkezeti elemmel kell tükrözni. Mármint a kocsitípus-tulajdonos viszony leírható bennünket érintő ismeretekkel, hiszen ez a viszony magában a kocsiiban testesül meg. Ezért a hálós viszony harmadik egyeddel való felbontása elméletileg nem váltható ki más megoldással.

### 3.10 Mellérendelt viszony

— *Az adatmodellben nem alkalmazunk mellérendelt viszonyokat.*

Magyarázat: Az előző pontban kimutattuk, hogy a modellben az M:N fokú viszonyokat külön egyed bevezetésével le kell bontani két hierarchikus, vagyis fölő/alárendeltségi viszonyra. Ez nemcsak a szerkezetet érintő korlát (amit azért nem fogalmaztunk meg külön feltételként, mert a kapcsolat meghatározása ezt megteszi), hanem a kezelésben is érvényesítendő elv.

DOLGOZÓ		GÉP	
Törzsszám	Költséghelykód	Gépazonosító	Költséghelykód
T1	K1	G1	K1
T2	K2	G2	K1
		G3	K2
		G4	K1

3.7 ábra: Mellérendelt egyedek

Probléma: Egyik egyed sem tartalmazza a másik kulcsát, ezért nincs köztük kapcsolat. Ennek dacára a közös Költséghelykód alapján valaki egy adott eljárásban megkísérelheti összekapcsolni a két egyed típus előfordulásait. Ekkor hamis ismeretre fog szert tenni. A T1 kulcsú dolgozó ugyanis csakis a G1 és a G4 gépekkel dolgozik. Viszont a keresés azt mutatja, mintha valami köze lenne a G2 géphez is.

HASZNÁLJA			
Törzsszám	Gépazonosító	...	Műszak
T1	G1		M1
T1	G4		M3
T2	G3		M2

3.8 ábra: Az eredeti két egyed típus korrekt összekapcsolása

Magyarázat: Ha M:N-es kapcsolati ismeretre van szükség, akkor azt sohasem érhetjük el *leíró relatív szerepű* tulajdonság (Költséghelykód) alapján. A mellérendelést ekkor is csak egy harmadik egyeddel lehet feloldani. A DOLGOZÓ - HASZNÁLJA (Törzsszám) kapcsolat mentén tudhatjuk meg, hogy a T1 munkatárs milyen gépekkel dolgozik.

### 3.11 Kölcsönös viszony

Definíció: A kölcsönös viszony a **mellérendelt viszony** olyan sajátos válfaja, amelyben az egyedek 1:1 lineáris - tehát nem M:N hálós - összefüggésben állnak egymással.

**3.9 példa** FÉRJ (**Férjkód**, Név, Foglalkozáskód)  
FELESEG (**Feleségkód**, Név, Foglalkozáskód)

KOCSI (**Rendszám**, Típus, Tulajdonoskód, *Cascoszám*)  
CASCO (*Cascoszám*, Típus, Alapdíj, *Rendszám*)

Probléma: Az első egyedpáros „egy-az-egy” viszonyban áll egymással. Ilyen modellt sohasem szabad tervezni. Nem csak azért nem, mert a tulajdonságsorok átfedőek és a kifelé (pl. FOGLALKOZÁS) mutató kapcsolatok is azok. Hanem azért sem, mert a befelé mutató kapcsolatok is többszöröződnek. Például a KOCSI tulajdonosa hol a FÉRJ, hol a FELESEG lesz. Ha pedig netán létezik egy TULAJDONOS egyed, akkor a redundancia mértéke már óriási. Mivel „homogén” dolgokról - személyekről - van szó, ilyen esetekben az egyetlen helyes megoldás az egyedek összevonása (→ Generalizáció és specializáció).

A második egyedpáros hasonló problémákat vet fel, de annál nem „homogén” dolgok állnak 1:1 fokú viszonyban. Ráadásul ez az összefüggés az egyik oldalon opcionális: nem minden kocsinak van Casco-ja. Éppen ezért ilyenkor a mellérendelést nem feltétlenül a két egyed összevonásával célszerű feloldani.

**3.10 példa** KOCSI (**Rendszám**, Típus, Tulajdonoskód)  
CASCO (*Cascoszám*, Alapdíj, *Rendszám*)

Megoldás: Az átfedéseket (Típus) ki kell küszöbölni. El kell hagyni a kölcsönös utalást is; jelen esetben az első egyedből az opcionális Cascoszámot. Végeredményben alkotunk egy olyan mesterséges hierarchiát (KOCSI - CASCO), amelyben az alárendeltek száma maximum egyre korlátozott. Ilyenkor az alárendeltet **kiegészítő egyednek** nevezzük, mert a CASCO adatsora mintegy kiegészíti a KOCSI tulajdonságsorát. A technikai megoldás hasonlít egy másik konstrukcióra (→ Egyedaltípus), de tartalmi lényege más.

**Figyelmeztetés!** A kezelők nem ismerik ezt a szerkezetet. Ezért explicit korlátait csak implicit módon - procedurális feltétellel - tudják támogatni.

### 3.12 Ellenőrző kérdések

3/1 Ön szerint az alábbi állítások közül melyik igaz (I) és melyik hamis (H):

- Van egy MAGNÓSZALAG és egy HANGLEMEZ állományom. Mivel mind a kettőben szerepelnek ugyanazok a zeneszámok, van egy jó adatbázisom.
- A fenti két állományomat bármikor összeválogathatom pl. a „Richard” név szerint és ezzel megfelelő ismeretekhez jutok.

- A két egyedtípus belső és külső szerkezete összefügg. Ha van egy külön SZEMÉLY állományom, akkor abból indulva lekérdezhetem azokat a szalagokat és lemezeket, amelyeken Cliff Richard énekel.
- 3/2 Ön szerint milyen viszonyban állnak egymással a következő jelenségek? Hálós (S), hierarchikus (H) vagy lineáris (L) megjelölést alkalmazzon a viszony 1:N, M:N illetve 1:1 foka szerint.
- Kocsi és CASCO-biztosítása.
  - Személyek és nyelvek.
  - Számla és az azon lévő tételek.
- 3/3 Adott a RENDELÉS a Rendelésszám, -dátum és Számlaszám tételekkel, továbbá a SZÁMLA egyed a Számlaszám és Számlaegyenleg tulajdonságokkal. Adja meg a tulajdonságok relatív és abszolút szerepeinek a párosait. A kulcsot A, a leíró L, a kapcsolót K jelölje. Például a SZEMÉLY-ben a Személynév szereppárosa LL lenne.
- RENDELÉS/Rendelésszám
  - RENDELÉS/Rendelésdátum
  - RENDELÉS/Számlaszám
  - SZÁMLA/Számlaszám
  - SZÁMLA/Számlaegyenleg
- 3/4 Ön szerint milyen a számla és az azon lévő tételek közötti viszony? Az alábbi számok egyikét adja meg:
- A számla felől kötelező, a sor felől nem az (1).
  - A sor felől kötelező, a számla felől nem az (2).
  - Mindkét oldalról kötelező (3).
  - Mindkét oldalról opcionális (4).
- 3/5 Ön szerepeltetné-e a SZEMÉLY egyedtípusban a Képzettség tulajdonságot az alábbiak szerint? A helyes válasz sorszámát adja meg.
- Nem, mert egy személynek több képzettsége is lehet.
  - Ismétlődő csoporttal megoldható a dolog.
  - Mindez nem probléma, ha csak a legmagasabb képzettséget veszem fel.
- 3/6 Adja meg a válasz sorszámával, hogy milyen következtetést vonna le Ön abból, ha a SZEMÉLY egyedek annak Dátum (értsd: születési dátum) tulajdonságán összekapcsolva a RENDELÉS egyedek (Rendelésdátum) kiderülne, hogy mindig Kovács születésnapján rendelnek eperfagyit:
- Kovács szereti a fagyaltot.
  - Kovács csakis az eperfagyaltot szereti.
  - Lehet, hogy Kovács epres, de két egyedek leírón nem szabad kapcsolni.
- 3/7 Adott a SZÁMLA egyedtípus Számlaszám azonosítója, amely voltaképpen a Vevőkód értékéből és egy Sorszámból áll. Legyen kedves és adja meg a három tétel abszolút és relatív szerepét az azonosító (A), kulcsrész (R) és kapcsoló (K) megjelölésekkel. Például a RENDELÉSTÉTEL Rendelésszám és Cikkszám összetett azonosítójában az utóbbi jele AK lenne, mivel azonosító a CIKK egyedben, de ugyanakkor kapcsoló is odafelé a RENDELÉSTÉTEL-ben.
- 3/8 Mit tenne Ön akkor, ha olyan számla érkezne be, amelyen a Számlaszám értéke részben elmosódott? A helyes válasz sorszámát adja meg.
- Kidobnám a számlát.
  - A felismerhető karakterekkel vinném be a számlát.
  - Mesterséges azonosítót - például napi dátumot és sorszámot - alkalmaznék.
  - Külön állományba tenném a kétséges tételt annak tisztázásáig.

## 4. SZERKEZETI FINOMSÁGOK

### 4.1 Az adatmodell sokszínűsége

Feltehetőleg mindenki ismeri a felülről-lefelé történő megbontás és a fekete-doboz elvét. Az előbbi azt jelenti, hogy először a magasabb szintű konstrukciókat kell tisztán látnunk és csak azután törődhetünk a mélyebb részletekkel (vertikális gondolkodás). Az utóbbi azt sugallja, hogy bármelyik szinten is gondolkodunk, az adott lényeg tartalma egyre több sajátosság megismerése által válik világosabbá (kívülről-befelé gondolkodás).

*Ennek a fejezetnek az a célja, hogy a modellel kapcsolatos alapvető ismereteinket tovább finomítsuk és újabb szerkezeti tényezőket feltárásával gazdagítsuk.*

A fentiek értelmében két dologról van szó. Egyrészt arról, hogy újabb adatmodellezési tényezőket kell feltárunk. Másrészt arról, hogy - az előbbtől egyáltalán nem függetlenül - a régiakat pontosítanunk kell.

Mindaddig a **tulajdonságot** egyféle lényegnek tekintettük. Most rá kell mutatnunk arra, hogy e tényezőnek van abszolút és relatív értelme is (4.2 **Értéktartomány**). Fogalmaink tágabb-szűkebb - minősített - tartományokat ölelnek át. A ház is épület, de nem minden épület ház, a társasház pedig nem csak egyféle fizikai elrendezést jelent (4.3 **Szerepnév**). A nem tiszta gondolkodású tervezők visszaélnek azzal a ténnyel, hogy a fogalmakat nevek tükrözik és rossz szerkezeteiket a jónak vélt nevek mögé rejtik (4.4 **Ál-szerepnév**).

A korábbiakban már láttuk, hogy az egyedek kapcsolatai azok tulajdonságain alapulnak ( $\leftarrow$  **kapcsolótulajdonság**). Eddig azt feltételeztük, hogy két eltérő lényegű egyed között kell kapcsolatot kialakítani egy közös (nevű) tulajdonság alapján. Azonban vannak olyan viszonyok is, amelyeket egy egyedtípus előfordulásai között kell meghatározni egyazon lényegű, de eltérő nevű tulajdonságok szerint. A viszony foka szerint másféle modellezési megoldásokat kell választanunk (4.5 **Visszamatató kapcsolat** és 4.6 **Családja**).

A leendő adatbázis tartalma - pontosabban egyes tartalmak hiánya - visszahat magára a modellre. Ezért komolyan kell foglalkozni az értelmezhetőség kérdésével (4.7 **Üres érték** és 4.8 **Egyedaltípus**).

Az adatbázis és az azt meghatározó adatmodell alapvető titka a kapcsolatokban rejlik. Nem ott és nem úgy tudunk egymással összefüggésbe hozni információvá kapcsolható ismereteket, ahol és ahogyan az nekünk tetszik. A minden szempontból korrekt viszonyok meglehetősen merev, de ugyanakkor sokszínű sémákon alapulnak. Ezeknek némelyikét már a korábbi fejezetek ismertették, másik részüket az alábbi pontokban mutatjuk be. A szabályokat a 4.9 **Korrekt és inkorrekt kapcsolatok** cím alatt összegezzük.

### 4.2 Értéktartomány

+

Az értéktartomány [domain] az adott jelentésű tulajdonság általánosan felvehető értékeinek a halmaza.



**4.1 példa** KOCSI (**Rendszám**, *Típuskód*)  
KOCSITÍPUS (*Típuskód*, Férőhely, ...)

RENDELÉS (**Rendelésszám**, ...)  
SZÁMLA (**Számlaszám**, ...)

Magyarázat: Más szerzők az adott egyedhez kötött sajátosságot *attribútumnak* hívják. Náluk a tervező (ha úgy látja jónak, erre semmi sem kötelezi) kijelölhet egy értékalmazt is a megengedett értékek korlátozására. Ők ezt a korlátot nevezik *doméjnnak*. Tehát pl. a Típuskód a fenti egyedekben attribútum és a tervező - ha kedve van - készíthet egy listát, amely a megengedett típuskód értékeket tartalmazza. A Rendelésszám és a Számlaszám attribútum, ha pedig mindkettő 1-99999 értéket vehet fel, akkor számukra kijelölhető egy közös Sorszám doméjn az értékek kontrolálására.

Probléma: Ez a megközelítés teljesen helytelen. Egyrészt a doméjn használata nem ízlés dolga, mert a tulajdonság egyedtől függetlenül is létező lényeg. Így például létezhet olyan kocsitípus, amely egyik egyedben sem szerepel még, de majd meg fog jelenni. A doméjn nem a felvett, hanem a felvehető értékek halmaza. Másrészt a tartomány nem választható el a jelentéstől. Holnap bővül a számlák száma, de nem a rendeléseké. A Számlaszámhoz már az 1-999999 értékeket kell megengedni, de a Rendelésszámokhoz még nem. A két tartományt utólag szét kell választani. Jobb lett volna, ha ezt eleve megtették volna.

Megoldás: A mi megközelítésünk korrektebb. Szerintünk *létezik egy abszolút és egy relatív tulajdonság* fogalom. Fenti definíciókban nagyon fontos az „adott jelentésű” rész. A Rendelésszám és a Számlaszám sohasem alkothat egy doméjnt, mert eltérő a jelentésük. Ugyanakkor a Típuskód doméjn mint adott jelentésű lényeg az egyedektől függetlenül is létező - *abszolút* - tulajdonság. Ennek az értéktartománynak ilyen-olyan valós/nemvalós részhalmazai kerülnek értékek-ként alkalmazásra a különböző egyedekben, tehát *relatív* - az egyedhez képest viszonylagos - módon. A KOCSITÍPUS egyedben leírhatunk olyan típusokat is, amelyekhez nem kapcsolódik a KOCSI egyedben előfordulás.

A mi koncepciónkban magát a sajátosságot (Típuskód) nevezzük tulajdonságtípusnak. Ha a sajátosság adott egyedben megjelenik, akkor *egyed-tulajdonság-viszonynak* hívjuk. Egy ilyen viszony a KOCSI/Típuskód, egy másik pedig a KOCSITÍPUS/Típuskód tétel. A doméjn/attribútum esetleges és ad-hoc módon alkalmazott párossal szemben a tulajdonság és az egyed-tulajdonság-viszony tudatos egymásra építése korrektebb modellstruktúrákra vezet. Lásd még a következő pontot is.

## 4.3 Szerepnév

+

Szerepnévnek hívjuk az egyeden belül sajátos értelmezésben használt általános tulajdonságot.

#### SZÁMLA

Számlaszám	Rendelő kódja	Fizető kódja
M	X	Y
N	Q	Z

4.1 ábra: Egy megfelelőnek látszó adatbázisrészlet

Magyarázat: Nagyon sokszor előfordul, hogy egy egyeden belül többször kell utalni azonos végső lényegű ismeretekre. A rendelő éppen úgy partner, mint a vele nem minden esetben meg-  
egyező fizető. A tervező nem alkalmazhatja kétszer egy tulajdonságsorban ugyanazt a Partnerkód  
nevet, amely ráadásul a partnerek szerepeire sem utalna. (NB.: Az itteni „szerep” lényegnek  
nincs köze a tulajdonság feladataihoz. Lásd  $\leftarrow$  Relatív szerep.) Ezért úgy tűnik, hogy a 4.1 ábra  
megoldása korrekt.

Probléma: A megoldás valójában hiányos. Nem tükrözi azt az általános tulajdonságot (lásd a  
definíció végét), amelyet sajátos kettős értelmezésben használtak az adott egyeden belül (lásd a  
definíció közepét). Ez az általános tulajdonság a Partnerkód, ami korábbi ismereteink szerint  
generikus tulajdonság, azaz *értéktartomány*. Nem az a baj, hogy az alkalmazott nevek nem  
utalnak arra, például „Rendelő partnerkód” módon. Semmi köze sincsen a szerepnévnek a név  
írásmódjához: a szerepnév nem az általános tulajdonságnak (partnerkód) az előlről minősített  
(rendelő) neve, noha bevett az efféle megoldás is. Baj az, hogy a modell formailag hiányos, mert  
nem rögzíti a Rendelő kódja  $\subseteq$  Partnerkód összefüggést. Azt a tényt, hogy a SZÁMLA/Rendelő  
kódja egyed-tulajdonság-viszony (relatív tulajdonság) sajátosan értelmezett Partnerkód (abszolút  
tulajdonság).

Kiegészítés: A szerepnév nem formai, hanem tartalmi lényeg. Van olyan modellezési irányzat  
[<sup>7</sup>], amely megkövetelné, hogy az általános tulajdonságot az összes egyedben viszonylagos  
tételként mindig minősítsék, azaz más névvel jelöljék. Ez *formai* megoldás, amely a legtöbb  
esetben nem segíti, hanem zavarja a megértést. Miért adnánk más nevet a Rendelésszámnak a  
RENDELÉS és a RENDELÉSTÉTEL egyedekben?

A szerepnevek két valódi *tartalmi* feladatot látnak el. Az egyik az érdemi minősítés, amire már  
a 4.1 ábra is példát mutatott. Mivel a rendelő és a fizető nem mindig azonos, minősíteni kell a  
partnernek a számlához való viszonyát. A másik feladat a szűkítés. A számlán megjelenhet a  
bankra vonatkozó ismeret is. A bank is partner, de nem minden partner bank. A bank tehát szű-  
kebb fogalom a partnernél. A szerepnevek mindkét felhasználásra bővebben is kitérünk a fejezet  
során.

## 4.4 Ál-szerepnév

Háttér: *Szerepnevet* a modellben két esetben kell alkalmazni. Akkor, ha a minősítés az egy-  
értelműség miatt elengedhetetlen és akkor, ha szűkítésre van szükség. Nem kizárt az sem, hogy a  
tervező az érthetőség miatt vezet be minősítést. Például a személynek csak egyféle neve van, azt  
mégsem a sima Név, hanem a minősített Személynév módon jelöli. A túlminősítéseket persze  
kerülni kell (ilyen a Személy anyja neve, hiszen Anyja neve csak személynek lehet), de most nem  
erről van szó. A tervezők minősített nevek mögé rejtenek alapvetően hibás struktúrákat.

**4.2 példa** SZEMÉLY (Személy azonosító, Nyelvkód-1, Nyelvkód-2, Nyelvkód-3)  
SZÁMLA (Számlaszám, Összeg-1, Összeg-2, Összeg-3)

Probléma: Az első egyedben a három nyelvkód nyilvánvalóan egyazon értékhalmazból kell, hogy származzon. A minősítés ezért látszólag korrekt. Valójában csak arra szolgál, hogy egy rossz struktúrát leplezzon (← *Ismétlődő csoport*).

A második egyedhez meg kell jegyeznünk, hogy a számszerűségeen kívül a háromféle összegnek semmilyen jelentésbeli köze sincs egymáshoz. Viszont ennek ellenkezőjét sejteti a minősítés. Úgy tűnik, mintha az adatok azonos értéktartományból származnának. Az adatmodellben kerülni kell az efféle hamis látszatokat. A három összegnek a tartalmakat megfelelően kifejező neveket kell adni.

## 4.5 Visszamatató kapcsolat

+

Visszamatató [involuted v. recursive] kapcsolatról beszélünk akkor, ha egy egyedtípus előfordulásai ugyanannak az egyedtípusnak más előfordulásaival állnak összefüggésben.

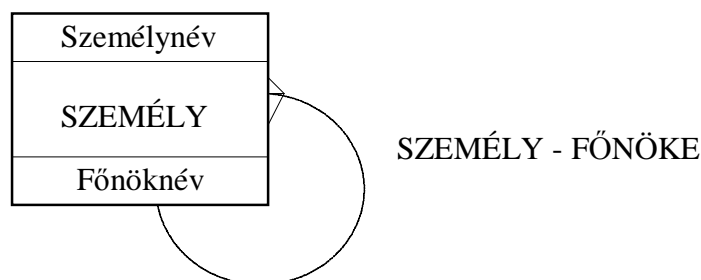
**4.3 példa** „Jóska főnöke Laci.”  
„Feri főnöke Laci.”  
„Laci főnöke Peti.”

SZEMÉLY

Személynév	Főnöknév
Jóska	Laci
Feri	Laci
Laci	Peti

4.2 ábra: Beosztottak és főnökök ismeretei

Magyarázat: Az előző fejezetben az *inhomogén*, vagyis eltérő lényegű egyedek közötti viszonyokról volt szó. Ebben a példában egymással azonos természetű jelenségek között fejeztünk ki összefüggéseket, amelyeket ezért *homogén* viszonyoknak tekintünk. A kis tábla látszólag tökéletes. Azonban ha a Laci kezdetű sort töröljük, akkor a másik kettővel is tennünk kell valamit, mert ha Laci kilép, akkor nem lehet Feri főnöke. Ezért az adatmodellben rögzíteni kell, hogy létezik a FŐNÖK - SZEMÉLY kapcsolat. Az adatmodell-diagramokban a visszamatató viszonyokat az alábbi módon tükrözzük:



4.3 ábra: A visszamutató kapcsolat ábrázolása

Jellemzők: A visszamutató viszony **nemét** tekintve ( $\leftarrow$  Kapcsolatjellemzők) homogén birtoklási kapcsolat. A dolgozónak van főnöke.

A **fokot** illetően a fenti példa hierarchikus, azaz 1:N fokú viszonyt mutat, mivel egy (1) főnöknek lehet több (N) beosztottja is, de ez megfordítva nem igaz. Léteznek 1:1 fokú viszonyok is, mint amilyen például a házastársak mint személyek közötti kapcsolat. Az M:N fokú homogén összefüggéseket nem kapcsolattal modellezzük. A rájuk vonatkozó tudnivalókat külön pontban ismertetjük ( $\rightarrow$  Családfa).

A **függésérővel** vigyázni kell a rekurzív kapcsolatoknál. 1:N fok esetében az mindig kétoldalról opcionális. Ha nem az lenne, végtelen ciklust kapnánk, mert minden főnöknek lenne további főnöke és beosztottnak további beosztottja. 1:1 foknál ritka kivételes esetben elképzelhető a kötelező kapcsolat is. Például ha csak házasságokat tartunk nyilván.

**+** Az egyedtípus kapcsolatban áll saját magával, ha van olyan leíró tulajdonságtípusa, amely az azonosító szűkítő szerepneve.

Magyarázat: Az inhomogén kapcsolat mindig két egyed közös tulajdonságán alapul ( $\leftarrow$  Kapcsolótulajdonság). Itt erről nem lehet szó, hiszen csak egy egyedünk van és abban nem lehet két azonos nevű tulajdonság. Azonban nem a név, hanem a tartalom rokonsága a fontos. Jelen esetben a kapcsolatot egyazon egyednek két olyan tulajdonsága hordozza, amelyek közül az egyik (Főnöknév) a másinak (Személynév) a **szerepneve**. Még hozzá olyan módon az, hogy nem minden személy lehet főnök (szűkítés).

**Figyelmeztetés!** A mai adatkezelők közül kevés tudja kezelni a visszamutatást. Ezért a korlátokat procedurális módon kell betartatni.

SZEMÉLY	
Személynév	Főnöknév
Jóska	Laci
Feri	Feri
Laci	Jóska

4.4 ábra: Inkonzisztens visszamutató

Probléma: Az értékazonosságok bizonyos eseteit ki kell zárni. Például senki sem lehet a saját főnöke (Feri) és az sem szerencsés, ha valakik közvetlenül vagy közvetve egymásnak a főnökei (Laci és Jóska).

## 4.6 Családfa

+

Családfa [family tree] szerkezetéről beszélünk akkor, ha egy egyedtípus előfordulásai ugyanannak az egyedtípusnak más előfordulásaival M:N fokú összefüggésben állnak.

TÉTEL		CSALÁDFA		
Tételkód	Tételnév	Tételkód-1	Tételkód-2	Darab
T1	A	T1	T2	2
T2	B	T1	T3	2
T3	C	T2	T4	12
T4	D	T3	T4	8

4.5 ábra: Tipikus családfa adatbázisrészlet

Magyarázat: A példa arról szól, hogy az autógyárban a kocsit, annak szerelvényeit, azok alkatrészeit, legvégül pedig az anyagokat (részben) azonos ismeretekkel írják le és ezért egy közös egyedben (TÉTEL) tükrözik. Ki kell tudni mutatni, hogy egy tétel milyen másikkából (M) épül fel illetve milyen másikkba (N) épül be. Vagyis a tételek között M:N fokú viszony áll fenn. Mivel pedig a viszony elemei egyneműek, a tételek **homogén hálót** alkotnak.

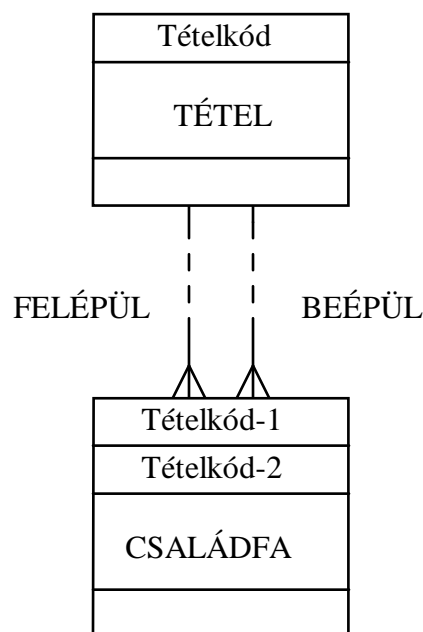
Probléma: Az egyedek tulajdonságai nem vehetnek fel többszörös értékeket, mert az gondokat okoz (← Ismétlődő csoport). Ezért a TÉTEL egyedben nem vehető fel N darab a beépülőre mutató és M darab a befogadóra utaló tulajdonság.

Megoldás: Az M:N fokú kapcsolatokat mindig egy további egyed segítségével oldottuk fel (← Hálós viszony). Ezt tesszük most is. A CSALÁDFA egyed tükrözi a hálót. Nagyon speciális megoldásról van szó, amely rengeteg sajátossággal rendelkezik.

Jellemzők: A családfa kapcsolat **nemét** tekintve (← Kapcsolatjellemzők) homogén birtoklási viszony. Az egyik tételnek van másik beépülő/befogadó tétele.

A **fok** mint említettük mindig M:N (háló), mert egy tétel többől (N) épül fel és többbe (M) épül be. (Az iparban a tartalmazási oldalt darabjegyzéknek [parts list], a befogadást beépülési jegyzéknek [goes-into list] nevezik.)

A **függésérő** mindig kétoldalúan opcionális, mert a végtermék már nem épül be másik tételbe, az anyag pedig már nem épül fel másik tételből.



4.6 ábra: A családfa-viszony modelldiagramja

Struktúra: A családfa szerkezet eléggé összetett. Mindig feltételez egy alapegyedet, ami esetünkben a TÉTEL. Ennek alárendeltje a CSALÁDFA. Ennek azonosítója az alapegyed kulcsa két szűkítő szerepnevének a konkatenációja. E tényezők kulcsrészként-kapcsoló szerepűek (← *Kombinált szerepek*) és ekként két felülről opcionális, alulról kötelező 1:N fokú kapcsolatnak adnak alapot. Lásd a fenti ábrát.

Kiegészítés: A családfa előfordulásai ún. *hierarchikus hálót* alkotnak. Ez nem tévedés. Az M:N fok miatt a viszony hálós. Ugyanakkor léteznek fölé/alárendeltségi viszonyok is. Például a kocsi a motor, az ajtó stb. fölérendeltje. Ezért a viszony egyben hierarchikus is.

SZEMÉLY		...	HÁZASSÁG		
Személykód	Személynév		Férjkód	Feleségkód	H. dátuma
S1	Jóska		S1	S2	D1
S2	Panni		S3	S4	D2
S3	Feri				
S4	Kati				

4.6 ábra: Házastársi viszony

Magyarázat: Az előző pontban már volt szó az 1:1 fokú - vagyis *homogén lineáris* - viszonyról. Mint minden esetben, az ilyen viszonyt csak akkor lehet kapcsolattal tükrözni, ha magára az összefüggésre vagyunk kíváncsiak (ki kinek a férje illetve felesége). Ha a viszonyt ismeretekkel is le akarjuk írni (házasság dátuma), akkor csak külön egyed a megoldás (különben két helyen tárolnánk, esetleg inkonzisztensen, a vonatkozó adatot). A családfa struktúrájának sajátos esete a *házastársi* [marriage] viszony, amiben a két kapcsolat mindegyike 1:1 fokú. Az olvasó gondolja

meg, hogy ezzel a strukturális korláttal elkerülhető az az anomália, hogy Panninak két férje legyen, vagy Ferinek ugyanaz legyen a felesége, mint Jósának, netán Ferinek Feri legyen a partnere.

## 4.7 Üres érték

Probléma: Az adatkezelő rendszerek nem tesznek különbséget az üres érték egymástól nagyon eltérő jelentései között. A lényegét a *menyiségre* redukálják („not null”), holott az üresek mögött eltérő *minőségek* húzódnak meg.

**4.4 példa** „Hányszor szült Panni? - Kétszer.”  
„Hányszor szült Zsóka? - Egyszer sem.”  
„Hányszor szült Kati? - Nem tudom.”  
„Hányszor szült Feri? - Hahaha...”

Magyarázat: Az üres érték - amit tévesen hívnak nullának, mert karakteres, dátumos stb. típusú adat is lehet kitöltetlen - háromféle eltérő jelentést hordozhat.

Az üres lehet *valós* ismeret („egyszer sem”) éppen úgy, mint bármelyik érdemi kitöltött adat („kétszer”). Lehet *inszignifikáns* információ is. A jelentéktelennek önmagában véve is három eltérő tartalma van. Egyik a „nem tudom”. A másik a „nem érdekel”. Ha valaki megadja az adatot, úgyis jó, ha nem, úgyis jó. A harmadik a „lényegtelen”. Az ötven forint alatti tartozások nem mindegyike nulla, de mindet semmisnek tekintjük.

Szemponunktól a harmadik változat érdekes. Az üres lehet *értelmetlen*. Feri mint férfi nem szülhet. A szögletes alkatrésznek nem lehet megadni az átmérőjét. Annak értéke nem nulla, hanem értelmetlen. Adott esetben eléggé nehéz a valós és az értelmezhetetlen tartalom elkülönítése. Egy két éves kislány még nem szülhet, esetében a dolog értelmetlen. Viszont igen nehéz korlátként megadni azt a korhatárt, amikor az egyikféle üres átvált a másikba és a kislány szülőképes nagylánnyá nő fel.

Probléma: A magyar ismeretközlésben olykor megtévesztő az üresek elkülönítése.

Magyarázat: Az angolban az „n.a.” [not applicable] betűk azt mutatják, hogy az ismeret *nem alkalmazható*, mert értelmetlen. A magyarban az „n.a.” sokszor azt jelenti, hogy *nincs adat*, vagyis az inszignifikanciára utal. A két dolog egymásnak ellentmond. Bár a statisztikákban hallatlanul fontos a valós, az inszignifikáns és az értelmetlen üres megkülönböztetése, itt most nem ez a szempont érdekel bennünket. Az értelmetlen üres értéken alapul egy igen gyakran alkalmazott modellezési konstrukció.

## 4.8 Egyedaltípus

+

Az egyednek a nem-értelmezhető tulajdonságok szerinti megbontását specializációnak, az eredmény-egyedeket az eredeti egyed altípusainak nevezzük.

### RENDELÉS

Rendelészám	Rendelésirány	Devizanem	Devizasorzó
1	hazai		
2	export	dollár	X
3	export	márka	Y
4	hazai		

4.7 ábra: Egy megbontható egyedtípus

Magyarázat: Ha a Rendelésirány hazai értékű, akkor a Devizanem és a Devizasorzó nem vehet fel értéket (legalábbis régebben ez volt a gyakorlati helyzet), vagyis értelmetlen értékű (← Üres érték). Ilyen esetekben megfontolható az egyedtípus megbontása.

### RENDELÉS

Rendelészám	Rendelésirány
1	hazai
2	export
3	export
4	hazai

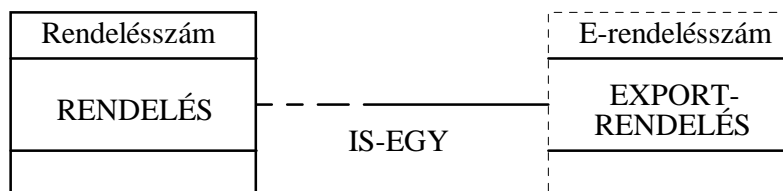
### EXPORT

E-Rendelészám	Devizanem	Devizasorzó
2	dollár	X
3	márka	Y

4.8 ábra: Egyedtípus megbontása

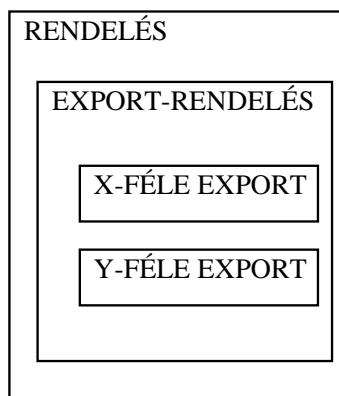
Magyarázat: A megbontást *specializációnak* nevezzük, mert a speciális ismereteket kiemeltük az eredeti egyedből. Az így keletkező EXPORT egyed a RENDELÉS *altípusa*. Itt meg kell jelezni, hogy nem valódi altípusról van szó. Csak olyan esetekben beszélünk altípusokról, ha alosztályozásra kerül sor, amely több változatot eredményez. Valójában az EXPORT a RENDELÉS-nek most csak *kiegészítő egyede* (← Kölcsönös viszony), ami azért nem lényeges eltérés a számunkra, mert a kiegészítés is az üres értékeken alapul.

Jellemzők: Az egyedaltípus ugyanazt a jelenséget tükrözi, mint az eredeti egyedtípus. Ezért ugyanaz kellene, hogy legyen a kulcsa. Viszont a modellezésben két egyednek nem lehet azonos a kulcsa (lásd a *relatív szerep* alatt leírt korlátokat). Ez a kettősség feloldható a szerepnév konstrukciója által. Az E-Rendelészám a Rendelészám szűkítő szerepneve (nem minden Rendelészám-érték utal export rendelésre). Ebben a kivételes esetben a két egyed közötti osztályozási jellegű kapcsolatot (← Kapcsolatjellemzők) a két kulcs teremti meg. A kapcsolat 1:1 fokú, felülről opcionális és alulról kötelező erejű.



4.9 ábra: Az egyedtípus és -altípus ábrázolása diagramon





4.10 ábra: Az egyedtípus és -altípus másíféle ábrázolása diagramon

Magyarázat: A szerző az első konvenciót részesíti előnyben. Ennek két oka van. Először a mai kezelők döntő többsége nem képes az altípusok explicit - a jellegüknek megfelelő - kezelésére, tehát az altípus ugyanúgy „viselkedik” a számukra mint a főtípus. Másodszor az altípusok számos csak rájuk vonatkozó ki- és befelé mutató kapcsolattal rendelkeznek. Ezek pedig aligha tükrözhetők kuszaság nélkül a második ábrázolási megoldásban. (Nem vitás, hogy képernyőn, kinyitható ablakok esetében a második megoldásnak is vannak előnyei. Azonban a szerző a végső dokumentációt mindig papíron képzei el.)

Probléma: A mai modellezési elméletekben csak egymást kizáró altípusok adhatók meg. Az iskolai személyek közül a tanár és a diák ki kell, hogy zárja egymást. Ha netán belép olyan személy is a képbe, aki diákként tanárfunkciót is ellát, akkor a modellt teljesen át kell alakítani és az altípus-hierarchiába fel kell venni a tanárként diák köztes altípust is. Ezekre a kérdésekre később még visszatérünk (→ *Specializáció és generalizáció*).

**FONTOS!** A szerző korábban nagy figyelmet szentelt az altípusoknak. A fentiekben összegzett korábbi nézeteit már réges-régen megtagadta, mert az altípus nem fogalmi szintű konstrukció, hanem a kezelők korlátai miatt használt logikai szintű megoldás. Tekintettel arra, hogy a mai kezelők az itt felvázolt megoldásokat kényszerítik ki, azt javasolja, hogy az olvasó - egyelőre - ragaszkodjon azokhoz.

## 4.9 Korrekt és inkorrekt kapcsolatok

Háttér: Az eddigi fejezetekben és pontokban nem véletlenül szenteltünk annyi figyelmet a kapcsolatoknak és az azokon alapuló szerkezeti megoldásoknak. Az adatbázis pontosan abban különbözik az ad-hoc módon meghatározott tartalmú állományok együttesétől, hogy benne az egyedek közötti kapcsolatok egzaktan, tulajdonságokkal támogatott módon szervezettek (← *Kapcsolótulajdonság*).

- +
- Két egyedtípus kapcsolattípust létesít akkor, ha az egyik azonosítója vagy annak szerepneve a másik egyedtípus leíró vagy kulcsrész szerepű tulajdonsága. Kapcsolattípus létezik az egyed fő- és altípus között is úgy, hogy az altípus kulcsa a főtípus azonosítójának a szerepneve. Egy egyedtípus önmagával létesít kapcsolatot akkor, ha tartalmazza kulcsa szerepnevét.**

Jól definiált kapcsolatok (az olvasó gondolja át, hogy miért azok):

- 11. eset: TULAJDONOS (Tulajkód), KOCSI (Rendszám, *Tulajkód*)
- 12. eset: CIKK (Cikkszám), RENDELÉSTÉTEL (Rendelésszám+*Cikkszám*)  
A fenti kapcsolatok normál leírón illetve kulcsrészen alapulnak.
- 21. eset: TELEPÜLÉS (Településkód), SZEMÉLY (Személy-az, *Lakik településkód*)
- 22. eset: TÉTEL (Tételkód), CSALÁDFA (*Tételkód-1+Tételkód-2*)  
A fenti kapcsolatok szerepnév leírón illetve kulcsrészen alapulnak.
- 31. eset: RENDELÉS (Rendelés az), EXPORT-RENDELÉS (*Export rendelés az*)  
Kapcsolat szerepnév azonosítón.
- 41. eset: SZEMÉLY (Személy azonosító, *Főnök azonosító*)  
Visszamatató kapcsolat az azonosító szerepnevén.

Rosszul definiált kapcsolatok (az olvasó gondolja át, hogy miért azok):

- 51. eset: SZEMÉLY (Személy az, *Költséghely az*), GÉP (Gép az, *Költséghely az*)
- 52. eset: RENDELÉSTÉTEL (*Rendelés az+Cikkszám*),  
SZÁMLA (Számla az, *Rendelés az*)
- 53. eset: RENDELÉSTÉTEL (*Rendelés az+Cikkszám*),  
DISZPOZÍCIÓTÉTEL (*Diszpozíció az+Cikkszám*)  
Leíró és kulcsrész párosokon nem létesíthető kapcsolat.
- 61. eset: RENDELÉSTÉTEL (*Rendelés az+Cikkszám*),  
SZÁMLA (Számla az, *Export rendelés az*)  
Leíró és kulcsrész párosok szerepnevein sem létesíthető kapcsolat.
- 71. eset: FŐNÖK (*Főnök azonosító*, Személy azonosító)  
Itt nem a leíró tulajdonság a kulcs szerepneve, hanem éppen fordítva.
- 81. eset: SZEMÉLY (Személy az), FELELŐS (Felelős az)  
Nem történt meg az explicit specializáció. Pl. a felelős lehet szervezet is.

## 4.10 Ellenőrző kérdések

4/1 Helyes (H) gyakorlatot jelentenek-e vagy sem (N) az alábbi megoldások:

- A RENDELÉSTÉTEL-ben a rendelésszámnak T-Rendelésszám a neve.
- A RENDELÉSTÉTEL-ben a mennyiséget az R-Mennyiség, a SZÁLLÍTÁSTÉTEL-ben az S-Mennyiség névvel illetjük.
- A RENDELÉS és a SZEMÉLY egyedben egyaránt van Dátum nevű adat.

4/2 Melyik igaz (I) és melyik hamis (H) megállapítás:

- A Rendelésdátum és a Szállításdátum egymás szerepnevei.
- Ha létezik a Dátum tartomány, akkor az előző két adat annak a szerepneve.

- A Születés kelte lehet a Dátum tartomány szerepneve.
  - A Hossz nem biztos, hogy a Méret tartomány szerepneve.
- 4/3 Melyik igaz (I) és melyik hamis (H) kitétel:
- A Dátum validáló rutinját a séma Dátum tartományához kell kötni.
  - A Dátum validáló rutinját az egyedi programokban kell megadni.
  - A Dátum validáló eljárását rutinként megírjuk és a programokba visszük.
- 4/4 Az alábbiak közül melyik jó (J) és melyik rossz (R) megoldás. Van szállásköltség, napidíj és dologi költség.
- KIKÜLDETÉS (**Személy**, Költség-1, Költség-2, Költség-3)
  - KIKÜLDETÉS (**Személy**, Szállásköltség, Napidíj, Dologi)
  - KIKÜLDETÉS (**Személy+Költség típus**, Költség)
- 4/5 Hogyan tükrözné az osztályfőnök és a diákok viszonyát? Az alábbiak közül melyik megoldás jó (J) és melyik rossz (R)? Vigyázzon!
- SZEMÉLY (**Személy az**, Tanár/Diák jel, *Ofő*) az Ofő a Személy szerepneve
  - TANÁR (**Tanárazonosító**, ...)
  - DIÁK (**Diákazonosító**, ...)
  - OSZTÁLY (**Tanárazonosító+Diákazonosító**, ...)
  - SZEMÉLY (**Személy az**, Tanár/Diák jel)
  - OSZTÁLY (**Tanárazonosító+Diákazonosító**, Ofőjel)
- A két azonosítórész a Személy az szerepneve.
- 4/6 Az alábbi két megoldás közül melyik tükrözi helyesen a fix szervezetek (S) és az egymásba fonódó projektek (P) ismereteit. Mindkettő részét egységnek hívjuk.
- EGYSÉG (**Egységkód**, ..., *Főlérendelt egységkód*)
  - EGYSÉG (**Egységkód**, ...)
- KAPCSOLAT (**Egységkód-1+Egységkód-2**, ...)
- 4/7 Mi a teendő a következő tervvel? A helyes válasz számát adja meg. Két jó válasz is van. Az Sz-adatok csak a szerzőt, az Í-adatok csak az írókat jellemzik.
- ZENESZERZŐ (Szerzőkód, Szerzőnév, Szerzőcím, Sz-adatok)  
SZÖVEGÍRÓ (Írókód, Írónév, Írócím, Í-adatok)
- A terv így kifejező, ezért jó.
  - A két egyed első három adata generalizálható a SZEMÉLY egyedbe.
  - A két egyed mindig teljesen generalizálható.
  - A két egyed generalizálható a SZEMÉLY egyedbe, ha az Sz- és Í-adatok száma nem nagyobb háromnál.
- 4/8 Ön tanár és nyilvántartást vezet a diákjairól illetve a barátairól. A diákok között is vannak barátai illetve megfordítva. Adja meg a helyes válasz(ok) számát:
- Készítetek egy DIÁK és egy BARÁT állományt.
  - Egyetlen SZEMÉLY egyedet képzelek el n.a. értékekkel.
  - Az előbbi teszem, ha a diákspecifikus jellemzők száma nem jelentős.
  - Készítetek egy SZEMÉLY egyed típust, meg egy DIÁK és BARÁT altípust, ha a specifikus jellemzők száma jelentős.
  - Három állományom lesz: DIÁK, BARÁT és DIÁKKÉNT-BARÁT.

## 5. MODELL, TERV, SZÓTÁR

### 5.1 Az adatbázis szemléleti síkjai

A fejlesztő által megtervezett ismeretegyüttes szolgáltatásait a végső felhasználó élvezi, ezért azt felhasználói vagy *alkalmazói adatbázisnak* hívjuk. Ma már az adatbázistervet magát is számítógépen állítjuk össze és tároljuk. Ha ezt nem tennénk, akkor is szervezett ismeretegyüttesként kellene felfogni a tervben foglaltakat. Ezért a fejlesztő számára fontos tervezési ismeretek halmaza is egy adatbázist alkot. Ezt nevezzük *fejlesztői adatbázisnak* (5.2 *Metaadatbázis*).

Az alkalmazói adatokat az adatmodell mintájára szervezzük meg. A fejlesztői ismeretek elrendezése sem tetszőleges. A metaadatmodell meghatározott és egymással adott módon összefüggő kategóriák szerint épül fel (5.3 *Metastruktúra*). A fejlesztő által elképzelt egyed-, tulajdonság- és kapcsolattípusokra és az azok korlátaira vonatkozó metaadatokat az információs erőforrás szótár tárolja a rendszer egyéb tényezőit leíró metaismeretekkel együtt (5.4 *Adatszótár*). Vagyis az adatszótár több, mint a metaadatmodell: az utóbbi az előbbinek csak az adatokkal kapcsolatos részét öleli fel.

*Ebben a fejezetben az adatbázisok két szemléleti síkjának az összefüggéseit, ezután pedig a fejlesztői adatbázis kettős lényegét fogjuk ismertetni.*

Fejezetünk legfontosabb üzenete, hogy az adatmodell nem egyenlő az adatbázistervvel (5.5 *Adatmodell és adatbázisterv*). Bár a fogalmi szintű modell ad alapot a logikai szintű tervnek (← *Absztrakciós szintek*), a kétféle terv két szempontból sem azonos egymással. Egyrészt tartalmi köreik csak átfedik egymást, mert a tervben vannak a modellre nem tartozó tényezők is és megfordítva, a modellnek nem minden eleme tükröződik a tervben. Érdekessége miatt egy ilyen konstrukciót külön kiemelten is ismertetünk (5.6 *Álgyedek*). Másrészt az adatbázisterv a kezelési, hatékonysági, biztonsági és egyéb, nem közvetlenül a valóság tükrözésére irányuló - másodlagos - szempontoknak megfelelően eltorzítottan, kompromisszumosan rendezheti el az ismereteket. Viszont az adatmodellnek mindig érthető, valóságghű, egyértelmű, teljes és minimális képet kell nyújtania, amint azt majd részletesebben is kifejtjük (5.7 *Optimumkritériumok*).

A fentiek következtében az adatszótár felépítése jóval összetettebb, mint azt feltételezik és mint az a mai tervezési segédletek alkalmazásából kitűnik. A szótárnak egyszerre és együtt kell(ene) tartalmaznia a kompromisszum-mentes fogalmi adatmodell tényezőit, a józan megfontolások után annak átalakításával születő logikai adatbázisterv elemeit, sőt, illenék rögzítenie a modellterv leképezés miéjtjeit és mikéntjeit is. Az, hogy miért nem így épülnek fel a mai szótárrendszerek illetve miért nem így alkalmazzák őket, már nem tartozik e könyv tárgykörébe.

### 5.2 Metaadatbázis

+

Az ismeretekre vonatkozó ismereteket metaadatoknak hívjuk.

- 5.1 példa** „Rózsa kocsija Lada típusú.”  
 „A Lada típusú kocsik ötszemélyesek.”  
 „A Kocsitípus tulajdonság karakteres adat.”  
 „A Kocsitípus tulajdonság a KOCSI egyedet jellemzi.”

Magyarázat: Az első két mondat „valós” jelenségekről közöl ismereteket. Ezt azt jelenti, hogy a felhasználók számára lényeges dolgokra vonatkozik. Ahhoz, hogy az adatokat majd számítógépen tárolhassuk és kezelhessük, meg kell határozni azok szerkezetét és sajátosságait. A harmadik mondat az ismeretről (Kocsitípus) közöl ismeretet (karakteres adat). Ez teszi a negyedik is, amely egy strukturális részletet rögzít.

**+** **A metaadatbázis meta-egyed-, tulajdonság- és kapcsolatelőfordulások szervezett együttese.**

Magyarázat: Az ismereteket négy plusz kettő strukturális tényező (← *Adatdimenziók*) szerint rendezzük el. A metaismeret is adat, ezért annak esetében is értelmeznünk kell a szóbanforgó dimenziókat.

A fenti harmadik mondatban egy tulajdonságérték (karakteres) jelenik meg. Világos, hogy ez az érték az Adattípus általános jellemző értékkeszletének az egyik eleme (a numerikus, dátum stb. értékek mellett). Az Adattípus tehát *meta-tulajdonságtípus*, míg a „karakteres” annak egyik tulajdonságértéke, másképpen előfordulása.

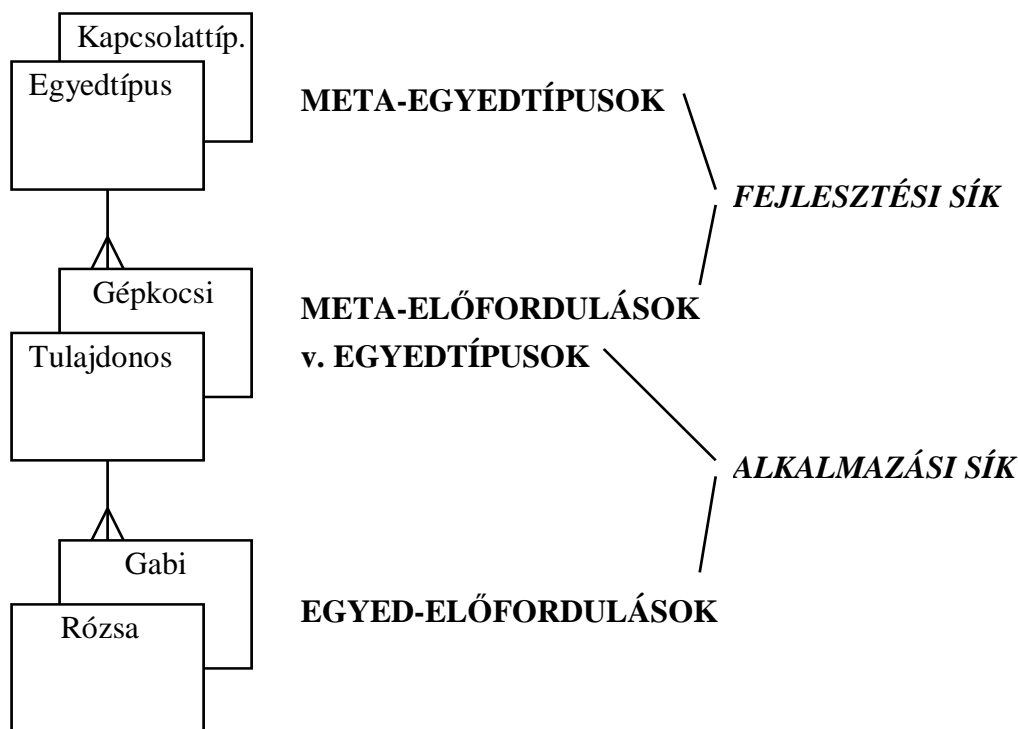
Az adattípus nem a kapcsolat és nem az egyed ismérve. Ez a jellemző a tulajdonságok sajátossága: a tulajdonságtípust írjuk le az adat típusával. Az ismeretekkel leírni kívánt dolgokat pedig egyedeknek nevezzük. Ezért a tulajdonságtípus - *meta-egyed*típus. Ebből következően a Kocsitípus annak egyik meta-egyedelőfordulása. A konkrét egyedtípus (pl. KOCSI) szintén ismerettel leírt jelenség, ami az egyedtípus metaegyed előfordulása.

Az adatmodellt alkotó tényezők nem függetlenek egymástól, hanem azokból struktúrák épülnek fel. Tehát nemcsak a KOCSI és a TULAJDONOS között létezik valós viszony, hanem kapcsolatot kell kialakítani a meta-egyedtípusok között. Példánk utolsó mondata voltaképpen egy konkrét metaviszonyt rögzít (KOCSI - Kocsitípus). Ez is általánosítható: az egyedek és a tulajdonságok között általában is viszonyok vannak. Ezért kell beszélni a *meta-kapcsolattípusokról* is. Ilyen az EGYEDTÍPUS - TULAJDONSÁGTÍPUS meta-kapcsolat, amelynek egyik előfordulása a KOCSI - Kocsitípus konkrét metaviszony.

**+** **A metaadatmodell a meta-egyed-, -tulajdonság- és -kapcsolattípusok szervezett együttese.**

Magyarázat: A fejlesztési adatbázis is adatbázis, tehát szervezett ismeretegyüttes. Nem tetszőleges a felépítése. Struktúrája meg kell, hogy feleljen a fejlesztők által általánosan használt kategóriáknak, azaz metatípusoknak, amelyek adott elvek szerint elrendezettek. Ez az elrendezés a metaadatmodell.

Az alkalmazási és a fejlesztési adatbázis egymással összhangban kell, hogy álljon. Ez a harmónia azon alapul, hogy az alkalmazási adatbázisban lévő tényezők mint típusok a fejlesztési adatbázis előfordulásaiént jelennek meg:

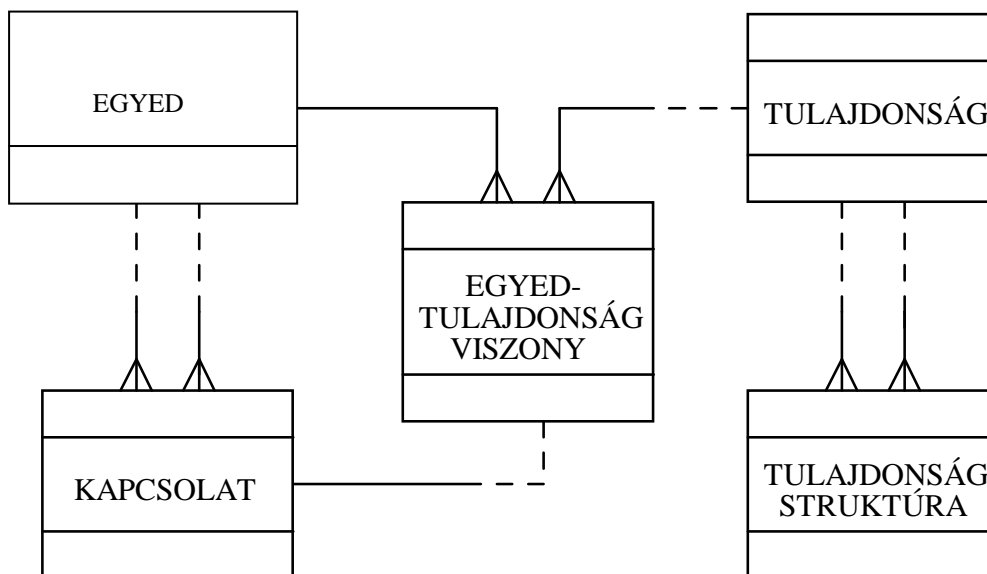


5.1 ábra: Az adatbázisok két szemléleti síkja

Magyarázat: A fenti ábra nem adatmodellt rögzít, hanem az alkalmazási és a fejlesztési adatbázis tényezőinek a sematikus összefüggését mutatja. Az alkalmazási síkon Rózsa egy egyedelőfordulás. Gabival és társaival együtt őket a Tulajdonos egyedtypusba osztályoztuk. Egy másik egyedtypus a Gépkocsi. A fejlesztési síkon a Tulajdonos és a Gépkocsi meta-egyedelőfordulás, amiket az Egyedtypus meta-egyedtypusba soroltunk. Másik ilyen tényező a meta-kapcsolattípus. A metamodellt alkotó tényezőket a következő pontban ismertetjük (→ *Metastruktúra*). Megjegyzés: A metamodell több, mint az alkalmazási adatbázis modelljének a modellje. A meta-adatbázis az ismeretekről szóló ismeretek tárháza. Nem csupán az alkalmazási adatbázis szerkezetét rögzíti, hanem az adatbázisra vonatkozó minden tudnivalót felölel. Tárolja a fejlesztéssel és a használattal kapcsolatos adatokat is. Azonban bennünket ebben a könyvben csak az adatstruktúrákra vonatkozó tényezők érdekelnek.

### 5.3 Metastruktúra

Háttér: Az alkalmazási adatbázis adatmodelljét a *metaadatbázis* írja le. Ez maga sem tetszőleges felépítésű, hanem a metaadatmodellnek felel meg. A metamodell viszont egy olyan struktúrát követ, amely a fejlesztési koncepció által meghatározott. Másképpen épül fel a relációs, mint az objektumos metamodell. Ebben a műben mi a saját metamodellünk szerint gondolkodunk. Lásd az alábbi ábrát:



5.2 ábra: A metamodel diagramja

Magyarázat: Struktúránk sajátossága, hogy - szemben minden más koncepcióval - mi megkülönböztetünk abszolút és relatív tulajdonságot. Azoknál, akik „rekordképekben” gondolkodnak, a tulajdonságtípus nem önálló lényeg, hanem az egyednek hierarchikusan alárendelt tényező. Ez kétszeresen is baj. Egyrészt azért, mert nemcsak egy egyedtípushoz tartozik több (M) tulajdonság, hanem egy tulajdonság több (N) egyedhez is kapcsolódhat. Tehát a két tényező M:N-es, azaz hálós viszonyban áll egymással. Másrészt az is gond, hogy az egyéb koncepciókban a tulajdonság nem írható le önmagában is illetve arra nem lehet saját szerkezeteket építeni. A fenti struktúra elemeinek a lényege a következő:

**Egyed(típus):** Itt is ugyanazt jelenti, mint a legtöbb koncepcióban.

**Tulajdonság(típus):** Az egyedtípustól függetlenül is tükrözhető tényező. Léteznek olyan tulajdonságok is (pl. konstansok), amelyek elvileg sem kapcsolhatók egyedekhez. Ezeket más koncepciókban egyszerűen nem lehet leírni a metaadatbázisban, ami hiányosság.

**Egyed-tulajdonság-viszony:** A fentiek értelmében a végső modellben nem lehet olyan egyedtípus, amely ne kapcsolódna tulajdonságtípushoz, de ez megfordítva nem igaz. Ezért a tulajdonságnak a viszony felé való kapcsolata opcionális.

**Kapcsolat(típus):** Az egyedek kapcsolatokban állnak. Ez a viszony családja jellegű, mert egy egyednek lehetnek alá- és fölérendeltjei. Minden kapcsolat egyedtípusok között létezik, ezért a viszonyok alulról kötelezőek. Azonban nem minden egyednek van fölé- és alárendeltje, ezért a kötődések felülről opcionálisak. Sajnos, azt nem lehet kifejezni ilyen ábrákon, hogy a két kapcsolat közül valamelyik felülről kötelező, azaz nem létezhet a modellben más egyedhez nem kötött egyed. A kapcsolat és az egyed-tulajdonság viszony közti vonal azt fejezi ki, hogy egyes attribútumok *kapcsolótulajdonságként* szolgálnak. Minden kapcsolat ilyen viszonyon alapul, de nem minden attribútum hordoz viszonyt.

**Tulajdonságstruktúra:** A tulajdonságok családfákat alkotnak. Háromféle struktúrát alkalmazunk. A *csoportokban* az egyik tulajdonság (maga a csoport) kapcsolódik a saját tagjaihoz (pl. Dátum-Év, Dátum-Hónap, Dátum-Nap). A *szerepnév* is hasonló viszony (pl. Személy az - Főnök az). Struktúrában lehet kifejezni a *számításokat* is. A származtatott ismeret a származtató tényezőkhöz kapcsolódik (pl. Érték - Ár, Érték - Mennyiség).

**Figyelmeztetés!** Ma még nem létezik olyan a piacon is kapható tervezési segédlet, amely az itt bemutatott metamodell-szerkezetet támogatná. Mivel pedig ez tükrözi a helyes összefüggéseket, a modellezőknek ezt a mintát kellene követniük.

## 5.4 Adatszótár

+

**Az információs rendszer mindenfajta tényezőjének a metaismereteit őrző adattárat információs erőforrás szótárnak [information resource dictionary - IRD] nevezzük.**

Magyarázat: Az információs rendszer adatoknak, feldolgozásoknak, erőforrásoknak, felhasználóknak és szabványoknak a szervezett együttese. Mindezeket a tényezőket le lehet és le kell írni metaismeretekkel. Régebben a hasonló adattárak csak az adatokkal kapcsolatos metaismereteket őrizték és ezért egyszerűen **adatszótárnak** [data dictionary] hívták őket. Azért adtuk a pontnak is ezt a címet, mivel e könyvben bennünket csak ezek a vonzatok érdekelnek.

Kérdés: A metaadatbázis nem adatszótár?

Válasz: A tulajdonképpeni metaadatbázis a metamodellt és az annak tényezőire illetve azok viszonyaira vonatkozó korlátokat tartalmazza. Viszont az adatszótárban nemcsak a fogalmi, hanem a logikai és a fizikai szintű tényezők is leírásra kerülnek. Mi több, a szótárban még teljesen „laza” elképzelések, átmeneti tervek stb. is helyet kaphatnak. Ezért az adatszótár és a metaadatbázis az egész és a rész viszonyában áll. A szótárnak a végső fogalmi adatmodellel kapcsolatos tényezői alkotják a metaadatbázist.

Az adatszótárban tárolt metaismeretek a következő főbb csoportokba tartoznak:

**Metaazonosítók:** Az egyed-, tulajdonság- és kapcsolattípusok kulcsai, amelyek nemcsak hivatkozásra, hanem a tényezők közötti viszonyok megteremtésére is szolgálnak.

**Nevek:** Az azonosítókkal általában szigorúan kölcsönös viszonyban álló megnevezések. Például az Egyed típus-név, a Kapcsolattípus-név stb. A metaazonosító a kezelőrendszer, a megnevezés az ember számára a hivatkozás eszköze.

**Leírások:** A JÁRMŰ egyed típus-név nem igazán árulja el, hogy abba valaki beleérti a kocsikon kívül a kismotoros járműveket is. Ezért van szükség szöveges leírásokra. Egyes szótárrendszerekben különbséget tesznek az egy-két soros meghatározás és a terjedelmes „igazi” leírás között. Ez felesleges és zavaró, mert a két dolog ellentmondhat egymásnak. Leírásban adjuk meg



a definíciót, a származtatást, továbbá mindazon korlátot, amit nem lehet vagy nem akarunk metajellemzőkben adatszerűen rögzíteni.

**Metajellemzők:** Szemben az alkalmazási adatbázis ismereteivel, amiket tulajdonságok [attribute] hordoznak, egyes koncepciókban a metatulajdonságot jellemzőnek [property] nevezik. A (meta)jellemző természetét tekintve nagyon sokféle lehet. Az *osztályozó* tétel feladata a természet szerinti besorolás. Ilyenben adjuk meg például a kapcsolat fokát és erejét (← *Kapcsolatjellemzők*).

Kevés szótárkoncepcióban alkalmaznak *minősítő* jellemzőket. Például sehol nem lehet találkozni az általunk nagyon fontosnak tartott relatív és abszolút szerep minősítésekkel, amelyek a tulajdonságokat jellemzik. Egyáltalán nincs mód az egyedek minősítésére. Pl. a Családfa cím alatt ismertettünk egy olyan igen összetett struktúrát, amit ezzel az egyetlen besorolással lehetne minősíteni ahelyett, hogy mindig felsoroljuk a kapcsolati korlátokat.

A metaadatbázist fejlesztési adatbázisnak is neveztük. A fejlesztésnek alapvető lényege az elemzés, aminek során a modell tényezőit - időlegesen - jónak, rossznak stb. tartjuk. A mai szótárrendszerek egyáltalán nem alkalmaznak ilyen *értékelő* jellemzőket. Nem lehet rögzíteni például azt, hogy egy adott egyed csak második normálformában van.

Megjegyzés: Amint látjuk, a szótár több információt hordoz, mint a metaadatbázis. A minősítés, az értékelés, sőt az osztályozás egyes mozzanatait is csak a fejlesztés közben játszanak szerepet, de semmilyen módon nem épülnek be magába a metaadatbázisba.

## 5.5 Adatmodell és adatbázisterv

— *Az adatmodell nem adatbázisterv, hanem annak csak alapja.*

Magyarázat: Nagyon sokan élnek abban a félreértésben, hogy amikor ők adatmodellt készítenek, akkor ezzel összeállítják az adatbázis tervét is és megfordítva, amikor adott eszközzel adatbázistervet alkotnak, akkor adatmodellezést hajtanak végre. Mindez azt mutatja, hogy nincsenek tisztában a tervezési síkokkal (← *Absztrakciós szintek*). Mivel pedig a legtöbb tervezési segédlet is ezt a hamis benyomást kelti, nem kerülhetjük el, hogy a két lényeg viszonyát alaposabban körüljárjuk.

### 5.5.1 Hatókör

— *Az adatmodell - térben és időben - mindig egy. Az adatbázis több is lehet.*

Példa: Az X segédlettel valaki elkészíti a könyvelés információs rendszerének a tervét. Tőle teljesen függetlenül másvalaki ugyanezzel az eszközzel összeállíthatja a szerződések nyilvántartási rendszerét. Következmény: a szervezetben lesz két összehangolatlan, együtt nem is használt Partner adatállomány. A cégnek van egy könyvelési és egy szerződési úgymond adatbázisa, csak éppen nincsenek integrált ismeretei, mert nincs - adatmodellje.

Probléma: Az ismeretek egy része *funkciókhoz* kötött. A funkciókat pedig általában meghatározott *szervezeti egységek* hajtják végre. Az informatikai fejlesztések többnyire a funkciókat, kevésbé szerencsés esetben az egységeket szolgálják ki. (Ez azért nagyobb baj, mert hasonló funkciókat több egység is elláthat.) A fejlesztés közben nem figyelnek arra, hogy az ismeretek másik része (vö. Partner) funkciótól/egységtől független lényeg. Ezért a partneradatokat tárolhatják ugyan több fizikai adatbázisban is, de csak azután, ha a közös fogalomról egységes képet (modellt) alakítottak ki.

Példa: A biztosítási rendszerben a Szerződésszámot Körzetkód+Sorszám összetételként határozzák meg.

Probléma: Az azonosítónak tértől és időtől független lényegnek kellene lennie. Mármint ha a szerződést az egyik körzet átadja a másiknak (ami gyakran előfordul), akkor a kulcs megváltozik és maga a jelenség előbb-utóbb követhetatlenné válik. Tehát nem az a baj, hogy nincs egyetlen központi adatbázis, hanem az, hogy a lokális adatbázisok (több) nem követnek közös modellképet (egy). Azért nem, mert az ismeretek *földrajzi helyhez* és/vagy *időtartamokhoz* kötöttek.

Példa: Egy rendőrségi információs rendszerben senki sem akarta megérteni, hogy magát a rendőrt is személyként - bár sajátosként - kell modellezni.

Probléma: A rendőr is bűnözővé válhat. Ha kettős, egymástól szétválasztott ismeretsort vezetünk róla, akkor redundáns, inkonzisztens, nehezen kezelhető adatbázishoz jutunk. Azért, mert nem látjuk meg a többen (adatbázis sík) az egyet (adatmodell sík). Vagyis nem alkalmazzuk korrekten a *szemléletek* lehetőségét (← *Absztrakciós vetületek*), hanem eleve nézetorientált terveket készítünk.

Összegzés: Az adatmodell a *közös és tartós* tényezőket tükrözi. A két jelző összefügg: ami ma még nem közös, az holnap már az lehet. A nem-közös és nem-tartós vonások alapján létrehozhatunk funkcionális, szervezeti, területi, időleges adatbázisokat. Ez csak akkor nem jár totális dezintegrációval, ha van egyetlen integráló erejű adatmodellünk.

### 5.5.2 Szint és nézet

— ***Az adatmodellezés a globális nézetű fogalmi szintre irányul. Viszont az adatbázis logikai/fizikai szintű, adott esetben parciális nézetű lényeg.***

Példa: Ha valamit eladunk egy vevőnek, akkor általában számlának nevezzük a dolog közvetlen következményét. „Modellünkben” vevő és számla egyedtípusokat alkalmazunk.

Probléma: Ez a megközelítés se nem fogalmi szintű, se nem globális nézetű. Nem *globális* nézetű, mert a raktáros, a piackutató stb. is érdekelt az eladott cikk/szolgáltatás nevében, darabszámban stb., de egyikük sem igazán kíváncsi a könyvelési/elszámolási vonzatokra. A „számla” szó csak a könyvelő nézetének felel meg. Kétségtelen tény, hogy az adás/vétel számlával is jár. A *fogalmi* szintű lényeg azonban nem ez, hanem az, hogy eladás történt. Vegyük észre, hogy mindez nem játék a szavakkal. A mérnök műszaki objektumnak nevezi ugyanazt a dolgot, amit a közigazda vagyontárgynak hív. A kétféle nézet miatt két adatbázist alakítanak ki, holott pl. egy épület mindig azonos önmagával.

Példa: Az egyik alkalmazásban a számlát a Számlaszám azonosítja, amit sorszámként kezelnek. A másikban azt a Vevőkód+Relatív sorszám párosaként határozzák meg. A Vevőkód karakteres, mnemonikus - emlékeztetően rövidítő - adat.

Probléma: Az adatnak van jelentése, tartalmi elrendezése és formátuma. Ugyanaz a lényeg (jelentés - fogalmi szint) számtalan eltérő módon rendezhető el és önthető alakba (logikai és fizikai szint). Az adatmodellben tökéletesen lényegtelen, hogy a számlaszámot karakteresen vagy numerikusan ábrázolják-e. Csak az a fontos, hogy a számlának kell, hogy legyen egy egyértelmű azonosító tulajdonsága. Viszont az adatbázistervben le kell írni, hogy a Számlaszám pl. hatjegyű numerikus adat.

Példa: A biztosítási adatbázisban az ügynök neve 30, a biztosítotté 40 karakteres.

Probléma: Az ügynök maga is biztosított. Nem tisztázták a fogalmak összefüggéseit, hanem azonnal logikai/fizikai szintű tervezésbe fogtak. Biztosan korszerű segédlettel...

Összegzés: Az adatmodellezés a fogalmak lényegének a megértésére irányul. A fogalmi szinten kell belátni, hogy az ügynök és a biztosított egyaránt - partner, tehát közös lényeg. Az adatbázistervezés jó esetben a már elrendezett fogalmaknak a tartalmi (logikai szint) elrendezésére és a formai részletezésére (fizikai szint) törekszik. Rosszabb esetben a fogalmi összefüggések tisztázása nélkül azonnal a „rekordképek” leírásával kezdődik.

### 5.5.3 Alapegység

— ***A modellt mindig természetes elemi tényezők alkotják. Viszont a tervben szerepelhetnek mesterséges és összetett alkotóelemek is.***

Példa: Ismét hivatkozunk a Számlaszám = Vevőkód+Relatív sorszámra mint kulcsra.

Probléma: Egyáltalán nem baj, ha az adatbázistervben ilyen összetételű azonosítót alkalmazunk. Baj az, ha a tervben a Számlaszám tétel összetétele rejtett (implicit) marad, mert sehol nem fejtjük ki (expliciten) az összefüggéseket. Nagyon sokszor előfordul, hogy egy adatbázistervből egyszerűen nem lehet kiolvasni a lényegi viszonyokat, mert a nevek alá rejtik a különböző tartalmakat. Az ember nem érti, hogy a számla hogyan kapcsolódik a vevőhöz, mire az „adatbázis-tervező” közli, hogy a számlaszámban benne van a vevőkód is. A probléma lényege az, hogy az ilyen tervező azonnal adatban, nem pedig jelentésben, fogalmakban gondolkodik és fogalmazza meg elképzeléseit.

Példa: A Személyi szám első karaktere adott értékek esetén a nemre, mások esetében a származási helyre és/vagy a születés évszázadára utal.

Probléma: A logikai/fizikai tervezési szinteken a fejlesztő az adatokon takarékoskodik (amivel persze bonyolultabbá teszi a procedúrákat, de ez most mellékes). Tehát **összetett** adatokat alkalmaz úgy is, hogy a Személyi szám több részre tagolható, meg úgy is, hogy egy rész egyszerre többféle jelentést is hordoz. A fogalmi szinten ez nem megengedett. Ott **elemi** fogalmakban kell

gondolkodni. Előbb látni kell a nem, a származási hely, a születési évszázad lényegét, csak azután lehet eltöprengeni a technikai megoldáson.

Példa: A személy fájlban a férfit F, a nőt N értékkel kódolják. A származási ország megjelölésére egy háromkarakteres egyezményes kódot alkalmaznak.

Probléma: Egyesek teletűzdelik a fogalmi modellt kód-megnevezés párosokat kifejező úgynevezett egyedtípusokkal. Ez alapvető hiba. Ha a nemet és a származási országot senki nem akarja érdemi ismeretekkel leírni, akkor azok nem tekinthetők egyedeknek. A nem kódja és az ország kódja pusztán csak *mesterséges* adatok. Mint ilyenek a logikai/fizikai adatbázistervezés szintjére tartoznak. Fogalmi szinten csak annyit kell tudnunk, hogy a „nem” fogalom *természetes* lényege szerint az emberek nők vagy férfiak. Ha akarjuk, az ismeretet kódoljuk. Ha kódoljuk, akkor azt megtehetjük ezerféle módon. Mindez már nem a fogalmi modellezés gondja, mert hatékonysági/ábrázolási megfontolásokról van szó.

Kivétel: A településkód is mesterséges adat. Azonban ennek nem egyetlen célja a méret csökkentése, ha egy ismeretekkel leírandó jelenség - a TELEPÜLÉS - *azonosítására* is szolgál. A nominális kulcsok (← *Azonosító*) a legtöbb esetben ilyen mesterséges tényezők.

Összegzés: Az adatbázistervező ezernyi ok miatt és módokon különböző mesterséges és összetett adatokat alkalmazhat. Sőt, ezt meg is kell tennie. Azonban egyrészt a célszerű megoldáshoz neki is tisztában kell lennie az elemi és természetes fogalmak lényegével. Másrészt az semmiképpen sem engedhető meg, hogy a fogalmi-logikai leképezés rejtve maradjon és valakiknek utólagosan és nagy kórával kelljen visszafejtenie azt, hogy az adatbázistervben az X valójában azt jelenti, hogy ... , valójában úgy összetett, hogy ... stb.

#### 5.5.4 Tartalmi kör

— ***A fogalmi modell és az azt megvalósító adatbázisok egymással metszet viszonyban állnak.***

Magyarázat: A fogalmi adatmodell és a logikai adatbázisterv egyaránt tényezőknek a szervezett együttese, de a két tényezőkör nem azonos. Az adatmodellben vannak olyan elemek is, amelyek nem szerepelnek az adatbázisban és megfordítva, az adatbázisban léteznek olyan tényezők is, amelyek nem kerülnek modellezésre. Az előző alponthoz már volt szó egy ilyesfajta jelenségről, a mesterséges kódokról. Ebben az alponthoz a további különbségekre hívjuk fel a figyelmet.

Fogalmi tényezők: A legtöbb tervezési segédlet módot ad arra, hogy meghatározzuk az egyedek közötti *kapcsolatokat*. A relációs adatbázisokban a kapcsolat nem önálló lényeg: például az adatkezelés során nem lehet rá névvel hivatkozni. A kapcsolat adattételekben (← *Kapcsoló-tulajdonság*) testesül meg úgy, hogy bizonyos korlátok ellenőrzését váltja ki. Hasonlóképpen fogalmi szinten lényeges strukturális tényező a *csoport* és a *szerepnév*, de logikai szinten egyik sem nyilvános (explicit), hanem rejtetten (impliciten) valósul meg.

Logikai tényezők: Az adatbázisokat gyakran kötegelten dolgozzák fel. A *bemenetekből* a validálás kedvéért képeznek kötegeket. A *kimeneteket* kiírás előtt így-úgy rendezendő kötegekben gyűjtik. Többszörös további műveletek esetén a kezelés közben leválogatott adatokat olykor

*átmenetekben* tárolják. A be-, át- és kimenetek maguk is adatállományok. Ezeket is meg kell tervezni, vagyis meg kell határozni a tábla- azaz rekordképeiket. Ezért az adatbázisterv egésze felöleli a felsorolt tényezők struktúráit is.

Azonban világos, hogy ezek a dolgok nem képezhetik az adatmodell részét. A bemenet nem tükröz(het) más fogalmi lényeget, mint a modellben lévő egyedek. Az abban tárolt ismeretek az érdemi egyedek tartalmához képest redundánsak (pl. a partner bemeneten a partner egyed adatai találhatóak). Ráadásul a bemenet - de főleg a kimenet - nem is a valós egyedeket tükrözi, mert több egyednek néhány tulajdonságát ölelheti fel. A „...menetek” nem a valós jelenségek tükörképei, hanem az adott feldolgozás jellegéből adódó technikai megoldások. Lehet „modellszerűen” tervezni azokat is, de nem a modell alkotórészei.

Technikai tényezők: Az adatbázisokban gyakran felvesznek olyan állományokat illetve mezőket, amelyek a *kezeléstechnikával* kapcsolatosak. Ezek rendkívül sokfélék lehetnek. Utalhatnak a kiírás módjára; arra, hogy megtörtént-e és mikor a kiírás; mikor és ki ellenőrizte utoljára a tartalmat; ki és hogyan rögzítette az adatot; ki tette a bizonylatot a levéltárba és hová; stb. stb. Ide sorolhatjuk a *biztonsági/hozzáférési* ismereteket is. Például azt, hogy ki és milyen módon jogosult bizonyos adatok kezelésére.

—

***Az adatmodell nem tartalmazhat technikai adatokat.***

Magyarázat: Természetesen megint csak igaz, hogy a felsorolt és egyéb tényezőknek is el kell készíteni a tervét. Még hozzá az adatmodellel összhangban, mert például miközben a felhasználó valós egyed is, rá hivatkozni kell tudni a jogosultság technikai táblájából is. A fenti szabály azt mondja ki, hogy az érdemi ismereteket (adatmodell) el kell választani a technikai/menedzselési/adminisztrációs adatoktól (technikai adatbázisrész).

Nem szabad például a Partner egyed tulajdonságaként felvenni az Utolsó kiírás adatot. Ezt a megkötést több érv támasztja alá. Ad 1) Nemcsak a partnerek, hanem a szerződések adatait is ki kell írni. Ezért ki tudja hány egyedben lesz Utolsó kiírás adat. Ad 2) Maga a kiírás egy technikai esemény, amit éppen ezért egy külön technikai táblában kell vezetni. Ad 3) Ezt azért is meg kell tenni, mert általában a technikai adatok többszörösek. Erre utal az „utolsó” jelző is.

Kiegészítés: Az adatmodell és az adatbázisterv tartalmi köre még további tényezőkben is eltér egymástól. Lásd a következő pontot.

## 5.6 Álegyedek

+

***Az álegyed olyan generalizált egyedtípus, amelyet nem egy-az-egyben valósítunk meg az adatbázisban.***

Példa: Egy alkalmazásban ügyekről vezetünk ismereteket. Az ügyeket olyan események alapozzák meg, amelyek személyeket, szervezeteket, tárgyakat és helyeket érintenek. Ki kell mutatni az ügyek és e tényezők összefüggéseit; a tényezők egymás közötti viszonyait (például mi köze van egy személynek egy tárgyhoz); végül a tényezők típusain belüli kapcsolatokat is (például két személy viszonyát).

Probléma: Az eseményt is ideértve van ötféle jelenségünk, amelyek mindegyikét az ügghöz kell kötnünk (5 darab kapcsolat). Közöttük  $5 \cdot 4/2 = 10$  darab további viszony áll fenn, mert az eseményhez hozzá kell kapcsolni a másik négyféle jelenséget stb. Végül az öt tényező 5 darab struktúrát alkot (személy-személy, tárgy-tárgy stb.). Amikor a modellt alkotó tényezők ilyen körkörös módon kapcsolódnak egymáshoz, fellép az a jelenség, amit *pókháló-effektusnak* nevezünk. Ha újabb tényező lép be, akkor újabb körök és átlók fognak ráépülni a hálóra, de nem is igazán ez a komoly probléma. Hanem az, hogy a további (pl. az intézkedés, határozat, tevékenység stb. felé) kimutató kapcsolatok maguk is mind N-szeresek lesznek. Az adatmodell nem áttekinthető, nem elemezhető, a ráépülő adatbázis pedig igen összetett szerkezetű és nehezen kezelhető lesz.

Megoldás: Vegyünk fel egyetlen OBJEKTUM egyedet, amelynek altípusa az esemény, a személy, a szervezet, a tárgy és a hely. Persze bármikor felvehetünk újabb altípusokat is ( $\leftarrow$  *Egyedaltípus*). Az objektum a felsorolt tényezőkből generalizált egyedaltípus, azok pedig az ő specializáltjai ( $\rightarrow$  *Generalizáció és specializáció*). Mit nyerünk a megoldással? Az öt ügy-személy, ügy-tárgy stb. helyett egyetlen ügy-objektum viszonyt kell alkalmazni. A tíz inhomogén viszony és az öt homogén struktúra helyét szintén egyetlen egy tényező, az objektum-struktúra veheti át. Eredeti tervünk 20 egyedet 2-re redukáltuk! Ezzel nem veszítettünk ismeretet, sőt, rugalmasabban bővíthető struktúrát alakítottunk ki. Már csak az a kérdés, hogy az OBJEKTUM miért „álegyed”?

Magyarázat: Az öt tényezőt nem azonos ismeretekkel kell leírunk, vagyis teljesen más azok tulajdonságsora. Ezért a logikai/fizikai szintű megvalósításban valóságosan öt táblát kell majd alkalmaznunk. Nem lesz külön objektum táblánk: azt nem valósítjuk meg az adatbázisban. Ugyanakkor a ráépülő OBJEKTUM-STRUKTÚRA valós tábla lesz. Oda fog kapcsolódni a SZEMÉLY, a TÁRGY stb. táblák mindegyike.

Az „álgeneralizálás” két dolgot szolgált. Egyrészt nélküle nem találtuk volna meg ezt a célszerű logikai adatbázistervet. Másrészt a modell nem lett volna elemezhető. Végül is az álegyed újabb példa arra, hogy a fogalmi adatmodell és a logikai adatbázisterv metszet viszonyban áll egymással. Az előbbinek nem minden tényezője szerepel az utóbbiban is.

## 5.7 Optimumkritériumok

— ***Az adatmodell nem pusztán egy adatbázisterv, hanem a létező legjobb adatbázisterv.***

Magyarázat: Az előző pontokban kimutattuk, hogy az adatmodell és az adatbázisterv tartalmilag nem teljesen fedi egymást. Arról is volt szó már, hogy különböző okok miatt a fogalmi szinten kialakított képet a tervező *tudatosan* lerontja a logikai szinten. Itt a hangsúly a kiemelt jelzőn van: csak azt lehet tudatosan lerontani, amit előzőleg tudatosan jól építettünk fel. Vagyis az adatmodellezés során törekednünk kell a tökéletességre. Erre akkor van módunk, ha tisztában vagyunk a lehetséges problémákkal ( $\rightarrow$  *Az adatmodellek hibái*). Azokat pedig akkor érthetjük meg, ha ismerjük az optimumkritériumokat.

**Érthetőség.**     **5.2 példa**     SKK (Szerződés-az+Kote-sorsz)

Magyarázat: Az olvasó aligha tudja, hogy a Kote-sorsz kötelezettség sorszámot jelent. Egy szerződéshez több kötelezettség is tartozhat. Az SKK tábla a szerződéssel kapcsolatos feladatok adatait rögzíti. Ezt a tervrészletet a könyv írója a „Szerződések adatmodellje” c. irományból vette át. Az olvasó megérti, hogy ebben az esetben az adatmodell megjelölést nem tudja elfogadni. A modell legelső kritériuma az érthetőség. Az adatbázistervben lehet rövidített, magyartalan, érthetetlen neveket alkalmazni. Az adatmodell szintjén ez kizárt.

**Valósághűség. 5.3 példa** SZEMÉLY (Személy az, Legmagasabb iskola)  
SZEMÉLY (Személy az, Név-1, Név-2)  
KOCSI (Rendszám, Kocsitípus)

Magyarázat: A legmagasabb iskola itt nem szintet jelent, hanem a konkrét iskolára kell hivatkozni. Mivel vannak többdiplomások is, a tervrészlet nem valósághű: nem engedi meg, hogy az adatbázisban egynél több egyetem nevét adják meg a személyhez. Általában a *szervezeti* torzítás a számszerű, mennyiségi viszonyok félreértelmezésének köszönhető. A második esetben az olvasó nem sejteti, hogy a Név-2 rovatba a foglalkozást írják be! Ez már minőségi baj, amely a *tartalommal* függ össze. A harmadik példarész mögöttes tartalma is rejtett. A Kocsitípus adat neve nem véglegesen torzít, de a benne alkalmazott kódértékek mint jelsorok nem azok, amiket a felhasználó alkalmazna (*formai* torzítás).

**Egyértelműség. 5.4 példa** VEVŐRENDELÉS (Rendelésszám, Rendeléstípus)  
SZÁLLÍTÓRENDELÉS (Rendelésszám, Rendelésfajta)

Magyarázat: A kétféle rendelésnek nincs köze egymáshoz. Nem lehet azonos a kulcsuk. Nem is az, csak a név közös, vagyis *homonima*. A másik két adat azt tartalmazza, hogy milyen módon adták fel a rendelést. Nevük eltérő, de lényegük azonos, tehát *szinonimák*. Már az eddigiekből is látszik, hogy a kritériumok nem függetlenek egymástól. Ha a terv nem egyértelmű, akkor nem igazán valósághű illetve érthető. Azért nem érthető, mert nem valósághű stb. Ha az egyértelműség követelménye nem teljesül, akkor a következő két kritérium is sérelmet fog szenvedni.

**Teljesség. 5.5 példa** VEVŐ (Vevőkód)  
CIKK (Cikk azonosító)  
RENDELÉS (Rendelésszám, Vevőnév)  
RENDELÉSTÉTEL (Rendelésszám+Cikkszám)

Magyarázat: A rendelésből hiányzik a Vevőkód, ezért nem kapcsolható a vevőhöz. Ez a hiányosság nyilvánvaló: a modellrész nem teljes. A rendeléstétel látszólag nem köthető a cikkhez, mert nincs közös adatuk. Valójában van, de nem látható, hogy a Cikk azonosító és a Cikkszám ugyanazt jelenti (szinonimák). Tehát a teljesség nem vizsgálható akkor, ha a modell nem egyértelmű.

**Minimalitás. 5.6 példa** VEVŐ (Vevőkód, Vevőtípus, Vevőnév)  
RENDELÉS (Rendelésszám, Vevőtípus, Vevőnév)

Magyarázat: A Vevőnév redundanciája világosan látható: a modellrész nem minimális. Az olvasó úgy látja, hogy a Vevőtípussal hasonló a helyzet. Eláruljuk, hogy az az első egyedben a személyes, szervezeti stb. vevőt jelenti, míg a másodikban arra utal, hogy a számlát majd kész-

pénzzel, átutalással stb. fogják-e kiegyenlíteni (homonima). Vagyis a minimalitás sem vizsgálható akkor, ha a modell nem egyértelmű.

A modellel szemben léteznek egyéb követelmények is, amelyeket csak futólag említünk meg. Ilyen a **robosztusság**, ami a darabszámokkal kapcsolatos. Olyan modellt kell alkotni, amelynek megvalósítása esetén a táblák méretei nem okoznak kezelési gondokat. Másik kritérium a **bővíthetőség**, ami a struktúrára vonatkozik. A modellt úgy kell felépíteni, hogy újabb tényezők hozzáadása esetén lehetőleg ne kelljen a már meglévőkhöz nyúlni.

Módszerek: A további fejezetekben részletesen be fogjuk mutatni azokat az eljárásokat, amelyekkel a kritériumokat sértő hibák kiszűrhetők. A módszerek részben matematikai (mennyiségi), részben szemantikai (minőségi) természetűek. A kritériumokat tudatosan ismertettük a fenti sorrendben. Felülről-lefelé egyre több szerep jut a matematikának és egyre kevesebb a szemantikának. A valóságghűség nem vizsgálható matematikai módon, a teljesség és a minimalitás annál inkább. Természetesen a legjobb módszer nem más, mint a tiszta gondolkodás és a józan ész párosa. Nincs szükség matematikára és szemantikára ahhoz, hogy érthető, tisztán elrendezett modellt alkossunk.



## 6. AZ ADATMODELLEK HIBÁI

### 6.1 Az adatbázis hibáinak a forrásai

Alapvetően három oka lehet annak, ha az *adatbázisunkból* vagy nem nyerjük vissza a szükséges ismeretet, vagy az nem a várt tartalommal jelenik meg. Az első ok általában az, hogy az adatkezelő programokat nem megfelelően írták meg. A program működik, csak éppen nem a kellő eredményeket szolgáltatja (vö. pragmatika). Ez visszavezethető például arra, hogy a korlátok ellenőrzését nem vagy nem jól hajtja végre a vonatkozó eljárásrész. A második ok többnyire az, hogy a felhasználó vagy nem töltötte ki, vagy tévesen vitte be a kérdéses konkrét adattartalmakat. Nem ritkán ekkor is a korlátoknál kell keresni a bajt, amennyiben a tervezők nem vagy nem eléggé szigorúan határozták meg azokat. Ezekkel a jelenségekkel a továbbiakban nem foglalkozunk.

Az adatbázis nem csak a helytelen adatkezelés miatt lehet hibás. Tökéletlen állapotának legtöbbször a szerkezetet meghatározó *adatmodell* nem megfelelő kialakítása az oka. Így például tipikus jelenség, hogy a modell redundanciát mutat, ami azzal jár, hogy ugyanaz az adat(sor) egynél több példányban szerepel az adatbázisban. Az egyiken átvezetik az új tartalmat, a másikon nem - és máris megsérül az adatbázis épsége, hiszen ellentmondásos ismereteket fog tárolni. A redundancia a sokféle hibának csak az egyik fajtája. Azonban rá is jellemző az, ami a többi modellszintű problémára: sokkal nagyobb gondokat okozhat, mint az adatkezelés hiányosságai.

Emiatt az adatbázis tervét nagyon gondosan kell összeállítani. Természetesen ahhoz, hogy kellő körültekintéssel végezhessük ezt a munkákat, ismernünk kell az elkerülendő csapdákat. Jó modellt nem lehet ösztönösen alkotni. A rossznak a tudatos elkerüléséhez mindig tisztában kell lennünk a problémák gyökereivel.

*Ennek a fejezetnek az a célja, hogy bemutassa az adatmodell általános, tipikus hibáit és rávilágítson ezeknek a bajoknak a tervezési szemléletben rejtőző végső okaira.*

Ebben a célmeghatározásban nagyon fontos mozzanat a tervezési szemlélet. Ugyanis a tervező hiába érti *technikailag* az adatbázis-tervezés csínjait-bínjait. Ha nem sajátítja el a kemény önfegyelmet feltételező korrekt *szemléletet*, akkor nem lesz képes jó adatbázist építeni. Azért nem, mert az adatbázis mindig kettős rendszer. Az alkalmazói adatbázis szerkezetét és korlátait a fejlesztői adatbázis, vagyis az adatmodell határozza meg. Ennek hibái a felhasználás közben már nem korrigálhatók.

Ez a fejezet csak két nagy témakörrel foglalkozik: 6.2 *Tipikus adatmodellezési hibák* és 6.3 *Az adatmodellezési hibák gyökerei*.

### 6.2 Tipikus adatmodellezési hibák

A rendszerfejlesztésben olykor különbséget szoktak tenni a tervezés és az elemzés fázisai illetve tevékenységei között. Ezzel szemben az adatbázis-terv mindig úgy születik, hogy azt folyamatosan

értékeljük, mert egy lényegtelennek látszó részlethiba miatt az egész terv felborulhat. Ezért az adatmodellezés és a modellelemzés valójában ugyanazt jelenti. Nem először modellezünk és azután elemzünk, hanem a kettőt együtt végezzük.

Az adatmodell meg kell, hogy feleljen bizonyos **formai** kritériumoknak (← Adatmodell diagram), mert az értelmezhetetlen terv nem elemezhető. Viszont semmi akadálya sincs annak, hogy valaki egy alakilag tökéletes leírásban teljesen rossz modellt állítson össze. Ezért nagy figyelmet kell szentelni a **tartalmi** követelményeknek. A modellezés nem más, mint a legjobb adatbázisterv kemény munkával való kialakítása. Ahhoz, hogy a legjobb tervet összeállíthassuk, ismerni kell a vezérlőelveket (← Optimumkritériumok). Ezek betartásához viszont meg kell érteni a modellekben fellépő hibák természetét.

A továbbiakban az alábbi közös példán keresztül fogjuk szemléltetni azt, hogy valóban lehet formailag megfelelő, de tartalmilag nagyon rossz modellt készíteni. Az eset rávilágít az általános adatmodellezési problémákra, és ezzel közvetve az alapvető kritériumok szemléltetésére is szolgál.

#### TULAJDONOS

Törzsszám	Típus	Tulajdonos	Foglalkozás	Telephely	Felügyelet
111111	Magán	Kovács R.	Ápolónő 11	Budapest	-
222222	Kft	AB Kft.	-	Szeged	-
333333	Magán	Kovács R.	Mérnök 32	Pécs	-
444444	Rt	XY Rt.	-	Budapest	Z

#### KOCSI

Rendszám	Típus	Tulajdonos	Foglalkozás	Telephely	Főhatóság	Férőhely
ABC 134	Lada	Szabóné R.	Ápolónő Á	Budapest	-	5 fő
BCD 265	BMW	AB Kft.	-	Szeged	-	5 fő
DEF 896	Lada	Kovács R.	Mérnök M	Pécs	-	5 fő
FGH 333	Polski	XY Rt.	-	Budapest	Z	4 fő

#### KÁR

Kárszám	Dátum	Típus	Tulajkód	Kárösszeg
23000	99.05.14	Lada	111111	X
23000	99.05.14	BMW	222222	Y
34000	99.06.06	Lada	333333	Z
45000	99.06.31	Lada	111111	Q

6.1 ábra: „Beteg ló” mintaadatbázisunk részlete

#### 6.2.1 Nyílt logikai átfedés

+

Ha egy tulajdonság több egyedhez is azonos névvel kapcsolódik úgy, hogy mindegyikben azonos tartalmú, de egyikben sem azonosító szerepű, akkor nyílt logikai átfedésről beszélünk.

KOCSI

Rendszám	Típus
ABC 134	Lada
BCD 265	BMW
DEF 896	Lada
FGH 333	Polski

KÁR

Kárszám	Típus
23000	Lada
23000	BMW
34000	Lada
45000	Lada

6.2 ábra: Példa a nyílt logikai átfedésre

Magyarázat: A 6.1 ábra mindhárom egyedét jellemzi a Típus tulajdonság, méghozzá olyan módon, hogy egyikben sem azonosító szerepű. A KOCSI és a KÁR egyedekben a Típus értelme/tartalma azonos: mindkettőben a kocsi fajtáját jelenti. A redundancia nyílt, ami azt jelenti, hogy látható. Azért nevezzük logikainak, mert nemcsak a tartalmakban (→ *Fizikai redundancia*), hanem magában a szerkezetben is megmutatkozik.

Probléma: A fizikai redundancia miatt feleslegesen fogy a tárolóhely. Többszörösen kell ugyanazokat az ismereteket kezelni, esetleg többszörösen megírt programokkal. Ha a változtatást nem egyidejűleg hajtjuk végre az összes példányon, akkor az adatbázis ellentmondásos - inkonzisztens - lesz. Ha valaki nem a legfrissebb példányt kérdezi le, akkor nem jut aktuális ismeret-hez. Mindez nem látszik annyira a Típus esetében, ami nem változékony sajátosság. Viszont a példában a Tulajdonos neve is nyíltan átfedő tétel. Ennek értékeinek többszörös tárolása, bevitele, módosítása már aligha lenne kívánatos. A Kovács R. név (lásd a TULAJDONOS egyed első sorát) Szabóné R.-re változott (lásd a KOCSI egyed első sorát). Nem tudhatjuk, hogy csak tökéletlenül végrehajtott frissítésről van-e szó, vagy a gépkocsi ténylegesen gazdát cserélt.

—

**Az adatmodellben célszerű kerülni a nyílt logikai átfedéseket.**

A szabály nem szigorú. Az ilyen redundanciákat „célszerű” kerülni. Jóllehet Codd úr - a relációs adatbázisok atyja - megkövetelné, hogy minden tulajdonság minden egyedben eltérő nevet kapjon, sarkos nézetével nem érthetünk egyet. Hiszen előfordulhat, hogy több egyedben is szerepel például a Mértékegység tulajdonság teljesen azonos tartalommal.

Kiegészítés: Két egyednek nem lehet ugyanaz az *azonosítója*. Ha pedig a tulajdonság az egyik egyedben azonosító, a másikban nem-azonosító *relatív szerepű*, akkor a jelenség nem tekinthető „rossz” redundanciának: éppen ellenkezőleg (*Kapcsolótulajdonság* →). Ezért nyílt logikai átfedés valóban csak a nem-azonosító tulajdonságok között léphet fel.

### 6.2.2 Látszólagos logikai átfedés

+

**Ha egy tulajdonság azonos névvel, de eltérő tartalommal kötődik több egyedhez, akkor látszólagos logikai átfedésről beszélünk.**

TULAJDONOS

Törzsszám	Típus
111111	Magán
222222	Kft
333333	Magán
444444	Rt

KOCSI

Rendszám	Típus
ABC 134	Lada
BCD 265	BMW
DEF 896	Lada
FGH 333	Polski

6.3 ábra: Példa a látszólagos logikai átfedésre

Magyarázat: A Típus a KOCSI egyedben a kocsi fajtáját jelöli, ámde a TULAJDONOS egyedben a birtokos természetes vagy jogi személy jellegét (az utóbbin belül is a konkrét cégformát) mutatja. Az átfedés azért csak „látszólagos”, mert az azonos nevek eltérő tartalmakat rejtnek, vagyis a tulajdonos és a gépkocsi típusa nem vesz fel azonos értéket. Ebben az esetben az úgynevezett *homonima* jelenséggel állunk szemben.

Probléma: A látszólagos redundancia egyértelműségi problémákat okoz. Tegyük fel, hogy valaki a következő feladatot kapja: „Készítsen kimutatást a tulajdonosokról és a kocsikról típus szerint!”. Vajon mit fog tenni? Elkészíti a jelentést kocsitípus szerint, holott olyat kértek tőle, amely a tulajdonosok típusai szerint tagolt.

A formai azonosság - közös név - miatt a látszólagos és a nyílt átfedést könnyen össze lehet téveszteni. Ebből két baj adódhat. Például két egyedben is szerepel a Mértékegység nevű tulajdonság, de más értékkészletekkel. A tervező redundanciát sejtve megszünteti az egyik tételt és ismeretet veszít. (A rendelésben üvegben kérték az üdítőitalt, a szállításban viszont kannát jelöltek meg egységként.) Ha pedig homonim kulcsokat használunk, akkor a kapcsolatok lesznek tisztázatlanok. Például ha a vevői és szállítói rendelések kulcsának egyaránt a Rendelészám nevet adjuk, akkor a modell valamelyik számla egyedében lévő Rendelészám leíró adatról nem tudhatjuk, hogy a kettő közül melyikhez kapcsol.

— **Az adatmodellből feltétlenül ki kell küszöbölni a homonimákat.**

Magyarázat: A szabályt nagyon komolyan kell venni. A látszólagos átfedések nem csak a modellelemzést, a teljes és minimális adatmodell kialakítását teszik lehetetlenné (lásd normalizálás és optimumkritériumok). Az adatbázis szerkezete az idők során változik. Az átalakítás munkája fog kínná válni, ha nem alkalmazzunk egyértelmű neveket. Utólag már nem fogjuk tudni helyesen értelmezni a tartalmakat.

Kiegészítés: Ha minden egyed minden tulajdonságának más nevet adunk, akkor nem léphet fel a homonimaprobléma. Ámde annál nagyobb gond lesz a rejtett logikai átfedés.

### 6.2.3 Rejtett logikai átfedés

+ **Ha egy tulajdonság eltérő névvel, de azonos tartalommal kötődik több egyedhez, akkor rejtett logikai átfedésről beszélünk.**

TULAJDONOS		KÁR	
Törzsszám		Kárszám	Tulajkód
111111		23000	111111
222222		23000	222222
333333		34000	333333
444444		45000	111111

6.4 ábra: Példa a rejtett logikai átfedésre

Magyarázat: A nevek alapján az olvasó nem találta volna ki, hogy a TULAJDONOS egyed Törzsszám tulajdonsága és a KÁR egyed Tulajkód attribútuma valójában egy és ugyanazon dolgot jelent. Pedig a konkrét értékek egyértelműen ezt mutatják. Tehát a terv redundanciát tartalmaz, de az rejtve marad előlünk. Ha két eltérő név tartalma azonos, akkor a *szinonima* jelenséggel állunk szemben. A Törzsszám és a Tulajkód - szinonimák.

Probléma: A szinonima egyértelműségi gondokat okoz. A következő igény zavarba hozhatja az embert: „Készítsen jelentést a károkról a törzsszám sorrendjében!”. Hiszen nincs is a KÁR egyedben Törzsszám nevű adat! Emellett ugyanazok a problémák lépnek fel, mint a *nyílt logikai redundancia* esetében. Az egyik egyedben átírjuk a Főhatóság adott értékét, de a másikban nem frissítjük a vele azonos lényegű Felügyelet tartalmát. Az egyértelműség hiánya lehetetlenné teszi a normalizálást illetve megnehezíti a kapcsolatok feltárását. Például a tervező az eltérő név miatt nem veszi észre, hogy a TULAJDONOS és a KÁR egyed egymáshoz kapcsolódik a közös tartalmú Törzsszám és Tulajkodó tételen.

— **Az adatmodell első változatából ki kell küszöbölni a szinonimákat.**

Magyarázat: A modellelemzés lezárásáig nem szabad szinonimákat használni. Mivel azonban a különböző felhasználók esetleg eltérő nevek alatt szeretnék látni ugyanazokat a tartalmakat, az elemzés befejezése után bevezethetünk **kontrollált szinonimákat**. Ez azt jelenti, hogy kiválasztunk egy elsődleges nevet (pl. Mértékegység) és a dokumentációban ahhoz kötötten jelöljük meg a másodlagos - szinonim - neveket (pl. Mennyiségi egység).

Kiegészítés: Ha ragaszkodnánk ahhoz az ötlethez, hogy minden tulajdonságnak minden egyedben eltérő nevet kell adni, akkor magunk kreálnánk mesterségesen a szinonimákat. Ez káoszra vezetne. Például volna Vevő-rendelészám és Szállító-rendelészám kétféle nevű és jelentésű adatunk, amely esetben a minősítés feltétlenül kívánatos. Viszont lenne egy Számla-rendelészám nevű tulajdonságunk is, amelyről azután már senki nem tudja eldönteni, hogy a két előbbi közül melyikkel azonos lényegű.

#### 6.2.4 Technikai homonimák és szinonimák

+ **Technikai homonimáról van szó akkor, ha két azonos nevű tulajdonság értékkészlete különbözik. Technikai szinonimáról beszélünk akkor, ha két eltérően írt nevű tulajdonság értékkészlete azonos.**

TULAJDONOS

Törzsszám	Foglalkozás
111111	Ápolónő 11
222222	-
333333	Mérnök 32
444444	-

KOCSI

Rendszám	Foglalkozás
ABC 134	Ápolónő Á
BCD 265	-
DEF 896	Mérnök M
FGH 333	-

6.5 ábra: Példa a technikai homonimára

Magyarázat: Példánk a Foglalkozás tulajdonságok alatt kettős értékeket tartalmaz. Az egyik a foglalkozás megnevezése, a másik annak kódja. Annak dacára, hogy a név és az amögötti elvi jelentés is azonos, az értékkészlet eltérő. Tehát a két tétel homonima, bár nem szántuk annak. Ezért minősítjük azt a „technikai” jelzővel.

Probléma: A technikai homonima az adatbázisban valósként viselkedik. A legkisebb baj az, hogy kétféle kódtáblát kell egyszerre kezelni. Nagyobb gond, hogy nem egyértelmű a lekérdezés, nem lehet összekapcsolni az ismereteket (ha azonosítóról van szó). Mindez pedig a neveken alapuló adatmodellel egyáltalán nem is látszik. Ezért tanácsos, hogy a modellt mindig a fentihez hasonló értékmintákkal is szemléltessük.

**6.1 példa** PÁLYÁZAT (Pályázat-az, Pályázat dátuma)  
BÍRÁLAT (Bírálat azonosító, Ályázat az, Pályázati dátum)

Magyarázat: Nem történt elírás, szándékosan használtuk a fenti neveket. A két egyed két megegyező tartalmú, de eltérő nevű tényezőt is tartalmaz. A Pályázat-az/Ályázat az illetve a Pályázat dátuma/Pályázati dátum tulajdonságok technikai szinonimák.

Probléma: A példa három hibát rejt. Technikai szinonimák származhatnak az eltérően ragozott nevekből, aminek következtében nem fedezhető fel a pályázat keltének a nyílt logikai redundanciája. A bajt okozhatja a gondatlanságból fakadó elírás is, ami miatt nem fedezhető fel a két egyed viszonya (*A logikai átfedés hiánya* →). Igen gyakori hiba a következetlen „konvenció”. Ha már az azonosítót az egyik helyen rövidítve és kötőjellel nevezzük meg, akkor a másik helyen ne írjuk ki teljesen és kötőjel nélkül.

— ***Az adatmodell nem tartalmazhat technikai homonimát és szinonimát.***

Magyarázat: A logikai tervezés szintjén - például az adatkezelő korlátai miatt - nem kizárt, hogy magyartalan (ékezet nélküli), rövid, a tartalmat ki nem fejező neveket kell alkalmaznunk. A fogalmi modellezés szintjén erre semmi sem kényszeríthet bennünket. Egyedül és kizárólag a tervező gondosságán múlik a technikai homonimák/szinonimák elkerülése. Sajnálatos módon a fejlesztők maguk vezetnek be ostoba „konvenciókat”, mint például a tulajdonság nevének az egyednév kódjával való minősítését.

Például K-Típus jelöli a kocsitípust, T-típus a tulajdonostípust, amivel ugyebár el lehet kerülni a homonimát. Valóban? És mi lesz a Kártípus neve? K-típus már nem lehet, tehát - ravaszul - az R-típus nevet adják neki. Ami után a rendelések nem osztályozhatók ezzel a névvel. Az effajta minősítési elv a fogalmi adatmodell szintjén teljesen kizárt, de nincs semmi értelme/haszna a logikai adatbázisterv szintjén sem.

Kiegészítés: A tervezőnek mindig szem előtt kell tartania a következő axiómát:

— ***Két modellezési tényező akkor és csak akkor azonos, ha elvi értelmük, nevük és előforduláshalmazuk százszázalékosan megegyezik.***

**6.2.5 A logikai átfedés hiánya**

+ ***Ha az adatmodell két elvileg összefüggő egyede nem kapcsolható, mert nincs közös kapcsoló tulajdonságuk, akkor a logikai átfedés hiányáról beszélünk.***

Magyarázat: Az *inkonnektivitás* problémájáról már volt szó (← *Kapcsolathiány*). Ha az adatbázis az előfordulások szintjén nem konnektív, akkor a baj nem feltétlenül szerkezeti eredetű, azt okozhatja a helytelen frissítés is (integritási gond). Például ha a Tulajdonos alapján akarnánk összekapcsolni a kocsit a birtokosával, akkor nem jutnánk sokra, mert Szabóné R.-ről nem tudhatjuk a kezelőrendszer, hogy az azonos Kovács R.-rel.

A meghatározás arra a helyzetre utal, amelyben a kapcsolat megteremtése az alapvető strukturális hiányosság miatt eleve elképzelhetetlen. Nem rossz ismerethez jutunk, hanem egyáltalán

nem nyerhetünk információt. A 6.1 ábrán a KÁR egyedben nem történik utalás magára a kocsira. Ezért sohasem fogjuk megtudni, hogy pl. a 23000 számú eseményben mely konkrét járművek vettek részt. Ez csak akkor lett volna lehetséges, ha a KÁR egyed tartalmazta volna a kocsik rendszámait is.

Probléma: Az, hogy nem kapunk ismeretet csak egy dolog. Az pedig már másik, hogy a korrekt kapcsolat hiánya miatt inkorrekt összefüggések alapján kezdünk nyomozni. Az ügyintéző egy régi esetre emlékezve azt a tulajdonost nyaggatja (rá történik hivatkozás a KÁR egyedben), aki időközben már régen eladta azt a kocsiját, ami a kár részese volt. Ez nem fordulhatna elő, ha az egyed a közvetlen érintett jelenséghez (kocsi) kapcsolnák és csak azon keresztül lehetne eljutni a közvetetten érintett jelenséghez (tulajdonos).

— **Az adatmodellben nem engedhető meg kapcsolati hiányosság.**

#### 6.2.6 Fizikai átfedés

+ **Fizikai redundanciáról akkor beszélünk, ha az egyedelőfordulásokban azonos tulajdonságérték részsorok jelennek meg egy - nem feltétlenül expliciten jelenlevő - további tulajdonságtípus értékétől függően.**

KOCSI

Rendszám	Típus	Férőhely
ABC 134	Lada	5 fő
BCD 265	BMW	5 fő
DEF 896	Lada	5 fő
FGH 333	Polski	4 fő

6.6 ábra: Példa a fizikai redundanciára

Magyarázat: A korábbi alpontokban mindig a tulajdonságtípusokra, az általuk alkotott struktúrákra figyeltünk. Mindig a nevek kettőssége - vagy éppen hiánya - mögött kerestük a bajokat. Azonban redundancia felléphet az értékek szintjén úgy is, hogy a modell nem mutat kettős neveket (logikai átfedés). Ha van százezer Lada típusú kocsi, akkor ennyiszer fogjuk tárolni a Lada - 5 fő ismeretpárost, fizikailag redundánsan. Ha az ábrában nem szerepelne a Típus tulajdonság - nem volna expliciten is jelen -, a megoldás akkor is rossz lenne, mert a rejtetten - impliciten - azonos típusú kocsik férőhelye ettől még ugyanaz.

Probléma: A fizikai átfedés ugyanazon tárolási és karbantartási gondokkal jár, mint a *nyílt logikai átfedés*. Baj az is, hogy a jelenséget nehéz megérteni és feltárni. Mivel egy egyed típusnak nem lehet két teljesen azonos előfordulása (azok egynek tekintendők), nem kell számolni teljes sorok azonosságával. A másik oldalon - szemben az azonosítókkal - a leírók több előfordulásban is lehetnek azonos értékűek. Tehát pusztán az a tény, hogy több sorban is megjelenik a Lada vagy az 5 fő érték, még nem okoz redundanciát. Ezért szól úgy a definíció, ahogyan megfogalmaztuk: a fizikai átfedés részsorokkal kapcsolatos.

#### 6.2 példa RENDELÉS (Rendelészám, Vevőcím)

Magyarázat: A részsorok átfedését akkor nehéz felfedezni, ha nincs jelen expliciten is a redundanciát mutató tulajdonság. Ha a RENDELÉS egyedbe „belelátnánk” a Vevőkódot is, akkor már világos lenne, hogy a Vevőkód - Vevőcím tulajdonságpár mögött azonos értékpárok is rejtőznek. Ezért fogalmazzuk meg az alábbi módon a következő szabályt:

— **A fogalmi adatmodell nem rejthet fizikai átfedéseket.**

Kiegészítés: A logikai tervben hatékonysági és egyéb okokból tudatosan bevezethetünk fizikailag redundáns adatokat. A tudatosság viszont azt feltételezi, hogy e lépés előtt már látjuk az optimális, átfedés-mentes megoldást.

### 6.2.7 Kiegyensúlyozatlanság

+ **Egyensúlytalan a szerkezet akkor, ha az adatmodell az elvileg egymás mellé rendelt egyed típusok közül az egyiknek a kezelését a másikénál jobban támogatja, vagyis azokból alá/fölérendelt párost kreál.**

KÁR

Kárszám	Dátum	Típus	Tulajkód	Rendszám
23000	99.05.14	Lada	111111	ABC 134
		BMW	222222	BCD 265
34000	99.06.06	Lada	333333	DEF 896
45000	99.06.31	Lada	111111	ABC 134

6.7 ábra: A módosított KÁR egyed típus

Magyarázat: A szemléltetés kedvéért átalakítottuk a 6.1 ábra példáját. Az ábrán az első két sort egyetlen előfordulásként kell értelmezni. Egy kár két kocsit érintett (← *Ismétlődő csoport*), ezért nem ismételtük meg a Kárszám és a Dátum adatok értékét. Most pedig bármennyire is nehéz, az olvasó tételezze fel, hogy nincs külön KOCSI egyed típusunk! Tehát a kocsik ismereteit csak ebben az egyedben vezetik.

Probléma: A kép féloldalas. Könnyen ki tudjuk keresni, hogy egy káreseményben mely autók vettek részt, mert a Kárszám szolgál azonosítóként. Viszont nehezen tudjuk meg, hogy egy kocsin milyen károkból volt részes, mert a Rendszám csak leíró adat. Egy kocsin több (M) kárban vehet részt, egy esemény több (N) kocsira vonatkozhat. A két jelenség M:N fokú hálós viszonyban áll egymással, amely hálót a fenti példa az egyszerűbb 1:N fokú hierarchiára redukálta. Így a megoldás nem valóságghű (← *Optimumkritériumok*).

Erre a problémára is vissza fogunk térni a normalizálás kapcsán. Itt csak arra hívjuk fel a figyelmet, hogy az egyensúlytalan szerkezet előbb-utóbb a modell átszervezését váltja ki.

— **Az adatmodellben nem lehetnek kiegyensúlyozatlan részszerkezetek.**

Kiegészítés: Egyesek most nem érzékelik a példa mögötti kiegyensúlyozatlanság összes következményét. Azt mondják, hogy a Rendszámra épített index megoldja a problémát. Ez elvi és gyakorlati tévedés. Elvi, mert az adatmodell szintjén még nincs szó indexről, az a fizikai



tervezésre tartozó tényező. Gyakorlati, mert az elérés csak az egyik probléma. A másik a fizikai átfedés, ami jól látszik az ABC 134 kocsinál. Ennél is nagyobb baj az, hogy az ilyen szerkezet nem bővíthető. Előbb-utóbb a kocsikról olyan ismeretet is vezetni akarnak, amit nem vesznek fel a káresemények kapcsán. Tehát létrehozunk egy KOCSI egyedtípust, ami által már nyilvánvalóvá lesz, hogy a fenti struktúra tarthatatlan.

### 6.2.8 Inkorrekt nevek

Háttér: Ez az alpont nem vet fel újabb problémát, inkább összegzésként szolgál. Az adatmodell fogalmak alkotják. A fogalmakat nevek hordozzák. Ezért roppantul fontos a nevek érthetősége, valóságghűsége és egyértelműsége.

#### 6.2 példa TULAJDONOS (Törzsszám, Telephely) KOCSI (Rendszám, Telephely)

Magyarázat: A pesti telephelyű vállalatnak lehet Érden tárolt kocsija is. Annak ellenére, hogy nevük és értékkészletük azonos, a két Telephely fogalom jelentése más, ezért a példa homonimát tartalmaz.

Probléma: Ha a második egyedben át is írjuk a tényező nevét Kocsi-telephelyre, az nem fogja kizárni, hogy a felhasználó a név alatt a tulajdonos telephelyét adja meg. Azért nem, mert nem látja át a két fogalom különbségét. Az efféle integritási hibákat csak egy módon lehet csökkenteni: a képernyőn pontos eligazításokat adunk a tárolt definíciók alapján.

— ***Az adatmodell tényezőit a beszélő nevek mellett rövid értelmező magyarázattal is el kell látni.***

Magyarázat: A fogalmi modell nemcsak struktúráknak, nemcsak tételesen megfogható korlátoknak, hanem egyben értelmező magyarázatoknak is a szervezett együttese. Mivel a fogalmak ismeretlenek, nem önmagyarázóak, azokról nem szabad feltételezni a közös, egyértelmű és valóságghű értelmezést. Tehát definíciókra van szükség.

#### 6.3 példa „Gabi kocsija Lada.”

Probléma: A modellezésben szabály, hogy ugyanaz a név nem lehet több másíknak is a szinonimája. Ha Gabit mondunk, akkor nem tudható, hogy kire gondolunk: Gáborra vagy Gabriellára. Ha egy név több másíknak is szinonimája, akkor az valójában félrevezető homonima. A kocsi példában a Tulajdonoskód lehet a Törzsszám szinonimája, de az már nem megengedhető, hogy egyben a Tulajdonostípus szinonimája is legyen.

Kiegészítés: A modellezésben nem a minimális és teljes szerkezet megtalálása jelenti a legnagyobb kihívást. A modellezőnek leginkább magával és társaival kell megküzdenie, hogy mindannyian korrekt nevek alkalmazására kényszerítsék magukat és egymást. Az optimális terv megalkotását ugyanis nagyban akadályozzák azok a váratlan szinonimák és homonimák, amelyekre a fentiekben mutattunk két példát.

### 6.2.9 A példa megoldása

Az olvasónak most megadjuk a 6.1 példa helyes megoldását:

#### TULAJDONOS

Törzsszám	Tulajtípus	Tulajnév	Foglalkozás	Telephely	Felügyelet
111111	Magán	Kovács R.	Ápolónő Á	Budapest	-
222222	Kft	AB Kft.	-	Szeged	-
333333	Magán	Kovács R.	Mérnök M	Pécs	-
444444	Rt	XY Rt.	-	Budapest	Z

#### KOCSI

Rendszám	Kocsitípus	Törzsszám	Tárolóhely
ABC 134	Lada	111111	Budapest
BCD 265	BMW	222222	Szeged
DEF 896	Lada	333333	Pécs
FGH 333	Polski	444444	Veszprém

#### KOCSITÍPUS

Kocsitípus	Férőhely
BMW	5 fő
Lada	5 fő
Polski	4 fő

#### KÁR

Kárszám	Dátum
23000	99.05.14
34000	99.06.06
45000	99.06.30

#### KÁR/KOCSI

Kárszám	Rendszám	Kárösszeg
23000	ABC 134	X
23000	BCD 265	Y
34000	DEF 896	Z
45000	ABC 134	Q

6.8 ábra: A hibákat kiküszöbölő adatmodell

Magyarázat: A járatlan olvasó számára a 6.1 ábra példája az első áttekintésre nem is tűnt annyira rémesen rossznak. A fejezet során kiderült, hogy jóformán nincs egyetlen egy ép porcikája sem. Pedig az összes problémát nem is tártuk fel. Az olvasónak remélhetőleg feltűnt, hogy a volt KÁR egyed negyedik sorában megjelenő dátum - 99.06.31 - helytelen, megengedhetetlen értéket tartalmaz. Azt, hogy miképpen lehet eljutni az itt mutatott jó megoldásra, majd a következő fejezetek során fejtjük ki.

Kérdés: Lehet, hogy a 6.8 ábra modellje nem teljes? Nem kellene kódolni a Tulajtípust létrehozva egy TULAJDONOSTÍPUS egyedet?

Válasz: Először is a kódolás nem fogalmi szintű kérdés. Másodszor a tervezők túlzottan gyakran alkalmaznak „kódegyedeket”, holott egyednek az ismerettel leírni kívánt dolgot nevezzük. A kocsitípust saját ismeretekkel írtuk le, de nem is „kódegyeddel” tükröztük! Ha csak az értékkészlet ellenőrzésére van szükség, akkor nem kell létrehozni új egyedet, hanem az általános tulajdonság fogalmi tényezőjéhez kell fordulni (→ Értéktartomány).

## 6.3 A modellezési hibák gyökerei

Megfelelő *modellezési technikák* használatával részben elkerülhetjük azt, hogy nagyon rossz adatbázisterveket készítsünk. Ennek ellenére láthatni olyan, a legkorszerűbb CASE eszközök

segítségével készült „adatmodelleket”, amelyek minden szempontból megsértik az optimum-kritériumokat. Ennek a helytelen *modellezési szemléletmód* az oka. Nincs az az eszköz, technika, módszer, amely garanciát nyújtana a hibák ellen. A tervezőnek négy ponton kell szemléletmódot váltania, ha optimális adatmodellt óhajt készíteni.

### 6.3.1 Nézetvakság

A fejlesztők nem figyelnek a gyakorlatban az *absztrakciós vetületek* elméletére. Nem a valós jelenségeket tükrözik az adatmodellben, hanem azt, ahogyan a valóságot az őket éppen aktuálisan megbízó felhasználó látja. A 6.1 ábra KÁR egyede a káreseményekben érdekelt ügyintéző oldaláról nézi az összefüggéseket („kár  $\sqcap$  kocsi”). Az eszébe sem jut, hogy lesz majd olyan alkalmazás is - vagy már folyik is a fejlesztése -, amelyben pont az ellentétes irányú („kocsi  $\sqcap$  kár”) nézet dominál. Eláruljuk, hogy a vonatkozó ábra példája azért olyan rossz, mert három egymástól független szervezeti egység dolgozta ki a tervet, egymásról nem is tudva. Csak mi töltük az egységeket egymás mellé, didaktikai okokból.

A bajokon a CASE használata semmit sem segít. Az egymástól független projekteket a segédletekben teljesen szeparáltan lehet dokumentálni, ezért nincs integráló erő. Csak egy dolog óvhat meg bennünket a teljesen rossz tervektől. Ha elsajátítjuk az alábbi szabályt:

— *Az adatmodell - egy*

Magyarázat: Erre az axiómára már utaltunk. Azonban nem győzzük hangsúlyozni a korrekt szemlélet elsajátításának a fontosságát. Erre két szinten van szükség. Egyrészt az *egyéni* fejlesztő a 12.8 ábra helyes megoldására akkor is el tud jutni, ha nem látja más felhasználók igényeit. Csak annyit kell tennie, hogy a valóságot modellezi (külön a kárt, a kocsit, a tulajdonost), nem pedig annak adott nézetét (a kárt és a kocsit egyben). Másrészt a *kollektív* fejlesztésben kerülni kell a párhuzamosságokat. Nem megengedhető például az, hogy öt részleg ötféle partneryilvántartást alakítson ki.

Kiegészítés: Az egyféle modell nem zárja ki a többféle megvalósítást, vagyis adatbázist.

### 6.3.2 Szintkeverés

A legtöbb adatbáziskezelő rendszer strukturálási képességei korlátosak. Ugyanakkor az adatbázis sémájában megkövetelik, hogy együtt adjuk meg a logikai szintű (tartalmi) és a fizikai szintű (ábrázolási, indexelési stb.) jellemzőket.

Ez a kettős tény arra ösztönzi a fejlesztőket, hogy azonnal a korlátos szerkezetekben gondolkodjanak és azonnal a tárolási/formai paramétereknek megfelelően lássák maguk előtt a tényezőket. Még nincs is jól strukturált tervük, de máris kijelentik, hogy például a Számlaszám a Vevőkód+Relatív sorszám együttese lesz 9 numerikus jelben és a tételre indexet fognak húzni. Elfeledkeznek a következő axiómáról:

— *Az adatmodell mindig tisztán fogalmi szintű*

Attól, hogy modellnek nevezzük a jelentést, a tartalmi elrendezést és a megvalósítási formát összekeverő tervünket, az még nem lesz valódi modell. Ha a fenti elvet nem tartjuk be, akkor helytelen szemléletünk két alapvető problémát fog okozni.

Vannak olyan struktúrák (visszamatató kapcsolat, csoport, szerepnév, altípus), amiket a kezelő nem vagy nem megfelelően támogat. Ha ezeket magukat nem rögzítjük sehol sem, hanem azonnal egyféle saját megoldást alkalmazunk, akkor a legkisebb modellváltozás is kín lesz a saját magunk illetve utódaink számára. Mert nem lehet tudni, hogy milyen eredeti problémát oldottunk meg az adott módon.

Az optimumkritériumok sem vizsgálhatók a vegyes logikai/fizikai szintű terven, hiszen az már mesterséges - az eredeti természeteseket elrejtő/eltorzító - tényezőket is tartalmaz. Optimalizálni a szó modellezési értelmében csak fogalmi szintű tervet lehet. Aki tehát lemond a tisztán fogalmi modell megalkotásáról, az nem törődik a terv jóságával sem.

### 6.3.3 Szerepkörtévesztés

A fejlesztéseknek három résztvevője van: vezető, felhasználó és fejlesztő. Nevük azt sejtetné, hogy mindegyik adott szerepkört lát el a fejlesztésben. A vezető elrendeli, majd erőforrásokkal támogatja a fejlesztést, végül ellenőrzi annak eredményét. A felhasználó a maga laikus módján megfogalmazza az igényeit és várja a megfelelő kiszolgálást. Végül a fejlesztő informatikai megoldásra konvertálja az adott igényeket a követelmények szerint. Ezzel az ideális képpel szemben a valóság a következő:

A **vezető** beleszól a modell apró részleteibe is. Ez a kisebbik baj. Nagyobb az, hogy igen sokszor ő maga hozza meg az integrálás elleni döntéseket. Megengedi, hogy a főkönyvelő saját partner-nyilvántartást vezessen, eközben megbíz egy külső céget szintén a partnerek adatainak a kidolgozásával, miközben folyik a szerződésekkel kapcsolatos projekt, amiben természetesen egy harmadikféle partnerismeretsort dolgoznak ki.

A **felhasználó** nem tudja kifejezni az igényeit, mert nincsen tisztában azon fogalmakkal, amelyek saját munkaköre ellátásához nélkülözhetetlenek. Saját maga talál ki olyan új kódokat - osztályozási ismérveket, azonosítókat -, amelyek teljesen ellentmondanak a már létrehozott adatbázis szerkezetének, tartalmának és formájának. Ragaszkodik ahhoz, hogy az állományok struktúrái az ő egyéni szemléletmódját tükrözzék.

A **fejlesztő** nagyon sokszor nem problémát old meg, hanem szoftvert alkalmaz. Nem az köti le a figyelmét, hogy miképpen kellene optimálisan kialakítani egy modellrészletet, hanem az, hogy az eszköz milyen technikai megoldásait fogja bevetni. Emellett kitalál olyan adatokat, állományokat, megoldásokat, amelyek csakis az ő kényelmét szolgálják.

Ezek a szemléleteken változtatni kellene.

— ***A modell a vezető, a felhasználó és a fejlesztő közös produktuma.***

Nem jó, ha a fejlesztő túl hamar utasítja el a felhasználót „az úgy nem lehet” jelszóval illetve ha magától próbál meg kitalálni új tényezőket. Nem szerencsés, ha a felhasználó nem látja be, hogy adott esetben a fejlesztő bizonyos korlátos megoldásokra kényszerül vagy ha - éppen megfordítva - ő kényszerít ki szuboptimális modellrészleteket. Létezik egy célszerű munkamegosztás, de a modell optimalitásáért minden szereplő együtt felelős.

### 6.3.4 Bemenet/kimenet szemüveg

A fenti három problémakör egy közösen is ötvöződni szokott annak következtében, hogy a fejlesztés résztvevői nincsenek tisztában az adatmodellezés szakmai lényegével. Ez főleg abban mutatkozik meg, hogy a felhasználó - de olykor a fejlesztő is - papírokbán gondolkodik, nem pedig valós jelenségekben.

A 6.1 ábra KOCSI egyedének az alkotója *bemenetorientált* tervet készített. Elégették a kocsi nyilvántartási papírját (ami voltaképpen egy inputbizonylat) és ő arról átvette az adatokat függetlenül attól, hogy a tulajdonos és a kocsi két valós lényeg. Az ilyen tervben mindig nagymérvű lesz a fizikai redundancia, mert hiszen egy tulajdonosnak több kocsija is lehet, tehát a tulajdonos ismeretei kocsinként megismétlődnek.

A 6.1 ábra KÁR egyede tipikus *kimenetorientált* tervrészlet. A felhasználó olyan statisztikai kimutatást akart, amely kocsitípus szerint összesíti a károkat. A fejlesztő pedig átvette a nézetét. Elfelejtette, hogy holnap a kapacitás, holnapután pedig a férőhely vagy a gépkocsi színe szerinti kimutatást fognak kérni. Mivel egyensúlytalan szerkezetet alkotott (← *Kiegyensúlyozatlanság*), a strukturális átalakítások elkerülhetetlenek.

— ***A modell szerkezete be- és kimenetfüggetlen kell, hogy legyen.***

A papírokon lévő ismeretek - ritka kivételtől eltekintve - rejtett hierarchikus vagy hálós szerkezeteket alkotnak. A fenti szabály nem azt mondja ki, hogy a modellt a bemeneti és a kimeneti *igényektől* függetlenül kell megalkotni, hiszen az lehetetlen. Hanem azt, hogy a modell *szerkezetének* kell önállóan lennie. A tervezőnek az a feladata, hogy a mondott rejtett összetett szerkezeteket feltárja és azok elemi tényezőiből építse fel a modellt. Ezt csak akkor tudja megtenni, ha modellezőszemüvegét veszi fel, vagyis elsajátítja a szabály által sugallt szemléletet.

## 6.4 Ellenőrző kérdések

6/1 Milyen problémákat fedez fel a következő adatbázisrészletben? A 6.2 pont megfelelő alpontjának a számát adja meg.

VEVŐ (**Vevőkód**, Vevőnév, Vevőcím)  
RENDELÉS (**Rendelésszám**, Rendelésdátum, *Vevőkód*, Vevőcím)

6/2 Az előző módon adja meg az alábbi terv hibáinak a számát. Ilyen több is van.

VEVŐ (**Vevőkód**, Vevőnév, Vevőcím)  
RENDELÉS (**Rendelésszám**, Rendelésdátum, Vevő megnevezés)

6/3 Melyik hibát mutatja az alábbi terv? A nyelv háromszorosan ismétlődő tartalmú.

SZEMÉLY (**Törzsszám**, ..., {Nyelvkód})

6/4 Tartalmaz-e hibát a következő terv? Nem (N), súlyosat (S), kicsit (K).

RENDELÉS (**Rendelésszám**, ..., Dátum)

6/5 Tartalmaz-e hibát a következő terv? Nem (N), súlyosat (S), kicsit (K).

CÉG (**Cég adószám**, ..., Cég név)

SZÁMLA (**Számlaszám**, ..., Számla adószám)

6/6 Ön szerint milyen szemléletben készült az alábbi terv? Helyes (H), bemeneti (B), kimeneti (K)?

SZEMÉLY (**Törzsszám**, ..., Szervezetkód, Szervezetnév, ...)

6/7 Ön szerint milyen szemléletben készült az alábbi terv? Helyes (H), bemeneti (B), kimeneti (K)? A „{ }” jelek közötti adatok ismétlődnek.

RENDELÉSEK (**Rendelésszám**, ..., {Cikktípus, Árfekvés, Rendelt darab})

6/8 Milyen problémát lát a következő tervrészletben? Szóban fejtse ki gondolatait.

CIKK (**Cikkszám**, ..., Ár)

RENDELÉSTÉTEL (**Rendelésszám+Cikkszám**, ..., Ár)

## 7. FÜGGÉSEK ÉS ELŐNORMALIZÁLÁS

### 7.1 Normalizálás

Az adatmodellezés lényege az, hogy feltárjuk és tükrözzük a valós jelenségeket (egyed), azok sajátosságait (tulajdonság) és összefüggéseit (kapcsolat). E művelet során nem csak egy *elfogadható* adatbázis-szerkezetet keresünk, hanem az *optimális* modell kialakítására törekszünk. A jelző itt nem a legjobb számítógépes megoldásra utal. Persze azt is ki kell dolgoznunk, de az a logikai/fizikai szintű adatbázis-tervezés feladata. Fogalmi szinten a modell akkor jó, ha megfelel az ismertetett optimumkritériumoknak.

Eredeti értelmében a normalizálás olyan *matematikai* eljárás, amelynek egyetlen célja a redundanciák kiszűrése a relációs táblákból (7.2 *Relációs adatbázis*). Az egyedek belső struktúráját a tulajdonságai közötti sajátos viszonyok (7.3 *Funkcionális függés*) alapján át lehet alakítani úgy, hogy az átfedések szempontjából kedvezőbb adatmodellt kapjunk. Később rájötték, hogy az egyedek külső szerkezetét is érdemes vizsgálni. A tulajdonságok viszonyait egyedektől függetlenül (7.4 *Tartományfüggés*) elemezve, fel lehet tárni addig rejtett kapcsolatokat. Azaz tehetünk valamit a teljesség követelményének a betartásáért is.

Ahhoz, hogy a tényezők összefüggéseit érdemben vizsgálhassuk, az egyedekből ki kell küszöbölni a többszörös tartalmú ismereteket ( $\leftarrow$  *Ismétlődő csoport*). A tulajdonképpeni normalizálás csak akkor kezdődhet meg, ha a modellben már nincsen ismétlődést mutató tábla (7.5 *Nem-normalizált egyed*). Vagyis legelőször össze kell állítani a normalizálási alapot, azaz előnormalizálást kell végezni.

***Ebben a fejezetben az alapfogalmak tisztázása után az adatmodell gyerekbetegségeit és azok orvoslási módjait - vagyis az alapnormalizálást - fogjuk bemutatni.***

Nem érezzük feladatunknak a relációs adatmodell [8] mélyreható ismertetését. Inkább arra törekszünk, hogy a normalizálást a más szakkönyvekben leírtaknál mélyebben és átfogóbban értelmezve mutassuk be. Ezt azért kell megtennünk, mert az előnormalizálást nem lehet mechanikus módon végrehajtani. Eredetileg tervezett egyedeinkben az ismétlődések nem mindig láthatóan jelentkeznek (7.5 *Többféle ismétlődés*). A mögöttes problémák ezért nagyon sokszor nem oldhatók fel tisztán matematikai eljárásokkal. A tervezőnek be kell vetnie *szemantikai* részmodszereket is. Például a valósághűség és az egyértelműség kritériuma jegyében elemi részekre kell bontania az összetett tulajdonságtípusokat, sőt, olykor a tulajdonságértékeket is.

Minden kezdet nehéz. Az előnormalizálás, a korrekt normalizálási alap megteremtése összetett feladat (7.6 *A normalizálás első lépése*). Azért az, mert ebben a fázisban sokkal nagyobb szerepet kap a tiszta józan gondolkodás, a helyes modellezési szemlélet, mint a sokkal könnyebben elsajátítható matematikai alapú mechanikus módszertár.

### 7.2 Relációs adatbázis

+

Legyenek adottak a  $D_1, D_2 \dots D_N$  egymást nem szükségszerűen kizáró értékhalmazok. Ekkor az  $R = d_1 * d_2 * \dots * d_n$  Descartes-féle szorzat az adott halmazokon képzett reláció úgy, hogy  $d_1 \in D_1, d_2 \in D_2 \dots d_n \in D_N$ . A relációs adatbázis relációk együttese.

Magyarázat: Az értéktartomány - idegen szóval **doméjn** [domain] - bármilyen jelentés nélküli jelsorok értékkészlete lehet. Például egy doméjnt alkothatnak a páratlan számok, egy másikat a cikkszámok és amint látjuk, e kettő értékkészlete nem zárja ki egymást.

TULAJDONOS

Törzsszám	Tulajtípus	Tulajnév	Foglalkozás	Telephely	Felügyelet
111111	Magán	Kovács R.	Ápolónő 11	Budapest	-
222222	Kft	AB Kft.	-	Szeged	-
333333	Magán	Kovács R.	Mérnök 32	Pécs	-
444444	Rt	XY Rt.	-	Budapest	Z

7.1 ábra: A TULAJDONOS reláció

Magyarázat: Póriásabb megfogalmazás szerint a reláció kétdimenziós **tábla**. Vagyis a reláció és a tábla valójában szinonima. (NB.: Sok tervező a táblák közötti viszonyt nevezi - tévesen - relációnak. A reláció [relation] mint tábla nem tévesztendő össze a viszonnal [relationship].) A két dimenzió azt jelenti, hogy a tábla sorokból illetve oszlopokból áll, és azok találkozásánál csak elemi értékek fordulhatnak elő.

Tényezők: Minden reláció tábla, de nem minden tábla reláció. A relációs tábla mindig csak kétdimenziós. Oszlopait (Törzsszám, Típus stb.) **attribútumoknak** nevezzük. Ezek az itt nem mutatott mögöttes tartományokból kapják értékeiket. Tehát az attribútum relációra leképezett doméjn. Az oszlopok homogének és sorrendjük közömbös. (Ennyiben térnek el a táblák a valódi matematikai relációktól). Az oszlopok számát a reláció fokának [degree] hívjuk.

A relációs sor neve a lefordíthatatlan **tuple**. A sorok egymásutánisága is közömbös. A sorok számát a reláció terjedelmének [range] hívjuk. A relációban nem lehet két teljesen azonos sor. Tehát legalább egy attribútumnak az értékében el kell, hogy térjen egymástól minden egyes sor. Magyarul: a relációban kell, hogy legyen azonosító tulajdonság.

Azt a tulajdonságot, amely minden egyes sorra eltérő értéket vesz fel, **kulcsjelöltnek** [candidate key] nevezzük. Ha minden tulajdonosnak más volna a neve - ami persze nem áll fenn -, akkor a fenti táblában két kulcsjelöltünk lenne: a Törzsszám és a Tulajnév. A kulcsjelöltek közül ki kell választani az azonosítóként alkalmazottat. Ezt **elsődleges kulcsnak** [primary key] hívjuk, miközben a többi volt jelölt az **alternáló kulcs** [alternate key] nevet viseli. Természetesen az elsődleges kulcs mint azonosító lehet összetett is.

Ha egy tábla elsődleges kulcsa másik táblában is megjelenik, akkor abban a másikban **idegen kulcsnak** [foreign key] nevezzük. Ez alkalmas az eltérő táblák közötti viszonyok megteremtésére. Például a 6.8 ábra KÁR táblájában is szerepel a Rendszám, ezért ez az idegen kulcs arra való, hogy vele a KÁR/KOCSI tábla felé kapcsolatot hozzunk létre.

Az idegen kulcs ad alapot a relációs adatbázisok egyik fontos szerkezeti korlátjának. A **hivatkozási integritás** [referential integrity] első megközelítésben azt mondja ki, hogy az idegen kulcs (Rendszám a KÁR/KOCSI-ban) csak olyan értéket vehet fel, ami elsődleges kulcsként már létező érték a kapcsolódó fölérendelt táblában (Rendszám a KOCSI-ban). Ez a korlát a beviteli és törlési műveletek feltételeként szolgál akkor, ha a viszony alulról kötelező jellegű. A KÁR/KOCSI-ba csak már létező kocsira való utalást lehet bevinni és ha törlik az adott kocsit, akkor annak minden KÁR/KOCSI alárendeltjét is törölni kell.

Megjegyzés: A hasonlóság ne tévesszen meg senkit sem: az egyedtípus és a tábla nem azonos lényeg. Az egyedtípusokat (fogalmi szint) táblákban valósítjuk meg (logikai szint), ha relációs



adatkezelőt használunk. A reláció matematikai, azaz *menyiségi* fogalom. Semmi köze sincs a *minőségi* „jószág” kérdéséhez. A 6.1 ábrán lévő táblák mindegyike reláció. Egyik sem „jól” tervezett, de ettől függetlenül relációsan kezelhető. Ha tehát valaki relációs adatbázistervet készít, az még messze nem biztos, hogy adatmodellt alkot. A meghatározás semmit sem mond arról, hogy (milyen módon) *szervezett* együttesről van szó. A relációs adatbázis lehet táblák akár teljesen laza, szervezetlen, kusza összessége is.

## 7.3 Funkcionális függés

+

**Az E egyed B tulajdonsága akkor és csak akkor funkcionálisan függ az egyed A tulajdonságától, ha az E minden egyes előfordulásában az A értéke minden időpontban csakis egy B értékkel társul.**

**7.1 példa** KOCSI (Rendszám, Kocsitípus, Szín, Törzsszám, ...)

Magyarázat: Minden kocsinak csak egy típusa lehet. Ezért a Rendszám funkcionálisan meghatározza a Kocsitípust, vagy másképpen az utóbbi funkcionálisan függ az előbbtől. A funkcionális függés [functional dependency] rövid neve **FD**. Jele: „ $\rightarrow$ ”. Feltéve, hogy a kocsikhoz mindig egy színt adunk meg és a kocsinak csak egy tulajdonosa lehet, az alábbi függéseket fedezhetjük fel:

Rendszám  $\rightarrow$  Kocsitípus és Rendszám  $\rightarrow$  Szín és Rendszám  $\rightarrow$  Törzsszám, másképpen  
Rendszám  $\rightarrow$  (Kocsitípus, Szín, Törzsszám).

Az utóbbi jelölést az FD úgynevezett **additivitási szabálya** alapozza meg, ami így szól:

Ha  $A \rightarrow B$  és  $A \rightarrow C$ , akkor  $A \rightarrow B, C$ .

Ez a megkötés a három Armstrong-szabály [9] egyike. A másik kettőről később lesz szó. Ezeken túlmenően az FD-re vonatkozik az ún. **reflexivitási szabály** is, amely szerint  $A \rightarrow A$ , vagyis természetesen minden tulajdonság meghatározza saját magát.

Az FD nem feltételezi azt, hogy a meghatározó tulajdonság értéke egyedi. Tegyük fel, hogy a tulajdonosoknak minden kocsija mindig azonos színű (ami persze nem igaz). Ebben az esetben a KOCSI egyeden belül a Törzsszám is meghatározná a Színt, ámbár a Törzsszám értéke nem egyedi, mert több kocsija lehet valakinek. Ezért alkalmazzuk a fenti definíciónak egy lazább változatát is:

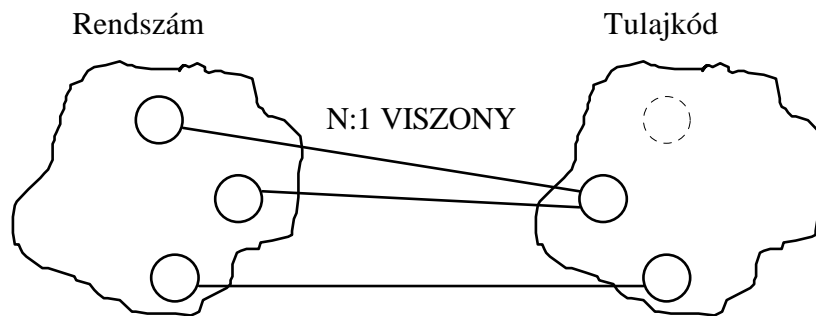
+

**Az E egyedben az A tulajdonság akkor és csak akkor határozza meg funkcionálisan a B-t, ha az utóbbi minden előfordulásban ugyanazt az értéket veszi fel, amikor az A értéke is ugyanaz.**

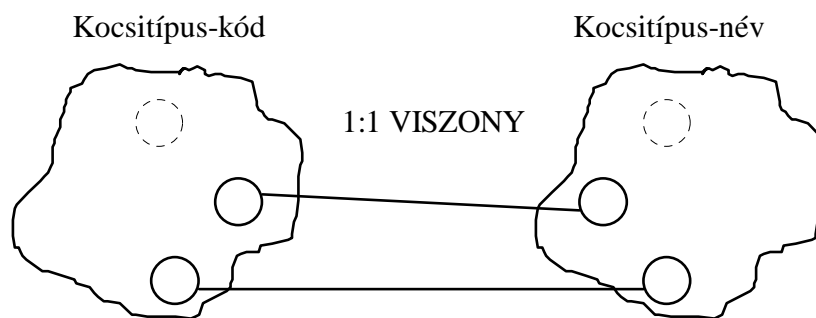
Az FD mindig fordított irányú **függetlenséggel** [independency] jár. Ha  $A \rightarrow B$ , akkor  $B \not\rightarrow A$ . Vagyis ha az A meghatározza a B-t, akkor a B nem határozhatja meg az A-t. Ha fordított irányú függés is fennállna, akkor **kölcsönös függésről** [mutual dependency] lenne szó. Ennek rövid neve **MD**, jele pedig „ $\leftrightarrow$ ”. Ha bevezetnénk a kocsikra az egyedi Kocsitípus név tételt, akkor létezne a Kocsitípus kód  $\leftrightarrow$  Kocsitípus név kölcsönös függés. Végül ha két tulajdonság

között egyik irányban sem tudunk megállapítani függést, akkor *kölcsönös függetlenségről* beszélünk. Ennek jele „<-/->„. Példánkban a Kocsitípus és a Szín között nincs függés, ezért Kocsitípus <-/-> Szín.

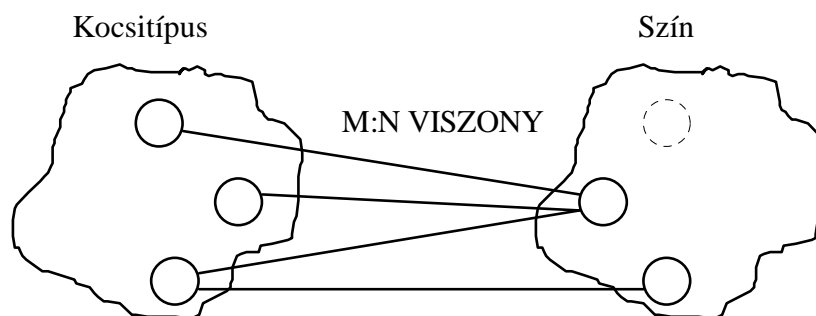
### FUNKCIONÁLIS FÜGGÉS



### KÖLCSÖNÖS FÜGGÉS



### FUNKCIONÁLIS FÜGGETLENSÉG



7.2 ábra: Függési viszonyok

A függés és a függetlenség számszerű - azaz mennyiségi - lényeg. Függés esetében a kapcsolat M:1 fokú hierarchikus viszony, mert több (M) Rendszám értékhez is tartozhat egyazon (1) Kocsitípus érték. Kölsönös függésnél a kapcsolat 1:1 fokú lineáris viszony. Végül kölcsönös függetlenségénél - Kocsitípus/Szín - a viszony M:N fokú hálól. Ezeket a változatokat a 7.2 ábra szemlélteti.

**7.2 példa** RENDELÉSTÉTEL (*Rendelésszám+Cikkszám*, Rendelt mennyiség, ...)  
Rendelésszám+Cikkszám  $\rightarrow$  Rendelt mennyiség

Magyarázat: Az eddigi függések *elemiek* [elementary] voltak, mert a meghatározó egyetlen tagból állt. Azonban gyakori az is, hogy a meghatározó több tényezőből épül fel. Ekkor *összetett* [compound] függésről beszélünk. Ennek jele nálunk  $A+B+\dots+X \rightarrow Y$  lesz, vagyis a meghatározó részeit a „+” jellel fogjuk összekötni. A függések kapcsán - a jelölés miatt - szoktak beszélni a függés bal- és jobboldaláról. A baloldal a meghatározót, a jobboldal a függőt jelenti. Vagyis a függés akkor összetett, ha a baloldal több tagból áll.

**7.3 példa** KOCSI (*Rendszám*, Kocsitípus, ..., CASCO-szám)

Magyarázat: A függéseket a függéserő szerint is osztályozzuk. Az  $A \rightarrow B$  függés csak annyit állít, hogy minden A-értékhez legfeljebb egy B-érték tartozhat. Azt viszont nem köti ki, hogy minden A-értékhez kell is, hogy tartozzon B-érték. Előfordulhatnak olyan esetek, amikor a B-érték nem értelmezhető vagy ismeretlen, mint a fenti példában. Ha minden baloldalhoz (Rendszám) kell is, hogy tartozzon jobboldal (Kocsitípus), akkor a függést *erősnek*, ha nem, akkor *gyengének* nevezzük. A Rendszám  $\rightarrow$  CASCO-szám gyenge függés, mert nem minden kocsinak van Casco biztosítása.

Probléma: Egyidőben egy kocsinak csak egy Cascoja lehet. Ezért egyesek úgy vélik, hogy a Rendszám és a Casco-szám közötti függés kölcsönös. Ez tévedés. A kölcsönös függésnek mindig mind a két oldalról erősnek kell lennie. Mivel pedig nem minden kocsinak van CASCO-ja, ez a feltétel nem áll fenn. Ugyanis a „nincs” Casco-szám értékhez több Rendszám érték tartozik.

**7.4 példa** RENDELÉSTÉTEL (*Rendelésszám+Cikkszám*, Mennyiség, Cikknév)  
Rendelésszám+Cikkszám  $\rightarrow$  Mennyiség  
Rendelésszám+Cikkszám  $\rightarrow$  Cikknév

Magyarázat: A függés akkor *teljes* [full], ha a függő a teljes összetett meghatározótól függ és ha abból bármelyik tagot kiemeljük, a függés megszűnik. A függés akkor *részleges* [partial], ha az összetétel valamelyik tagját elhagyva a függés továbbra is fennáll. Példánk első függése teljes, mert a Mennyiséget sem a Rendelésszám, sem a Cikkszám nem határozza meg önmagában. Ha egyiküket kiemeljük az összetételből, a függés elveszik. A második függés részleges, mert az akkor is fennmarad, ha kiemeljük a meghatározóból a Rendelésszámot. Ugyanis a Cikkszám önállóan is behatárolja a Cikksort.

Kiegészítés: A függés nem más, mint szerkezeti *integritási korlát*. Például azt mondja ki, hogy minden időpontban és minden Cikkszám értékhez csakis egy Cikksort tartalom tartozhat. Ezért figyelni kell az *idődimenzióra*. A függés mindig örökérvényű. Ha tehát egy adott cikkszámú tétel ára úgy változik meg az idők során, hogy kíváncsiak vagyunk továbbra is annak régi értékeire, akkor a Cikkszám  $\rightarrow$  Cikksort függés már nem létezik. (Erről bővebben AZ IDŐ MODELLEZÉSE című fejezetben szólnunk.)

## 7.4 Tartományfüggés

+

A B tulajdonság akkor és csak akkor tartományfüggéssel függ az A tulajdonságtól, ha az A doméjn minden értékéhez minden időpontban a B tartománynak csakis egy értéke társítható.

**7.5 példa** Eladóhely azonosító - *Mi a viszonyuk?* - Központ azonosító

Magyarázat: A tulajdonságokat egyedekhez kötöten (attribútum) és azoktól függetlenül (doméjn) is vizsgálhatjuk (→ Értéktartomány). Ennek megfelelően függéseik elemezhetők egyedeken belül (→ Funkcionális függés) és azokhoz nem kötöten is. Az utóbbi esetben beszélünk tartományfüggésről [domain dependency]. Ennek rövid neve **DD**, jele „=>„. Ha feltételezzük, hogy minden eladóhely minden időpontban csak egy központhoz tartozhat, akkor példánk esetében fennáll az Eladóhely azonosító „=>„ Központ azonosító DD.

**7.6 példa** KÖZPONT (Központ azonosító, ...)  
ELADÓHELY (Eladóhely azonosító, ...)  
RENDELÉS (Rendelésszám, Központ azonosító, Eladóhely azonosító)

Magyarázat: Egy cég tejtermékeket szállít mások részére. A rendeléseket olykor a vevő valamelyik eladóhelye adja fel, ekkor a cég közvetlenül szállítja ki az árut pl. a közértbe. Máskor maga a központ a rendelő, ekkor ő viszi el az árut saját raktáraiba, elosztóiba és ő gondoskodik a termékek terítéséről.

Probléma: Mivel egy eladóhely csak egy központhoz tartozhat, a harmadik egyedben az Eladóhely azonosító meghatározni látszik a Központ azonosítót. Ám a látszat csal, mert ha a központ adja fel a rendelést, akkor az Eladóhely azonosítónak az értéke értelmetlen (← Üres érték). Funkcionális függés akkor áll fenn, ha az A meghatározó minden egyes értékéhez a B függőnek mindig csak egy értéke tartozik. Maga az üres érték is egy adott értéknek tekintendő. Az pedig nem igaz, hogy valahányszor üres az eladóhely kulcsa, akkor mindig ugyanaz a központ kulcsértéke. Tehát nincs függés a két tétel között.

Viszont éppen emiatt a 7.6 példa megoldása nem teljes. Sehonnan sem tudhatjuk meg belőle, hogy mely központokhoz tartoznak az eladóhelyek. A két egyed nem kapcsolható egymáshoz (← Kapcsolathiány). Ezen a problémán segíthet a tartományfüggés.

Megoldás: A funkcionális függéseket egyedeken belül értelmezzük, ezért esetükben **egyeden belüli** [intra-entity] függésről beszélünk. Ha az ilyen viszonyban álló két tétel két egyednek a kulcsa, akkor a függést **egyedek közötti** [inter-entity] függésnek is tekintjük. A Rendelésszám → Központ azonosító kötődés nemcsak a RENDELÉS egyeden belül, hanem a RENDELÉS és KÖZPONT egyedek között is értelmezhető. Azonban az utóbbi esetben a viszony nem tekinthető FD-nek. Éppen ezért van szükség a DD-re. Egyedtől függetlenül, pusztán a tulajdonságok viszonyait vizsgálva felismerhetjük az Eladóhely azonosító => Központ azonosító tartományfüggést.

**7.7 példa** KÖZPONT (Központ azonosító, ...)  
ELADÓHELY (Eladóhely azonosító, Központ azonosító, ...)  
RENDELÉS (Rendelésszám, Központ azonosító, Eladóhely azonosító)

Magyarázat: A feltárt DD alapján az ELADÓHELY egyedet kiegészítjük a kulcsától függő Központ azonosítóval, ami után megszűnik a kapcsolati probléma.

Jellemzők: A doméjnfüggést ugyanazok a sajátosságok jellemzik, mint a funkcionálisat. Tartományok között is létezhet függetlenség illetve kölcsönös függés. A DD is lehet elemi vagy összetett, erős vagy gyenge. Azonban a DD egy fontos vonásban eltérhet az FD-től. Az utóbbiban a függéserőt mindig csak a „jobboldal” szempontjából vizsgáljuk. Tehát azt nézzük, hogy az  $A \rightarrow B$  függésben az A meghatározó minden értékéhez kell-e tartoznia B értéknek (erős) vagy sem (gyenge függés). Mivel egy egyeden belül a kulcs értéke sohasem lehet üres, a függéserő a másik irányban mindig értelemszerűen erős.

### 7.8 példa CSOPORT (Csoportkód, Kedvezmény, ...) KOCSI (Rendszám, Csoportkód, ...)

Magyarázat: A kocsikra csoportos biztosítást is köthetnek. Egy koci csak egy csoportba tartozhat, de nem biztos, hogy van csoportos biztosítása. Ezért a Rendszám  $\rightarrow$  Csoportkód FD jobboldaltól gyenge. Látszólag ugyanez a helyzet a Rendszám „ $\Rightarrow$ ”, Csoportkód tartományfüggés esetében is. Mivel azonban ezt a függést már nem egyedtípuson belül vizsgáljuk, a viszony ereje a fordított irányban is értékelhető. Felvethető az a kérdés is, hogy kell-e minden Csoportkód értékhez Rendszám értéknek tartoznia, vagy sem? Ha csak olyan csoportokat tartunk nyilván, amelyekhez valóban tartoznak kocsik is, akkor a DD „baloldaltól” nézve is erős, egyébként azirányból is gyenge. Semmi akadálya sincs annak, hogy előre meghatározzunk egy olyan csoportos biztosítási kedvezményt, amelyet eddig még senki nem vett igénybe (mert hiszen éppen most határoztuk meg).

Kiegészítés: A tervezőnek meg kell határoznia az egyes egyedek tulajdonságsorát és ki kell jelölnie az egyedek kapcsolatait ( $\leftarrow$  *Egyedszerkezet*). Ez a két feladat nem független egymástól. A funkcionális függések majd abban segítenek bennünket, hogy optimálisan tudjuk meghatározni az egyedek belső szerkezeteit. A tartományfüggések pedig ahhoz nélkülözhetetlenek, hogy hibamentesen alakíthassuk ki az egyedek külső struktúráit. Az FD egyeden belüli, a DD egyedek közötti korlátként szolgál.

## 7.5 Nem-normalizált egyed

+

Az egyed ismétlődő ismeretet tartalmaz, ha van olyan tulajdonsága, amely funkcionálisan független az azonosítójától. Ilyenkor az egyed nem-normalizált, másképpen szólva 0NF alakú.

SZEMÉLY

Törzsszám		Név	Nyelv	Vizsgaév	Vszint
AAA	...	Kovács L.	Angol	1972	felső
			Francia	1978	közép
			Olasz	1981	felső
BBB	...	Szabó P.	Olasz	1976	felső

7.3 ábra: A SZEMÉLY egyed részlete

Magyarázat: Az ábra első három sora egy egyedelőfordulást alkot. Az egyednek vannak olyan adatai, amelyek egy név alatt több értéket vesznek fel ( $\leftarrow$  Ismétlődő csoport). Mivel egy Törzsszám értékhez több Nyelv érték is tartozhat, nem áll fenn a Törzsszám  $\rightarrow$  Nyelv függés, tehát az utóbbi adat az előbbtől független. Ezt a tételt meg is fordíthatjuk: Ha az egyedben van olyan tulajdonság, amely nem függ a kulcstól, akkor az egyed ismétlődő ismeretet tartalmaz (lásd a meghatározást).

Probléma: Nagy általánosságban már kitértünk az ismétlődés által okozott gondokra ( $\leftarrow$  Ki-egyensúlyozatlanság). Most konkrétan is megfogalmazzuk azokat.

- Meg kell adni a maximális ismétlésszámot. Ha ez a nyelv esetében három, akkor a negyedik nyelv kimutatására nincs mód egyetlen egy személynél sem.
- Ha valaki a maximálisnál kevesebb nyelvet beszél, akkor feleslegesen foglaljuk a tárolót. (N.B.: Az üres érték is tárt igényel és kezelési időtöbblettel jár.)
- Ha a nyelvek fix helyhez rendelve, például mindig az angol az első nyelv, akkor nemcsak személyenként, hanem összesen is csak három nyelvet lehet kezelni.
- Ha viszont a nyelvek nem fix helyhez rendelve, akkor nehézkes például az olaszt beszélők kikeresése.
- A kezelés egyensúlytalan. Hamar megtaláljuk, hogy ki milyen nyelveket beszél, de nehezen leljük meg az adott nyelvet beszélő személyeket.

Megoldás: A nem-normalizált, ONF alakú egyedet néha N1NF [not first normal form] alakúnak is hívják. Ezzel a megjelöléssel utalnak a legelső feladatra: az egyedet legalább első normálformájúra (1NF) kell átalakítani. Mivel pedig a problémákat az ismétlődések, az azonosítótól független tulajdonságok okozzák, azokat kell eltávolítani az egyed belső szerkezetéből, vagyis tulajdonságsorából. A SZEMÉLY egyedben a Nyelv, a Vizsgaév és a Vizsgadatum többértékű, tehát másutt van a helyük.

SZEMÉLY

Törzsszám		Név
AAA	...	Kovács L.
BBB	...	Szabó P.

NYELVTUDÁS

Törzsszám	Nyelv	Vizsgaév	Vszint
AAA	Angol	1972	felső
AAA	Francia	1978	közép
AAA	Olasz	1981	felső
BBB	Olasz	1976	felső

7.4 ábra: Az ismétlődő tulajdonságok kiküszöbölése

Magyarázat: „Normális” esetben az ismétlődés megszüntetése úgy történik, hogy az alap-egyedből kiemeljük a többszörös értékű tulajdonságokat (Nyelv, Vizsgaév, Vszint) és ezt a sort kiegészítjük az alapegyed kulcsával (Törzsszám), hogy felé kapcsolatot tudjunk teremteni. Az új egyed kulcsa mindig összetett (Törzsszám+Nyelv).

SZEMÉLY

Törzsszám		Név
AAA	...	Kovács L.
BBB	...	Szabó P.

NYELV

Nyelvkód	Nyelvnév
01	Angol
04	Francia
07	Olasz

#### NYELVTUDÁS

Törzsszám	Nyelvkód	Vizsgaév	Vszint
AAA	01	1972	felső
AAA	04	1978	közép
AAA	07	1981	felső
BBB	07	1976	felső

7.5 ábra: Az új szerkezet kiegészítése

Magyarázat: A kódmaniás tervezők új egyedet (NYELV) és azonosítót (Nyelvkód) is bevezetnek. (Ezt akkor is megteszik, ha erre semmi szükség nincs, mert az eredeti kulcs maga is egyértelmű és nem változékony. Ám ezt a kérdést nem akarjuk most feszegetni.) Az olvasó meggyőződhet arról, hogy a 7.4 és 7.5 ábrák ugyanazt a lényegyet tükrözik, és mindkét megoldás elkerüli a fentebbi listában felsorolt problémákat.

#### KÁR

Kárszám	Dátum	Típus	Tulajkód	Kárösszeg
23000	99.05.14	Lada	111111	X
		BMW	111111	Y
34000	99.06.06	Lada	333333	Z
45000	99.06.30	Lada	111111	Q

7.6 ábra: Ismétlődést tartalmazó KÁR egyedtípus

Magyarázat: Az egyedben lévő ismétlődést a fentebb javasolt módon megszüntetjük.

#### KÁR

Kárszám	Dátum
23000	99.05.14
34000	99.06.06
45000	99.06.30

#### EGYEDI KÁR

Kárszám	Tulajkód	Típus	Kárösszeg
23000	111111	Lada	X
23000	111111	BMW	Y
34000	333333	Lada	Z
45000	111111	Lada	Q

7.7 ábra: A KÁR egyedtípus megbontása

Probléma: A megbontás után nem találunk megfelelő azonosítót az EGYEDI KÁR-nak. A tulajdonos (111111) egyik kocsijával nekitolatott a másiknak. Ez egy esemény (23000). Viszont a feltételezett kulcspáros értéke éppen emiatt nem egyedi. Az olvasó téved, ha ezt a gondot azáltal véli feloldhatónak, hogy a kulcsba beleérti a Típust vagy annak azonos volta esetén még a Kárösszeget is. (Ad abszurdum két azonos típusú kocsim ugyanakkora károkat is szenvedhetett. Tehát ez a kulcs elfogadhatatlan. Nem beszélve arról, hogy egy kulcsnak sohasem lehet eleme egy bármikor módosítható értékű összeg-jellegű adat.)

Megoldás: Nem az egyik tulajdonos, kocsitípus, kárösszeg ment a másiknak, hanem a kocsik ütköztek. Az EGYEDI KÁR egyed nem a típus, sőt nem is a tulajdonos kárát adja meg, hanem a (kérdéses tulajdonoshoz tartozó, de ez most mellékes) kocsi kárát. Ezért az egyetlen helyes megoldást nem találhatjuk meg *matematikai* alapokon.

#### EGYEDI KÁR

Kárszám	Rendszám	Típus	Tulajkod	Kárösszeg
23000	ABC 134	Lada	111111	X
23000	BCD 265	BMW	111111	Y
34000	DEF 896	Lada	333333	Z
45000	ABC 134	Lada	111111	Q

7.8 ábra: A kiegyensúlyozott modellrészlet

Magyarázat: Erre az eredményre nem a funkcionális függések - számszerű viszonyok - alapján jutottunk. A megoldás *szemantikai*: az EGYEDI KÁR fogalom lényegét kellett megtalálnunk. Miután kiderült, hogy az a kocsikra vonatkozik, már könnyű volt a KÁR egyedben addig nem is szereplő Rendszám felvételével megtalálni a helyes azonosítót. (A megoldás az ábrában kiemelt redundancia miatt még nem végleges, de egyelőre nem eme fizikai átfedéssel kellett foglalkoznunk. Azt majd a következő fejezetben tesszük meg.)

Kiegészítés: Azt, hogy egy egyed nem-normalizált (nem optimális), matematikai, azaz *menynyi-ségi* alapon is fel lehet tárni (vö. ismétlődések). Maga a megoldás sokszor nem matematikai, hanem szemantikai, azaz értelmezési, *minőségi* jellegű. Az értelmezés néha odáig vezethet, hogy teljesen átalakítjuk az eredeti elképzelést. Lásd a következő pontot.

## 7.6 Többféle ismétlődés

#### SZEMÉLY

Törzsszám	Név	Nyelv-1	Nyelv-2	Nyelv-3
AAA	Kovács L.	Angol	Francia	Olasz
BBB	Szabó P.	Olasz	-	-
CCC	Molnár E.	Spanyol	-	-

7.9 ábra: Nevek mögé rejtett ismétlés

Magyarázat: A mai kezelőrendszerek többsége nem engedi meg, hogy egy név alatt több tartalmat tároljunk, tehát eleve kizárják az ismétlődéseket. Ez abból fakad, hogy a relációt mindig kétdimenziós táblaként kell felfogni (← *Relációs adatbázis*). Ezt a korlátot egyes „ravasz” tervezők úgy oldják fel, hogy az ismétléseket nevek mögé rejtik, mint ábránkon.

Probléma: Az olvasó meggyőződhet arról, hogy az előző pontban felsorolt nehézségek egyike sem oldódik meg a fenti „trükkkel”. A 7.9 ábra egyede továbbra sem normalizált, azt tehát az előbbi pontban leírt módon meg kell bontani.

#### KIKÜLDETÉS-1

Kiküldetés	Szállás	Étkezés	Dologi
AAA	X Ft	A Ft	L Ft
BBB	Y Ft	B Ft	M Ft
CCC	Z Ft	C Ft	N Ft



# KIKÜLDETÉS-2

Kiküldetés	Ktsgtípus	Dologi
AAA	Szállás	X Ft
AAA	Étkezés	A Ft
AAA	Dologi	L Ft
BBB	Szállás	Y Ft
...		
CCC	Dologi	N Ft

7.10 ábra: Az „utazó ügynök” dilemmája

Magyarázat: Az egyed- és tulajdonságtípusok absztrakciók. Ezért igen gyakran egymást vált-hatják ki. Például ha külön személy és szervezet egyedhalmazokat képzelünk el, akkor van két olyan egyedtípusunk, amelyek egyikében sem kell utalni a partner jellegére. Ha viszont egyetlen partner egyedhalmazban gondolkodunk, akkor a partner egyedben fel kell venni azt a tulajdon-ságtípust, amely utal a partner személy/szervezet jellegére. Ez a kettősség másféle összefüggésben is jelentkezhethet.

A fenti példa első változatában a különböző nemű költségek tulajdonságtípusokként, míg a második verzióban egyedelőfordulásokként kerültek tükrözésre. Az olvasó úgy érzi, hogy ebben az esetben is valamiféle ismétlődésről van szó, de nincs biztos támpontja arra nézve, hogy a két mutatott megoldás közül melyiket fogadja el. Ebben kívánjuk segíteni.

Mérlegelés: A második verzióban minden *strukturális változtatás* nélkül új költségtípus vehető fel, mert az tulajdonságértékként, nem pedig új tulajdonságtípusként jelentkezik, mint az első változatban tenné. Például ha új költségfajta lesz a Reprezentáció, akkor az első esetben az egyedet ki kell egészíteni új tulajdonságtípussal (szerkezeti változtatás). A második esetben csak a megengedett értékek készletét kell bővíteni (korlátmódosítás) és máris bevihetők a repre-zentációs költségek. Régebben a szerző inkább a második verzióra voksolt, ha a költségtípusok köre bővíthetett. Ma már összetettebben fogja fel a dolgokat.

Kifejtés: Az első, második stb. nyelv csak *mennyiségi* kérdés. Bármikor felvehetjük a negyedik, ötödik stb. nyelvet, az a *minőségi* lényegen nem változtat. Minden ilyen esetben a nyílt vagy a rejtett ismétlésnek a normalizálással való feloldása elfogadható megoldás. Azonban a szállás, étkezési stb. költségek minőségileg nem azonos lényegeket. Például nem költethető el az egyik keret a másik célra illetve nem mindegyik kiküldetésben kapja meg az utazó mindegyik keretet. A lényegét úgy is megfogalmazhatjuk, hogy valójában szó sincs valós vagy rejtett ismétlésről! Ezért nem is alkalmazható a normalizálás megoldása.

Probléma: Nem hivatkozunk azokra a forrásokra, amelyek az itt felvetett problémakört boncolgatják. Azt viszont megemlíthetjük, hogy egyetlen egy sincs közöttük, amely keresné a mate-matikai-mennyiségi jellegű felvetés mögött rejtőző szemantikai-minőségi természetű alapokat. Ugyanis a valódi és mindeddig rejtett kérdés az, hogy hol tároljuk mindazokat az adatokat, amelyek magát a kiküldetést jellemzik?

Az első változatban mindezek magát a KIKÜLDETÉS-1 egyedet jellemzik. Viszont a második verzióban külön KIKÜLDETÉS egyedet kell létrehozni, mert a KIKÜLDETÉS-2 egyedben nem lehet minden költségtípus mellett megismételni például a kiküldetés helyét. Ezért a fenti mérle-gelés félrevezető. A második megoldásnál be kell vezetni egy külön értéktartománnyal kont-rollálandó új tulajdonságot (Ktsgtípus); egy helyett két egyedre van szükség (kiküldetés és annak

költségei); és a kettő között kapcsolatot kell létrehozni. Mivel az egyednek új tulajdonsággal való bővítése (1. verzió) csak ritkán előforduló és akkor is igen könnyen végrehajtható átszerkesztési művelet, egyáltalán nem biztos, hogy a két egyed közötti kezelést minden alkalommal igénylő megoldás (2. verzió) előnyösebb.

#### 7.9 példa SZEMÉLY (Személy azonosító, ..., Szakmakód)

Magyarázat: A 7.9 ábra a nevek mögé rejtett ismétlésről szól, itt pedig az értékek mögé rejtettekről fogunk beszélni. A tervezők gyakran alkalmaznak ún. *értékkombinálást*. (Ez többnyire a helytelen ismeretfeltárás következménye.) A tervező azt feltételezte, hogy egy személynek egy időben egy csak szakmája lehet. A Szakmakód X értékével akarta jelölni a mérnököt, az Y értékkel pedig a közgazdát. Időközben kiderült, hogy egyes személyek egynél több szakmát is elsajátítottak.

A nagyon jó tervező ilyen tévedést nem követ el. A jó elismeri volt hibáját és azt úgy küszöböli ki, hogy a szerkezetet átalakítja. Azaz létrehoz egy olyan SZEMÉLY/SZAKMA egyedet, amely a szakma ismétlődő ismeretet normalizálja. Lásd a 7.10 példát. Viszont a rossz tervező fel fog venni egy új Z kódértéket azokra a személyekre, akik mérnökök is és egyben közgazdák is.

#### 7.10 példa SZEMÉLY (Személy azonosító, ...) SZEMÉLY/SZAKMA (Személy azonosító+Szakmakód)

Magyarázat: Felesleges gyözködnünk az olvasót arról, hogy ez a teljesen egyértelmű és tiszta szerkezet mennyivel jobb megoldás, mint ha az értékek áttekinthetetlen útvesszőire kell építenünk az ismeretkezelést.

— ***Sohasem szabad az adatszerkezetet úgy egyszerűsíteni, hogy a programok bonyolultsága növekedjen.***

Magyarázat: Az adatstruktúra mindig egyszeres. A kezelőprogramok többszörösek. Ha a struktúra nem „tisza”, azaz rejtett megoldásokra épül, akkor a legelső változáskor fel fog borulni a rendszer. A 7.10 példa szerint mindig azonos logika alapján kereshetők ki a személyek szakmái illetve az adott szakmákat művelő személyek. A 7.9 példában nem kizárt, hogy valaki felveszi az új Q = közgazda+ügyvéd kódértéket. Vajon a már meglévő több tucat program közül hányat fognak átírni ennek megfelelően?

## 7.7 A normalizálás első lépése

#### 7.11 példa SZEMÉLY EREDETI (Személy azonosító, ..., {Nyelv, Vizsgaév,Vszint}) SZEMÉLY (Személy azonosító, ...) NYELVTUDÁS (Személy azonosító+Nyelvkód, Vizsgaév, Vszint) NYELV (Nyelvkód, Nyelvnév)

Magyarázat: A tervező első feladata az ismétlődések megszüntetése, a 0NF normálalakú (← *Nem-normalizált egyed*) relációk megbontása a funkcionális függetlenségek mentén. Az eredeti egyedből így kettő olyan születik (SZEMÉLY és NYELVTUDÁS), amelyek az eredeti

kulcsán (Személy azonosító) át egymáshoz kapcsolhatók. A megbontással olyan hierarchikus szerkezet jön létre (egy személynek több nyelvtudása van), amit - ha nem lett volna addig jelen - egy mesterséges kulcs bevezetésével hálóssá is tehetünk (egy nyelvet több személy beszélhet). A személyek és a nyelvek M:N fokú viszonyban állnak úgy, hogy az összefüggést a harmadik nyelvtudás reláció testesíti meg.

Kiegészítés: Egy eredeti relációban nem csak egyféle ismétlődés fordulhat elő. Például a személyhez a többféle nyelvismeret mellett kapcsolódhat többértékű bér adatcsoport is. Ekkor a relációt több szempont szerint is meg kell bontani és a NYELVTUDÁS mellett létre kell hozni a BÉREI relációt is. Az sem kizárt, hogy a leválasztás után nyert reláció további ismétlődést tartalmaz. Például a bérek jogcímei szerint ismétlődhetnek. Ezért a megbontást addig kell folytatni, ameddig ismétlésmentes relációkat nem kapunk.

**7.12 példa** SZEMÉLY EREDETI (Személy azonosító, ..., {Előző munkahelyek})  
 SZEMÉLY (Személy azonosító, ...)  
 ELŐZŐ MUNKAHELY (Személy azonosító+Sorszám, Előző munkahely)

Magyarázat: A megbontás nem mindig eredményez hálós struktúrát, mint a 7.11 példa esetében. Lehet, hogy megmarad a hierarchia szintjén. Kíváncsiak vagyunk arra, hogy a személyeknek milyen volt a szakmai előélete. Vezetni akarjuk az előző munkahelyeikkel kapcsolatos ismeretek némelyikét (egy személy - több munkahely hierarchia), de nincs sem szükségünk, sem lehetőségünk arra, hogy a magukra a cégekre vonatkozó adatokat is nyilvántartsuk. Tehát számunkra maga a volt munkahely nem ismerettel leírandó dolog, vagyis nem külön egyedként tükrözendő lényeg. Nem akarjuk és nem is tudjuk kikeresni azt, hogy kik dolgoztak ugyanazon a korábbi helyen (egy munkahely - több személy). A szerkezet megmarad „féloldalas” hierarchiának, nem formáljuk át azt hálóvá.

Kiegészítés: A normalizálást taglaló szakirodalom *matematikai* szemléletű. ONF alakú relációk átalakításáról szól, holott az ismétlődéses - többdimenziós - tábla valójában nem is reláció. Soha se feledjük, hogy a modellezés nem relációkra irányul. Az előző pontban már meggyőződhattünk arról, hogy bizonyos problémák csakis *szemantikai* módszerekkel számolhatók fel. A normalizálás hasznos segéd eljárás. Ámde az adatmodellezésnek nem a relációs táblák, hanem a valós jelenségek a tárgyai.

## 7.8 Ellenőrző kérdések

7/1 Az irányítószámmal azonosított helységek megyékben helyezkednek el. Minden megyében lehet több település és minden település csak egy megyében van. (Most tekintsünk el Budapesttől.) Az alábbi képletek közül melyik helyes?

1. Irányítószám  $\rightarrow$  Megyekód
2. Irányítószám  $\leftrightarrow$  Megyekód
3. Irányítószám  $\rightarrow$  Megyekód
4. Irányítószám  $\leftarrow$  Megyekód

- 7/2 Milyen jellegű a TELEPÜLÉS egyeden belüli Irányítószám → Településnév függés? Kétoldalról gyenge (G), kétoldalról erős (E), csak baloldalról erős (B), csak jobboldalról erős (J). Ne feledje, hogy létezik postafiók irányítószám is!
- 7/3 Adott a Gyerekkód és az Apakód tulajdonság. Adja meg a két tétel közötti viszony jelét akkor, ha a gyerek természetes apjáról van szó (A) és akkor, ha válás vagy egyéb ok miatt a kölknek több papája is lehet (B).
- 7/4 Adja meg a függési viszonyok jelét a következő tulajdonságpárosokra:
- Raktárkód/Cikk-kód - adott cikk több raktárban is lehet
  - Raktárkód/Cikk-kód - adott cikk mindig csak egy raktárban tárolódik
  - Rendelészám/Cikk-kód - a rendelés egytétel
  - Rendelészám/Cikk-kód - a rendelés többtétel
  - Irányítószám/Kerületkód - Budapesten
  - Féjrkód/Feleségkód - csak az aktuális házasság érdekel bennünket
  - Féjrkód/Feleségkód - minden volt és létező házasság érdekel bennünket
- 7/5 Az Irányítószám azonosítja a településeket. Milyen jellegű az Irányítószám => Településnév doméjnfüggés? Mindkét oldalról gyenge (G), erős (E), csak jobbról erős (J), csak baloldalról (B) az.
- 7/6 Adott a RENDELÉS (**Rendelészám**, ..., Vevőkód) egyed. Milyen természetű a Rendelészám és a Vevőkód függése? A helyes válasz számát kérjük.
- Egyeden belüli.
  - Egyedek közti.
  - Mindkettő.
- 7/7 Rendeléseink általában egytétel

SZEMÉLY (**Törzsszám**, ..., Páros havi számla, Páratlan havi számla)

- 7/9 Adja meg az előző példa helyes megoldását!

## 8. ALAPVETŐ NORMÁLFORMÁK

### 8.1 A normalizálás alapjai

Az adatmodellezés egyik fő célja az optimalizálás, tehát a modellt alkotó egyedek lehető legjobb *belső és külső szerkezetének* a megkeresése. A korábbiakban rámutattunk arra, hogy a két dolog kölcsönösen összefügg ( $\leftarrow$  *Egyedszerkezet*). Az egyed tulajdonságainak a sora a kapcsoló szerepű tulajdonságokon át meghatározza az egyed kapcsolattípusainak az együttesét és fordítva: a tervezett kapcsolatok befolyásolják azt, hogy az egyednek milyen tulajdonságai lesznek.

Az egyedek optimális tulajdonságsorának a kialakításában a normalizálás matematikai eljárása segít bennünket. Az egyedek lehető legjobb normálformáját (NF) több lépésben alakítjuk ki. Ebben a részben az eredeti normálalakok közül a legegyszerűbbeket mutatjuk be; most csak az 1NF, 2NF és 3NF lényegéről lesz szó (8.2 *Második normálforma* és 8.3 *Harmadik normálforma*). A 2NF jobb, mint az 1NF, de rosszabb, mint a 3NF. Minden magasabb alakkal egy-egy újabb problémát oldunk meg. Ezért a normálalakok egymásba skatulyáztak. Azaz ha egy egyed 3NF formájú, akkor már szükségszerűen 2NF alakú is.

*Ebben a fejezetben az egyedek alapvető hibáit, azok elméleti szerkezeti okait, a hibák kiküszöbölésére alkalmas normálalakokat és az ezek elérésére szolgáló normalizálási lépéseket tárjuk fel azok eredményeivel együtt.*

Bár a tapasztalt tervező nem lépésről-lépésre normalizál, egyszer be kell mutatnunk a teendőket elemi szinten is (8.4 *A normalizálás második lépése*). A szemléltetés kedvéért végig fogjuk vezetni a KOCSI mintaeset normalizálási megoldását (8.5 *Egy teljes példa*). Jóllehet a most tárgyalt normálforma *dekompozíció* eredetileg csak az egyedekre irányult, rá fogunk mutatni, hogy kihat a kapcsolatokra is (8.7 *Normalizálás és modellstruktúra*).

A matematikai eljárás többféle módon végezhető el. Sem a leendő munka volumene (8.6 *A normalizálás természete*), sem annak végső eredménye (8.8 *A dekompozíció sajátosságai*) szempontjából nem közömbös, hogy melyik megoldást választjuk. Végül azt is meg kell említenünk, hogy sajnos a szakirodalom egyes sajátos esetekben nem segíti a tervezőket, hanem rossz megoldásokon keresztül éppen hogy félreinformálja, mint például a kulcsváltozatok esetében is (8.9 *Alternáló kulcs*).

### 8.2 Második normálforma

+

Az egyed akkor és csak akkor van legalább első normálformában (1NF), ha minden nem-kulcs tulajdonságtípusa funkcionálisan függ az azonosítótól.

## KÁR

Kárszám	Dátum
23000	99.05.14
34000	99.06.06
45000	99.06.30

## KÁR/KOCSI

Kárszám	Rendszám	Típus	Tulajkod	Kárösszeg
23000	ABC 134	Lada	111111	X
23000	BCD 265	BMW	222222	Y
34000	DEF 896	Lada	333333	Z
45000	ABC 134	Lada	111111	Q

8.1 ábra: Egyed típusok 1NF alakban

Magyarázat: A két egyed egyike sem tartalmaz ismétlődő tulajdonságot, mint azt tette a megbontás előtt (7.6 ábra - *← Nem-normalizált egyed*). Azaz tulajdonságaik mindegyikét meghatározzák a kulcsok. Bennünket most csak a második tábla fog érdekelni, ami a definíció szerint legalább 1NF alakú. A „legalább” azt sejteti, hogy a megoldás még nem teljesen tökéletes.

Probléma: A kiemelt sorok *fizikai redundanciát* mutatnak. Ha egy kocsit ötször szenved kárt, akkor ötször fogják rögzíteni a típusát, ami felesleges.

+

**Az E egyed típus C tulajdonsága akkor és csak akkor függ részlegesen az A+B összetett azonosítótól, ha a C-t az A vagy a B is meghatározza.**

## RENDELÉSTÉTEL

Rendelészám	Cikkszám	Cikknév	Mennyiség
23000	XXX	A-csavar	X
23000	YYY	alátét	Y
34000	XXX	A-csavar	Z
45000	ZZZ	karmantyú	Q

8.2 ábra: Részleges függést tartalmazó RENDELÉSTÉTEL egyed

Magyarázat: A függést akkor neveztük részlegesnek, ha az összetett meghatározó egyik részét eltávolítva a függés továbbra is fennáll. A Rendelészám+Cikkszám → Cikknév függésből ki-vehetjük az első tagot, a Cikkszám → Cikknév függés ettől még megmarad.

Probléma: A normalizálási gondokat *anomáliáknak* (visszásságoknak) nevezzük. Ezek különösen akkor jelentősek, ha a problémát okozó tételek más egyedekben nem találhatók együtt. Ilyen helyzetben csak a *nyílt logikai átfedés* bajaival kell számolni. Az összes lehetséges gond szemléltetése kedvéért tételezzük fel, hogy nem létezik még olyan CIKK egyed, amiben együtt szerepel a Cikkszám és a Cikknév. Ekkor a bajok a következők:

- Tárolási anomália: A Cikknév fizikai redundanciát mutat és így pazarolja a tárat. Például az adatbázis többször tartalmazza az „XXX - A-csavar” párost.

- Frissítési anomália: A Cikknév változásakor többszörös módosításra van szükség például akkor, ha az „A-csavar” neve „B-csavar”-ra változik.
- Törlési anomália: Ha töröljük az adott cikkekre vonatkozó utolsó rendelést, akkor elveszik a cikknév ismeret is. Pl. az „45000” rendelés törlésekor megszűnik a „ZZZ - karmantyú” páros, ha csak ebben a rendelésben kértek ilyenféle cikket.
- Beviteli anomália: A Cikknév ismeretét nem tudjuk tárolni az olyan cikkekre, amelyekre még nem vonatkozik rendelés. Ha nem kértek csapágyat, akkor nem vihetjük be a „QQQ - csapágy” ismeretpárost.

Az utolsó kitélt kicsit bővebben is megmagyarázzuk, felhívva a figyelmet a tervezők egyik tipikus rossz szokására. A RENDELÉSTÉTEL kulcsa összetett. Az azonosítókra vonatkozó szabályok szerint a kulcs és annak részei nem lehetnek üres vagy ismeretlen értékűek. Ha tehát nem ismert a Rendelésszám, akkor nem vihetjük be önmagában a Cikkszám és a Cikknév páros tartalmát az egyedbe egy új cikkekre vonatkozóan. Ezen a gondon nem szabad úgy segíteni, hogy kiadnak egy ideiglenes azonosítóértéket (például felveszik a „99901 - QQQ - csapágy - 0” sort). Az adatstruktúra nem redukálható a programbonyolultság rovására (lásd az idevágó szabályt - ← Többféle ismétlődés). Az egyetlen helyes megoldás a RENDELÉSTÉTEL egyed normalizálása.

### 8.3 A normalizálás második lépése

RENDELÉSTÉTEL

<i>Rendelésszám</i>	<i>Cikkszám</i>	Mennyiség
<b>23000</b>	<b>XXX</b>	X
<b>23000</b>	<b>YYY</b>	Y
<b>34000</b>	<b>XXX</b>	Z
<b>45000</b>	<b>ZZZ</b>	Q

CIKK

<i>Cikkszám</i>	<i>Cikknév</i>
<b>XXX</b>	A-csavar
<b>YYY</b>	alátét
<b>ZZZ</b>	karmantyú

8.3 ábra: A RENDELÉSTÉTEL normalizálása

Definíció: *Normálforma dekompozíciónak* nevezzük azt az eljárást, amelynek során az eredeti egyedtypust a rossz függés mentén megbontjuk és a redundanciát okozó tételt egy másik egyedtypusba emeljük ki.

Magyarázat: Az eredeti tervben (lásd 8.2 ábra) fennállt a Cikkszám → Cikknév függés, ami fizikai redundanciát okozott. Ezért a volt egyedtypust megbontjuk. Kiemeljük belőle a bajt okozó tételt (Cikknév) annak meghatározójával (Cikkszám) együtt úgy, hogy az utóbbi az eredeti egyedben is megmarad. Ezek után több eset lehetséges.

- Ha már létezik a CIKK egyed, amelyben együtt van a két tulajdonság, akkor nincs is semmi teendő, csak a Cikkszámot kellett a RENDELÉSTÉTEL-ből eltávolítani.
- Ha már létezik a CIKK egyed, de abban csak a meghatározó (Cikkszám) szerepel, akkor kiegészítjük az egyed tulajdonságsorát a függő (Cikknév) tétellel.
- Ha még nem létezik a CIKK egyed, akkor azt létre kell hoznunk a meghatározóval mint kulccsal, a meghatározottal mint leíróval, amint azt a fenti ábra mutatja.

KOCSI

Rendszám	Típus	Törzsszám
ABC 134	Lada	111111
BCD 265	BMW	222222
DEF 896	Lada	333333
XYZ 999	Fiat	999999

KÁR/KOCSI

Kárszám	Rendszám	Kárösszeg
23000	ABC 134	X
23000	BCD 265	Y
34000	DEF 896	Z
45000	ABC 134	Q

8.4 ábra: Részleges függéstől mentes KOCSI-KÁR modellrész

Magyarázat: Könnyű meggyőződni arról, hogy mindkét példa esetében megszűnnek a tárolási és kezelési anomáliák. Az első példánál a Cikknév tárolása már csak egyszeres és így karbantartása is az. A cikk nevét be tudjuk vinni függetlenül attól, hogy vonatkozik-e az adott cikkre rendelés. Viszont akkor sem veszük el a cikknév ismeret, ha a cikk utolsó rendeltetését töröljük. Azonban a normalizálás nem pusztán a már meglévő ismeretek jobb elrendezése szempontjából alkalmazandó megoldás.

— ***A normalizált szerkezetű adatbázis több ismeret befogadására képes, mint a normalizálatlan.***

Magyarázat: A fenti ábrán kiemeltük az utolsó kocsi, mert az az eredeti 8.1 példában nem szerepelt. Ha a kocsis ismeretek nincsenek a káradatok közé ágyazva, akkor önállóan is kezelhetők. Vagyis egyéb célokból bármikor fel lehet venni új kocsikat az adott egyedbe (több ismeret), amit nem tehattünk meg az eredeti struktúra alkalmazása esetén.

Probléma: A normalizálást megnehezíti az egyértelműség hiánya. A homonimák éppen úgy gondokat okoznak, mint a szinonimák (← Látszólagos és Rejtett logikai átfedés). A következő esetekkel kell számolni:

- Egyed-homonima. Létezik már CIKK egyed, de nem a mi cikk fogalmunkat takarja.
- Egyed-szinonima. Létezik már CIKK egyed, de például TERMÉK néven.
- Kulcs-homonima. Létezik már Cikkszám kulcsú egyed, de ez a kulcs nem pontosan fedi a mi Cikkszám fogalmunkat.
- Kulcs-szinonima. A CIKK egyed kulcsának Cikk kód a neve, de nem vesszük észre, hogy az pontosan megfelel a mi Cikkszám lénységünknek.
- Leíró-homonima. A CIKK egyedben már van Cikknév tulajdonság, de nem pontosan azzal a tartalommal, mint amit mi hasonló név alatt oda akarunk vinni.
- Leíró-szinonima. A CIKK egyedben van egy Cikk megnevezés adat, aminek tartalma megfelel a mi Cikknév tulajdonságunk jelentésének.

Megoldás: A normalizálást nem lehet mechanikusan, csak matematikai alapon végezni. Szemantikai normalizálásra van szükség. Ez két dolgot jelent. Egyrészt a műveletek előtt - amennyire csak lehetséges - meg kell tisztítani előzetes modellünket az egyértelműségi hiányoktól: a homonimáktól és szinonimáktól. Másrészt mivel nem biztos, hogy ezt az előfeladatot tökéletesen hajtjuk végre, a normalizálás során mindig úgy kell keresnünk a kiemelendő tényező (Cikknév) új helyét, hogy közben felszámoljuk az esetleg fennmaradt egyértelműségi gondokat is.

Például ha van CIKK egyedünk, abban Cikk megnevezés leíró tulajdonsággal, akkor két eset lehetséges. Ha tartalma ugyanaz, mint a Cikknév jelentése, akkor az egész modellből kiírtjuk az előbbi nevet és mindenütt az utóbbit használjuk (szinonima). Ha nem ugyanaz a lénység



(homonima), akkor felvetődik az a kérdés, hogy miért van szükség egy egyedben kétféle névszerű hivatkozásra. A válasz függvényében fogunk eljárni.

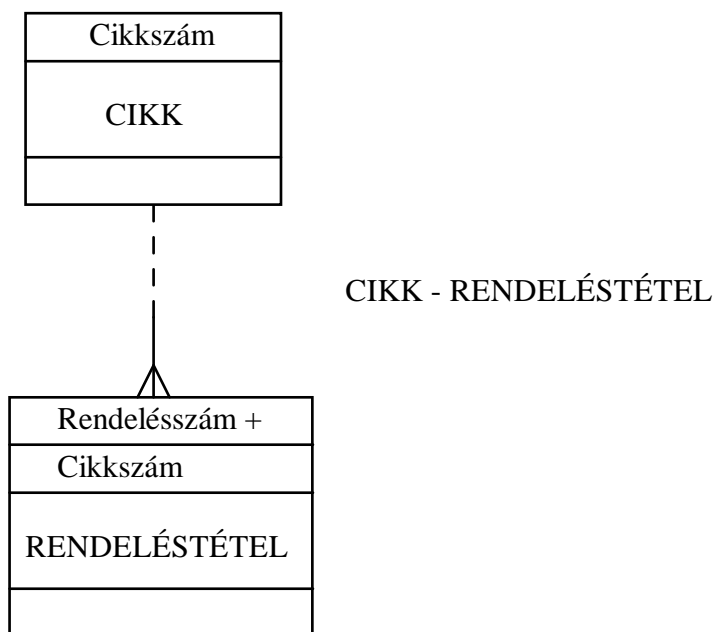
## 8.4 Normalizálás és modellstruktúra

+

**Az egyed akkor és csak akkor van legalább 2NF alakban, ha minden nem-kulcs tulajdonsága teljes függéssel függ az azonosítójától.**

Magyarázat: Ha egy egyed már 1NF alakú, vagyis nem tartalmaz ismétlődést, akkor vizsgálándó a részleges függés. Természetesen erre csak akkor van szükség, ha az egyed azonosítója összetett, mert részleges függés csak ekkor léphet fel. A részleges függéseket a megfelelő tulajdonságok eltávolításával szüntetjük meg. Ez a művelet egyaránt kihat az egyedek belső és külső felépítésére (*Egyedszerkezet*  $\Leftrightarrow$ ).

A kiemelt tulajdonságot (Cikknév) befogadó egyed (CIKK) azonosítója ilyenkor mindig az eredeti egyed (RENDELÉSTÉTEL) összetett kulcsának (Rendelészám+Cikkszám) a része (Cikkszám). Tehát a két egyed között kapcsolat létezik (*Kapcsolótulajdonság*  $\Leftarrow$ ). Ez a kapcsolat mindig birtoklási jellegű, 1:N fokú (hierarchikus) és alulról kötelező erejű. A normalizálás nem árulja el, hogy felülről milyen jellegű: azt külön el kell dönteni. Az összefüggéseket a következő ábra szemlélteti:



8.5 ábra: A normalizálással nyert RENDELÉSTÉTEL modellrész diagramja

## 8.5 Harmadik normálforma

+

Az E egyed nem-kulcs C tulajdonsága akkor és csak akkor tranzitívan függ az egyed A kulcsától, ha azt meghatározza az azonosítótól függő B tulajdonság is.

KOCSI

Rendszám	Típus	Tulajdonos	Foglalkozás	Telephely	Főhatóság	Férőhely
ABC 134	Lada	Szabóné R.	Ápolónő Á	Budapest	-	5 fő
BCD 265	BMW	AB Kft.	-	Szeged	-	5 fő
DEF 896	Lada	Kovács R.	Mérnök M	Pécs	-	5 fő
FGH 333	Polski	XY Rt.	-	Budapest	Z	4 fő

8.6 ábra: 2NF alakú KOCSI egyed

Magyarázat: A KOCSI egyedben nincs ismétlődő tulajdonság és annak azonosítója nem összetett, ezért szükségszerűen legalább második normál formájú. Ugyanakkor a példa a meghatározás szerint tranzitív függést tartalmaz.

RENDELÉS

Rendelésszám	Vevőkód	Vevőnév	...
23000	111	X	
23000	222	Y	
34000	333	Z	
45000	111	X	

8.7 ábra: 2NF alakú RENDELÉS egyed

Probléma: Mindkét ábrán kiemeltük azokat a részsorokat, amelyek *fizikai redundanciát* mutatnak. Ekkor pontosan ugyanazok az anomáliák lépnek fel, amelyeket már korábban ismertettünk (← *Második normálforma*). Ezért a tárolási és karbantartási problémákat itt már nem fejtjük ki még egyszer.

Háttér: Ennek a fizikai átfedésnek az elvi alapját a harmadik Armstrong-szabályban [9] kell keresni. Ez az ún. *tranzitivitási szabály*, amely szerint: Ha együtt fennáll az  $A \rightarrow B$  és a  $B \rightarrow C$  függés, akkor ezekből törvénytzerűen következik az  $A \rightarrow C$  függés is. A 8.6 ábrában a Rendszám meghatározza a Típus és a Férőhely tulajdonságot, viszont a Típus önmagában is meghatározza a Férőhelyet. Ezért a KOCSI egyed tranzitív függést tartalmaz. A 8.7 ábrában a Rendelésszám meghatározza a Vevőkódot és a Vevőnevet, de az utóbbi függ az előbbitől is. Ezért a RENDELÉS egyed sem tökéletes szerkezetű.

+

Az egyed akkor és csak akkor van legalább 3NF alakban, ha minden nem-kulcs tulajdonsága függ a teljes azonosítótól és csakis attól függ.

RENDELÉS		...	VEVŐ	
Rendelésszám	Vevőkód		Vevőkód	Vevőnév
23000	111		111	X
23000	222		222	Y
34000	333		333	Z
45000	111			

8.8 ábra: A RENDELÉS egyed megbontásának eredménye

Magyarázat: Az ábra mindkét egyede 3NF alakban van, mert eleget tesz a definíciónak. Már magán az ábrán is látszik, hogy ugyanazt a megbontási technikát kell alkalmazni, mint a részleges függés esetében (vö. 8.3 ábra). A korábbiakkal szemben csak annyi az eltérés, hogy a két új egyed nem az eredeti kulcsrcsészét képező adatán, hanem annak egy leíróként-kapcsoló tulajdonságán át köthető egymáshoz. Még a nyert új struktúráról is ugyanazt lehet elmondani, mint az előző alponthan.

KOCSITÍPUS	
Típus	Férőhely
BMW	5 fő
Lada	5 fő
Polski	4 fő

KOCSI					
Rendszám	Típus	Tulajdonos	Foglalkozás	Telephely	Főhatóság
ABC 134	Lada	Szabóné R.	Ápolónő	Á Budapest	-
BCD 265	BMW	AB Kft.	-	Szeged	-
DEF 896	Lada	Kovács R.	Mérnök	M Pécs	-
FGH 333	Polski	XY Rt.	-	Budapest	Z

8.9 ábra: A KOCSI egyedtípus megbontása

Magyarázat: A 2NF és a 3NF megbontás között gyakorlatilag csak egy lényeges eltérés van. A *kapcsolótulajdonság* 2NF esetében mindig azonosítórész, a 3NF-nél viszont leíró. Ezért az utóbbi esetben maga a kapcsolat alulról opcionális is lehet. Például vezethetünk ismereteket olyan kocsirol is, amelynek - egyelőre - nem ismerjük a típusát.

Probléma: Rendelés-példánk megoldása végleges. Kocsi-példánkról viszont sejthető, hogy további gondokat rejt, mert pl. a tulajdonos foglalkozásának a többszörös tárolását és kezelését feltételezi. Azonban az eddig tárgyalt módszerekkel nem juthatunk messzebbre.

## 8.6 A normalizálás természete

+

Az egyed akkor van harmadik normálformában, ha minden leíró tulajdonsága függ a kulcsától (1NF), csak a teljes kulcsától (2NF) és semmilyen más tulajdonságától, csak a kulcsától (3NF).

Magyarázat: A fenti meghatározással csak összegezni akartuk a 3NF lényegét. Az ilyen alakú egyedben nincs ismétlődő tulajdonság (0NF), részleges függés (1NF) és tranzitív függés (2NF). Ettől az egyed még nem feltétlenül jól-strukturált, de javításának további technikáira majd csak a következő fejezetekben térünk ki. Itt a normalizálás mikéntjéről kell közreadnunk néhány tudnivalót.

**Automatizálás:** A normalizálást lehet támogatni segédeszközzel, de az eljárás egésze nem automatizálható. Egyetlen szoftver sem képes magától felfedezni például azt, hogy KOCSE mintapéldánk Főhatóság és Felügyelet tulajdonsága valójában azonos; a látszólag egyező Foglalkozás adat némileg eltérő; a Típus itt meg ott mást jelent. Az egyértelműség biztosítása mindig emberi feladat fog maradni. Az ismétlődést tartalmazó egyedeket is csak az ember tudja értelmesen átszerkeszteni (← *Nem-normalizált egyed*).

**Iterációk:** Az ember maga sem képes egycsapásra kiküszöbölni az egyértelműséget sértő összes problémát. Nagyon sokszor előfordul, hogy ha az egyik egyedeket átalakítjuk és onnan áttesszük egy tulajdonságot egy másik egyedbe, akkor az utóbbi válik kedvezőtlen normálformájúvá, holott korábban már korrektnek tűnt. Az iterálás tehát elkerülhetetlen, de az iterációk száma csökkenthető a helyes sorrend megválasztásával.

**Sorrend:** Az iterációkat tekintve nem közömbös, hogy hol kezdjük el a megbontást.

**8.1 példa**    E1 (A+B+C, D, E, F ...)  
                  E2 (A+B, ...)  
                  E3 (A, F)  
                  A+B+C → F  
                  A+B → F  
                  A → F

Magyarázat: Van három egyedünk a mutatott azonosítókkal. Az F tulajdonságot kettő tartalmazza. Világos, hogy a részleges függés miatt azt ki kell emelni az E1 egyedből és csakis az E3 egyedben van a célszerű helye. Akkor követünk helyes sorrendet, ha abból az egyedből indulunk ki, amelynek a kulcsa a legösszetettebb és mindig azonnal az elemi függésekre koncentrálunk.

Ha a normalizálást az E2 vagy E3 egyedeknél kezdjük, akkor semmit sem oldunk meg (mindkettő korrektnek tűnik), de újbóli vizsgálatukra lesz szükség (vö. iteráció). A három mutatott függés közül az első kettő a harmadikból következik. Ha nem azonnal az elemi  $A \rightarrow F$  függésre koncentrálunk az E1 egyed vizsgálatánál, hanem az  $A+B \rightarrow F$  függést fedezzük fel, akkor az F tulajdonság először az E2 egyedbe kerül: E2 (A+B, F ...). Majd itt vesszük észre újra az  $A \rightarrow F$  függést, ami alapján az F tulajdonságot át akarjuk tenni az E3 egyedbe. Már ott van. Ezért ezt az egész bonyolult procedúrát megúszhattuk volna azzal, hogy egyszerűen elhagyjuk az F tételt az E1 egyedből.

Kiegészítés: Előfordul, hogy a normalizálási eljárást egy adott tényezőnél *tudatosan* nem hajtjuk végre.

—    **A normalizálás nem végcél, hanem eszköz.**

Magyarázat: Az 1NF és 2NF alakú egyedek **karbantartási** anomáliákat okoznak. Ezért emeljük ki például a KOCSI egyedből a Férőhely tulajdonságot és alkotunk egy teljesen új KOCSITÍPUS egyedet. Tegyük fel, hogy a kocsi férőhelye nem változik, tehát az adat nem okozhat karbantartási anomáliát. Tudatosan - azaz így átlátva a helyzetet - úgy is dönthetünk, hogy nem vágjuk le a kocsitípusra jellemző tulajdonságokat külön egyedbe, hanem azokat az eredeti KOCSI egyedben tartjuk.

Mi több, az is előfordulhat, hogy már van KOCSITÍPUS egyedünk, és azt - belátva adatainak változatlanóságát - szándékosan összeolvasztjuk a KOCSI egyeddel. Vagyis pl. a Férőhely tulajdonságot az előbbiből áttesszük az utóbbiba (a többivel együtt). Ebben - de csakis ebben - az esetben **denormalizálásról** beszélünk. Ha a tervező ismeri és ezért meg is alkothatná a jó egyed-szerkezetet, mert azt már az elméjében normalizálta, de bizonyos másodlagos megfontolások miatt mégis az alacsonyabb alakot választja, csakis akkor lehet szó denormalizálásról. Visszaalakítani csak már egy kialakított képet lehet. Ha egy egyed úgy csak 2NF alakú, hogy a tervező nem látta meg a tranzitív függéseket, akkor nem lehet az egyedre ráfogni, hogy denormalizált: az egyszerűen csak rosszul tervezett.

**Következetesség:** A normalizálás megszakítását illetve a denormalizálást nem szabad féloldatosan végrehajtani! Jól gondoljuk meg, hogy a normalizálás - szemben az egyéb szakkönyvekben leírtakkal - nem csak az anomáliák felszámolására szolgál! A magasabb normálformájú egyedekben *több ismeret* tárolható, mint az alacsonyabb alakúakban (lásd 8.4 ábra). Ha tehát nem hozzuk létre vagy megszüntetjük a KOCSITÍPUS egyedet, akkor nem fogunk tudni tárolni ismereteket olyan kocsitípusokról, amelyekhez nem tartozik az adott időpontban konkrét jármű. Az pedig végleg megengedhetetlen, hogy a Férőhely tulajdonság hol a KOCSI, hol a KOCSITÍPUS egyedben szerepeljen. Ezért általában a denormalizálást jelen szerző inkább csak elméleti szakemberek újabb ötletének, semmint a gyakorlatban is követendő célszerű megoldásnak tartja.

## 8.7 Egy teljes példa

A profi tervező sohasem lépésenként, az 1NF-2NF-3NF normalizálási sorrendet követve tervez, hanem együtt látja át és oldja meg a problémákat. Azonban a kezdőknek inkább azt ajánljuk, hogy szépen vegyék sorra a teendőket. Ezt mi az alábbiakban mindenképpen megteesszük a KOCSI mintapélda kapcsán, hogy az olvasó legalább egyszer együttesen is láthassa a normalizálás lépéseit. Most a tömörebb formát választottuk a szemléltetésre.

TULAJDONOS	( <b>Törzsszám</b> , Típus, Tulajdonos, Foglalkozás, Telephely, Felügyelet)
KOCSI	( <b>Rendszám</b> , Típus, Tulajdonos, Foglalkozás, Tárolóhely, Főhatóság, Férőhely)
KÁR	(Kárszám, Dátum, Típus, Tulajkód, Kárösszeg)

1. lépés: A normalizálás feltétele az egyértelműség. Ezért legelőször is fel kell számolni a szinonim és homonim neveket, azaz meg kell teremteni a korrekt normalizálási alapot.

TULAJDONOS	( <b>Törzsszám</b> , Tulajtípus, Tulajdonosnév, Foglalkozás, Telephely, Felügyelet)
KOCSI	( <b>Rendszám</b> , Kocsitípus, Tulajdonosnév, Foglalkozás, Tárolóhely, Felügyelet, Férőhely)
KÁR	(Kárszám, Dátum, Kocsitípus, Törzsszám, Kárösszeg)

2. lépés: Az első két egyeddel nem foglalkozunk, mert azok legalább 1NF alakúak. A KÁR egyednek nincs igazi azonosítója, mert rejtetten ismétlődő értékeket tartalmaz. Ezért ez az egyed nem-normalizált (ONF). Mivel a kocsiakat éri kár, az egyedet kiegészítjük a Rendszámmal és az ismétlődő részeket azonnal ki is emeljük belőle. Egyedül a Dátum a nem-ismétlődő adat, tehát az marad az eredeti egyedben. A kiemeléssel új - KÁR/KOCSI - egyedet kell kreálnunk. Ennek kulcsa a szabályok szerint összetett: az eredeti egyednek a kulcsából és az ismétlődés alapjául szolgáló tulajdonságból (Rendszám) áll.

TULAJDONOS	( <b>Törzsszám</b> , Tulajtípus, Tulajdonosnév, Foglalkozás, Telephely, Felügyelet)
KOCSI	( <b>Rendszám</b> , Kocsitípus, Tulajdonosnév, Foglalkozás, Tárolóhely, Felügyelet, Férőhely)
KÁR	( <b>Kárszám</b> , Dátum)
KÁR/KOCSI	( <b>Kárszám+Rendszám</b> , Kocsitípus, Törzsszám, Kárösszeg)

3. lépés: Az első három egyed legalább 2NF alakú, mert egyszerű a kulcsuk. Ezért nem is lehet bennük részleges függés. Így csak az utolsó egyedben vizsgáljuk ezt a problémát. Csak a Kárösszeg függése teljes. A másik két leíró tulajdonság részlegesen függ a kulcs egy részétől, a Rendszámtól. Tehát az egyedet meg kell bontani. Mivel a Kocsitípus már a KOCSI egyedben van, azt egyszerűen elhagyjuk. Viszont a Törzsszám még nem található ott, ezért azt oda be kell illesztenünk.

TULAJDONOS	( <b>Törzsszám</b> , Tulajtípus, Tulajdonosnév, Foglalkozás, Telephely, Felügyelet)
KOCSI	( <b>Rendszám</b> , Kocsitípus, Tulajdonosnév, Foglalkozás, Tárolóhely, Felügyelet, Férőhely, <i>Törzsszám</i> )
KÁR	( <b>Kárszám</b> , Dátum)
KÁR/KOCSI	( <b>Kárszám+Rendszám</b> , Kárösszeg)

4. lépés: A tranzitivitás mindig két meghatározást, két függést feltételez. Két utolsó egyedünkben csak egy-egy függés létezik, de nem fedezünk fel keresztfüggéseket az első egyed tulajdonságai között sem. Ezért csak a KOCSI-val kell törődnünk. Azért, mert abban a kulcsán kívül szerepel egy másik kulcsjellegű tulajdonság is. A Tulajdonosnév, Foglalkozás és Felügyelet tranzitivitását nem nehéz felfedeznünk. Mivel ezek a tételek szerepelnek a Törzsszám azonosította TULAJDONOS-ban is, egyszerűen csak elhagyjuk őket a KOCSI-ból.

A Kocsitípus eddig nem szerepelt kulcsként. Mivel fennáll a Kocsitípus → Férőhely függés, belőle azonosítót kell kreálnunk. Ez pedig egyet jelent azzal, hogy új egyedet is létre kell hoznunk. Lásd a megoldás KOCSITÍPUS egyedét.

TULAJDONOS	( <b>Törzsszám</b> , Tulajtípus, Tulajdonosnév, Foglalkozás, Telephely, Felügyelet)
KOCSITÍPUS	( <b>Kocsitípus</b> , Férőhely)
KOCSI	( <b>Rendszám</b> , <i>Kocsitípus</i> , <i>Törzsszám</i> , Tárolóhely)
KÁR	( <b>Kárszám</b> , Dátum)
KÁR/KOCSI	( <b>Kárszám+Rendszám</b> , Kárösszeg)

Probléma: Nem akartuk a gondolatmenetet megszakítani, azért nem mutattunk rá egy értelmzési gondra. Amikor a Foglalkozást el akartuk tüntetni a KOCSI egyedből, annak helyét a TULAJDONOS-nál találtuk meg. Azonban - emlékezzünk az eredeti példára - a foglalkozás nem egyértelmű, azt a két egyedben másként kódolták. Ilyenkor két eset lehetséges.

- Ha elvileg azonos lényegekről van szó, akkor ki kell irtani az egyikféle kódolást és az egész adatbázisban/adatmodellben a másikfélét kell következetesen alkalmazni.
- Ha a lényegek nem pontosan fedik egymást, akkor két megfelelő nevű tulajdonságot kell a TULAJDONOS-ban felvenni. Ilyesmi bizony előfordul a gyakorlatban. A FEOR-számot a foglalkozások egységes országos jegyzéke szerint vezették. Ezt a jegyzéket néha átértelmezték, vagyis a mai FEOR nem felel meg a réGINEK. Ha az utóbbira is szükség van, akkor nincs más megoldás, vezetni kell a FEOR-mai/FEOR-régi ismeretpárost.

## 8.8 A dekompozíció sajátosságai

Definíció: Eddig a normalizálást dekompozícióval végeztük. A kedvezőtlen formájú egyedeket több egyedre bontottuk le. Azt a műveletet, amelynek során oszlopokat emelünk ki egy relációból, *kivetítésnek* [projection] hívjuk. Azt a másik eljárást, amelynek során két relációt azok közös oszlopai alapján egy harmadikba egyesítünk, *összekapcsolásnak* [join] nevezzük. A normalizálás kivetítések és összekapcsolások sorozata.

**8.2 példa**      VEVŐ-1 (**Vevőkód**, Vevőnév)  
                     RENDELÉS-1 (**Rendelészám**, Vevőkód, Vevőcím)

                    VEVŐ-2 (**Vevőkód**, Vevőcím)

                    VEVŐ (**Vevőkód**, Vevőnév, Vevőcím)  
                     RENDELÉS (**Rendelészám**, Vevőkód)

Magyarázat: Az első lépésben a RENDELÉS-1 egyedből kivesszük a Vevőkód-Vevőcím párost. (Ez annyiban nem valódi kivetítés, amennyiben a Vevőcímet végleg eltávolítjuk az egyedből.) Így átalakul az eredeti reláció: a RENDELÉS mutatja az eredményt. Létrejön egy átmeneti reláció (VEVŐ-2) a kiemelt tételekkel. Majd a második lépésben ezt a Vevőkód alapján összekapcsoljuk a VEVŐ-1-el. Így születik meg a végső VEVŐ reláció.

—      ***A dekompozíciónak veszteségmentesnek kell lennie.***

Magyarázat: A dekompozíció megfordítható - *reverzibilis* - művelet. Ez azt jelenti, hogy az eredmény-relációkból mindig visszaállíthatók az eredetiek. A Vevőcímet ki lehet vetíteni a VEVŐ-ből és újra át lehet tenni a RENDELÉS-be (vö. denormalizálás). Ha e művelet során az eredeti tábláknak nemcsak a szerkezete, hanem a tartalma is helyreáll, akkor beszélünk *veszteségmentes dekompozícióról* [non-loss decomposition]. Az 1-3NF szerinti megbontás mindig ilyen. (A későbbiekben lesz szó veszteséges lebontásokról is.)

Megjegyzés: A visszafordíthatóság nem jelent egyenértékű terveket. A normalizálással nyert struktúra több ismeret befogadására képes (← *A normalizálás természete*), mint az eredeti. Például olyan kocsitípusról is vezethetünk adatokat, amelyek nem szerepeltek a normalizálatlan KOCSI egyedben (vö. 8.8 ábra). A veszteségmentesség csak azt jelenti, hogy az eredeti adatbázis

visszaállítható, abból (!) nem veszítünk ismereteket. Ugyanakkor a denormalizálással természetesen elvesznek a többletismeretek.

— ***A dekompozíciónak független relációkat kell eredményeznie.***

Magyarázat: A veszteségmentesség a dekompozíciónak szükséges, de nem elégséges feltétele. Az általános szakirodalom megkülönböztet *jó* és *rossz* lebontásokat (lásd például [10], 253. oldal). Vizsgáljuk meg közelebbről ezt a minősítést!

### 8.3 példa RENDELÉS (**Rendelésszám**, Vevőkód, Vevőnév)

VEVŐ (**Vevőkód**, Vevőnév)

RENDELÉS-1 (**Rendelésszám**, Vevőkód)

RENDELÉS-2 (**Rendelésszám**, Vevőkód)

RENDELÉS-3 (**Rendelésszám**, Vevőnév)

Magyarázat: A RENDELÉS-t mi természetesen az első két párosra bontjuk le. Azonban az egyes szakírók szerint - elvileg lebontható a második két párosra is. Mindkét változat veszteségmentes. Ennek dacára az első lebontást jónak, a másodikat rossznak tartják.

A második megoldás frissítési anomáliát mutat. Tegyük fel, hogy X-ről Y-ra írjuk át az 1 számú rendelésben a vevő nevét! A volt X nevet a vevő minden rendelésében frissíteni kell, ami problémát okoz, mert nem csak egy vevőnek lehet ilyen neve! Ezért kikeressük a RENDELÉS-2 egyedből az 1 számú rendelést és megállapítjuk a Vevőkód értékét. Legyen az például Q. Ezek után ki kell keresni az összes rendelést, amelyben a Vevőkód szintén Q, összeállítva a rendelésszámok vektorát. Ez alapján a RENDELÉS-3-ból kell kikeresni a megfelelő előfordulásokat és átírni azokban is a vevő nevét Y-ra. Ebben az esetben a megbontás **nem-független** [non-independent], mert az egyik egyed karbantartásának a feltételeit a másik egyed több előfordulása rejti.

Az első esetben nem lép fel a jelzett probléma. A megbontás **független** [independent]. Nem rejtettük el a tranzitivitást, mint a másik megoldásnál, hanem azt tovább hordozza az új szerkezet is a két egyed kapcsolótulajdonsága által. A vevő nevét mindig csak egy helyen kell átírni. A Vevőkód karbantartására pedig csak kapcsolati korlát vonatkozik. Az első megoldás tehát jó, miközben a második rossz.

**Figyelmeztetés!** Mindezt csak az elméleti teljesség kedvéért idéztük fel. A kérdéses szakírók gondolkodásmódja ugyanis tökéletesen hibás. Az adatmodell kizárja, hogy két egyednek ugyanaz legyen az azonosító tulajdonsága. Tehát a második megoldást nem is kell mérlegelni: az sohasem lehet jó.

Kiegészítés: Mi a dekompozíciót nem *bármilyen* függés mentén végezzük el, hanem *a rossz* - részleges, tranzitív - függést küszöböljük ki. Ez a megbontás csak azzal jár, hogy az **egyeden belüli** korlát **egyedek közötti** feltétellé válik, de maga a korlát nem veszik el. Az első megoldás után a két egyed viszonya 1:N fokú, a másodiknál viszont M:M fokú. Márpedig többször utaltunk arra, hogy hálós viszonyban nem létezik közvetlen kapcsolat. Az 1-3NF alakok esetében mindig csak egyetlenegy jó megbontás lehetséges. Az egész problémakört éppen azért kellett felvetnünk, mert a későbbi alakok némelyikénél több megbontási változat is akad, amelyek a fenti értelemben nem egyenértékűek.



## 8.9 Alternáló kulcs

— **Alternáló kulcson nem szabad dekompozíciót végezni.**

**8.4 példa** VEVŐ (Vevőkód, Vevőnév, Vevőcím) - a vevőnevek is egyediek

VEVŐ-1 (Vevőkód, Vevőcím)

VEVŐ-2 (Vevőnév, Vevőcím)

Magyarázat: A relációs adatbázisban a kulcsjelöltek közül ki kell választani azt, amit elsődleges kulcsnak óhajtunk tekinteni, a többi alternáló kulcs marad. Mivel példánkban a Vevőnév is egyedi értékű, a Vevőkóddal kölcsönös függésben áll.

Probléma: Egyes szakírók azon medítálnak, hogy miképpen kell a fennálló függés miatt a VEVŐ egyedet megbontani. Vajon a Vevőcímet a VEVŐ-1 vagy a VEVŐ-2 egyedbe kell-e tenni. Vagyis szerintük az alapegyednek két egyenrangú megbontása létezik.

Megoldás: Ez a gondolatmenet teljesen hibás. Ugyanis a kölcsönös függés miatt vagy a VEVŐ-1 egyedbe kellene tenni a Vevőnevet, vagy a VEVŐ-2 egyedbe a Vevőkódot. Így pedig visszakapnánk az eredeti relációt. Az alternáló kulcs azért az, ami, mert maga is lehetett volna elsődleges kulcs. Ezért értelemszerűen minden függési viszonya azonos az elsődleges kulcsként kiválasztott tételével. Ezért a „valódi” alternáló kulcsokat egyáltalán nem kell vizsgálni a függések elemzése során, mert azokon nem szabad normalizálni.

**8.5 példa** KOCSI (**Rendszám**, ..., Casco-szám, Kötés-dátum, Önrész, ...)

Kérdés: A fentiek szerint nem szabad a KOCSI egyedet megbontani és létrehozni egy külön CASCO egyedet?

Válasz: A fentiekben „valódi” alternáló kulcsokról volt szó. Olyan esetekről, amikor két tulajdonság kölcsönösen meghatározza egymást. Bár a Casco-szám és a Rendszám egy-az-egy-es viszonyban áll egymással, függésük nem kölcsönös, mert nem erős. Ugyanis nem minden kocsinak van Casco-biztosítása. Ezért a Casco-szám nem alternáló kulcs (mert nem is lehetne a KOCSI azonosítója). A Rendszám → Casco-szám → Önrész függéssor miatt felmerülhet a normalizálás igénye. Létre lehet hozni a külön CASCO egyedet, de azt nem az eddig ismertetett módon tesszük (→ Feltételes függés).

## 8.10 Ellenőrző kérdések

8/1 Hányadik normálformában van a következő egyed? Számmal válaszoljon.

RENDELÉS (**Rendelésszám**, R-dátum, ..., Cikkszám, Rendelt-mennyiség)

R1	D1	C1	M1
		C2	M2
R2	D2	C1	M3

8/2 Legalább hányadik NF alakban van az ismétlődést nem tartalmazó egyed?

8/3 Hányadik normálformájú a következő egyed:

SZÁMLATÉTEL (*Számlaszám*+*Cikkszám*, ..., Tételérték, Egységár)

8/4 Mi az előző példa helyes megoldása?

8/5 Adott a két alábbi egyedtípus. Mit tenne Ön a tranzitivitás megszüntetése során?  
Adja meg a helyes válasz sorszámát.

VEVŐ (**Vevőkód**, ..., Vevő levelezési cím)

RENDELÉS (**Rendelésszám**, ..., *Vevőkód*, Vevőcím)

- Mivel a Vevőcím duplikátum, egyszerűen elhagynám a RENDELÉS-ből.
- Felkeresném a felhasználót, hogy a két cím azonos tartalmú-e.
- A cím és a levelezési cím két dolog. Tehát a Vevőcímet a VEVŐ-be tenném.

8/6 Hányadik normálformában van a következő egyed (KB = Kötelező biztosítás)?

KOCSI (**Rendszám**, ..., KB kötvényszám, KB kötésdatum)

## 9. MAGASABB NORMÁLFORMÁK

### 9.1 Hány, melyik és milyen a kulcs?

Az alapnormalizálás (lásd az előző két fejezetet) viszonylag könnyű, rutinszerű feladat. Az első nagyobb gondok akkor támadnak, ha az egyedben több kulcsjelölt ill. úgynevezett *kulcsjellegű* tulajdonság van. Tudni kell azt, hogy a szakirodalomban nemcsak az olyan tételeket nevezik kulcsoknak (sajnos), amelyek az adott relációban valódi kulcsjelöltek, hanem azokat is, amelyek más tulajdonságot a reláción belül meghatároznak. Tehát pl. a RENDELÉS (Rendelészám, Vevőkód, Vevőnév) egyedben a Vevőkód is kulcsjellegű (meghatározza a Vevőnevet), sőt, egyedi értéke esetén a Vevőnév is az.

A magasabb formáknál a normalizálás nem abból indul ki, hogy adott a kulcs és milyen „nem-kulcs” tételek miképpen függenek vagy nem függenek tőle. Azokat a problémákat vizsgáljuk, amelyeket a „kulcs” tételek közötti mindenféle viszonyok okoznak. Például az ún. csupakulcs relációkban funkcionális függés nem is létezhet, mégis adott esetben az ilyen táblákat is meg lehet bontani.

*Ennek a fejezetnek az a célja, hogy feltárja a többféle kulcsjelölt esetén alkalmazandó megoldásokat és ismertesse az olyan egyedek megbontásait, amelyek csak kulcsot tartalmaznak, de mégis karbantartási anomáliákat mutatnak.*

Fejezetünk három helyzetet ismertet. Az első eset még úgy, ahogy átlátható (9.2 *Külső kulcs-törő függés* és 9.3 *Boyce-Codd normálforma*). Ekkor a nem-csupakulcs relációban van olyan tétel, amelyik meghatározza a kulcsjelölt valamelyik részét. Ezt a szituációt olykor nem lehet feloldani (9.4 *A normalizálás negyedik lépése*).

A másik két problémakör csak csupakulcs relációk esetében léphet fel. Már az egyik helyzet sem egyszerű (9.5 *Többértékű függés* és 9.6 *Negyedik normálforma*), a másik pedig kimonódottan bonyolult (9.7 *Kapcsolásfüggés* és 9.8 *Ötödik normálforma*).

A normálformák egymásba ágyazottak. Ha egy egyed 5NF alakban van, akkor egyben eleget tesz a 4NF, 3NF stb. követelményeinek is. Ezért egyes szakírók azt ajánlják, hogy a tervezők ne is törődjenek az alacsonyabb alakokkal, hanem azonnal törekedjenek a létező legjobbra, az 5NF alakra. Ettől mi mindenkit óva intünk!

A szerző a 4-5NF alakokat gyakorlatilag használhatatlan elméleti huncutságnak tartja. Majd az olvasó is meggyőződhet arról, hogy e két formának már az elvi megértése is komoly erőfeszítést igényel, nemhogy gyakorlati alkalmazása. Ezért a többértékű és a kapcsolási függés alapján normalizálni azt az egyedet, amelynek dekompozíciója a tiszta és világos funkcionális függések mentén is lehetséges, finoman szólva nem célszerű. A magasabb normálformák nagyon sok esetben igen mesterséges konstrukciók. Mi pedig azt ajánlottuk eddig is, hogy a tervezők ragaszkodjanak a természetes megoldásokhoz.

## 9.2 Külső kulcstörő függés

TANÍTÁS

Diák	Tárgy	Tanár
Kovács	matek	Fehér
Kovács	fizika	Fekete
Szabó	matek	Fehér
Szabó	fizika	Barna

9.1 ábra: Kétes szerkezetű TANÍTÁS tábla

Magyarázat: Az esetet C. J. Date-től vettük át [<sup>10</sup>, 251. oldal]. A feltételek az alábbiak:

- Minden tárgyra nézve igaz, hogy az azt tanuló diákok csak egy tanár tanítja a kérdéses tárgyra. Ezért fennáll a Diák+Tárgy  $\rightarrow$  Tanár függés.
- Minden tanár csak egy tárgyat tanít. Ezért létezik a Tanár  $\rightarrow$  Tárgy függés.
- Egy tárgyat több tanár is taníthat. Ezért nem létezik a Tárgy  $\rightarrow$  Tanár függés.

Probléma: A tábla **karbantartási anomáliát** mutat. Ha töröljük belőle az utolsó sort, amely szerint Szabó fizikát tanul, akkor elveszik az az ismeret is, hogy Barna tanár úr fizikát tanít, ha Szabó volt az utolsó tanítványa. (A gyakorlatban nem valószínű, hogy egy tanár csak egy diákokat tanít egy tárgyra. A példával csak a baj jellegét szemléltetjük.) Nem tudjuk beilleszteni azt az ismeretet, hogy Zöld tanár úr földrajzot tud okítani mindaddig, ameddig nem akad ilyen tárgyat tanuló diák. Ha Fehér tanárnő férjhez megy és Pirosra változtatja a nevét, akkor a módosítást több soron kell végrehajtani. A karbantartási visszasságok miatt úgy tűnik, hogy a táblát át kellene alakítani.

TANÁR

Tanár	Tárgy
Fehér	Matek
Fekete	Fizika
Barna	Fizika

TANULJA

Diák	Tárgy
Kovács	Matek
Kovács	Fizika
Szabó	Matek
Szabó	Fizika

9.2 ábra: A TANÍTÁS egyféle megbontása

Magyarázat: Az eredeti táblában nincsen elemi meghatározó, amit elsődleges kulcsként vehetnénk fel. Van viszont több „kulcsjellegű” tulajdonság. A szakirodalom szerint azok a tételek ilyenek, amelyek egyenként vagy együttesen másokat meghatároznak. A fenti feltételek szerint ilyen a Diák+Tárgy páros és a Tanár. Korábbi definícióink szerint az alapvető normálforma hibák csak akkor lépnek fel, ha a „nem-kulcs” tulajdonságok függetlenek a kulcstól, vagy attól részlegesen illetve tranzitíven függenek. Az irodalom úgy tartja, hogy a fentiek szerint a TANÍTÁS tábla 3NF formájú és nem normalizálható az eddigi alapokon. Azért nem, mert **egyik meghatározó sem a tábla kulcsjelöltje**.

Probléma: Ha a Diák+Tárgy meghatározóból indulunk ki, akkor a Tanár  $\rightarrow$  Tárgy függés miatt ezt a párost kell kiemelni a táblából. Azonban a 9.2 ábra eredménye rossz megbontáshoz vezet

(← *A dekompozíció sajátosságai*). Bár a két táblából visszaállítható a TANÍTJA egyed, karbantartási gondok lépnek fel. Például Szabóhoz bevihetünk olyan tantárgyat, amelyet egyetlen tanár sem oktat. Ezt azért tehetjük meg, mert a két egyed között nincs kapcsolat. A TANÍTJA egyeden *belüli* korlátot (a Tárgy mellett mindig van Tanár is) nem őriztük meg a két egyed *közötti* korlátként (a TANULJA nem kötődik a TANÁR-hoz). A megbontás két hálós viszonyú - nem kapcsolható - táblát eredményezett.

TANÁR		TANÍTJA	
Tanár	Tárgy	Diák	Tanár
Fehér	Matek	Kovács	Fehér
Fekete	Fizika	Kovács	Fekete
Barna	Fizika	Szabó	Fehér
		Szabó	Barna

9.3 ábra: A TANÍTÁS másféle megbontása

Magyarázat: Erre a megoldásra úgy jutottunk el, hogy a Tanár  $\rightarrow$  Tárgy függés alapján a korábban megszokott normalizálási utat választottuk. Ott pedig a szabályok úgy szóltak, hogy a függő tulajdonságot (Tárgy) a meghatározójával (Tanár) együtt ki kell emelni a relációból, de a meghatározó az eredeti táblában is megmarad. Ez a megoldás egy fokkal jobb, mint az előző. Azért az, mert a két egyed hierarchikus viszonyban áll, vagyis a belső korlátból külsőt kreáltunk.

Probléma: Ez a megoldás is nem-független lebontást eredményezett. A második táblába be lehet vinni a Szabó-Fekete párost, hiszen létezik Fekete tanár úr, mint fölérendelt. Ám ezzel a bevitellel megsértjük azt a korlátot, amely szerint egy tárgyat (fizika) a diák csakis egy tanártól tanulhat, hiszen Szabót már Barna tanár úr tanítja fizikára. Ezt a bevitelt csak a két tábla összetett elemzésével lehet megakadályozni (vö. nem-független lebontás).

**+** Az E egyed típus C tulajdonsága akkor és csak akkor okoz külső kulcstörő függést, ha függ az A+B összetett azonosítótól és ugyanakkor meghatározza annak A vagy B részét.

Magyarázat: Tételezzük fel, hogy a TANÍTÁS egyednek a Diák+Tárgy páros a kulcsa, ugyanis ettől függ a Tanár. Ugyanakkor fennáll a Tanár  $\rightarrow$  Tárgy függés is. Vagyis az egyed egyik leíró tulajdonsága meghatározza a kulcs egy részét, azt mintegy megtöri. A példa tehát külső kulcstörést mutat. A „külső” jelző arra utal, hogy a kulcson kívüli tétel okozza a bajt ( $\rightarrow$  *Belső kulcstörő függés*).

### 9.3 Boyce-Codd normálforma

**+** Az egyed akkor és csak akkor van Boyce-Codd normálformában, ha minden meghatározó tulajdonsága egyben kulcsjelölt is.

9.1 példa TANÍTÁS (Diák, Tárgy, Tanár)

Magyarázat: A  $\text{Diák} + \text{Tárgy} \rightarrow \text{Tanár}$ ,  $\text{Tanár} \rightarrow \text{Tárgy}$  és  $\text{Diák} + \text{Tanár} \rightarrow \text{Tárgy}$  három függés alapján a táblában három meghatározó van (még akkor is, ha a harmadik függés részleges jellegű). Közülük a két összetett tétel kulcsjelölt, mert tőlük függ a harmadik tulajdonság is. Viszont a Tanár olyan meghatározó, amelytől nem függ a reláció összes többi tétele, ezért ő nem lehet kulcsjelölt. Az egyedre nézve nem igaz a fenti definíció. A TANÍTÁS 3NF alakban van, de nincs Boyce-Codd [11] normálformában (BCNF). A *külső kulcstörő függés* egy sajátos ciklust okoz ( $\text{Diák} + \text{Tárgy} \rightarrow \text{Tanár} / \text{Tanár} \rightarrow \text{Tárgy}$ ), ami miatt az egyednek nem létezik független dekompozíciója.

## 9.2 példa VIZSGA (Diák, Tárgy, Helyezés)

VIZSGA-1 (*Diák* + *Tárgy*, Helyezés)

VIZSGA-2 (*Tárgy* + *Helyezés*, Diák)

Magyarázat: A diákok adott tárgyakból vizsgáznak. A feltételezés az, hogy a diákokat a tárgyban úgy minősítik, hogy helyezési számot adnak nekik és minden helyezés egyedi. Vagyis két diák nem kaphatja ugyanabból a tárgyból ugyanazt a helyezést. Két kulcsjelölt adódik: a *Diák* + *Tárgy* és a *Tárgy* + *Helyezés*. Bár ezek átfednek, a VIZSGA mégis BCNF alakban van, mert nincs benne olyan meghatározó, amely nem kulcsjelölt. A kérdés most az, hogy vajon a két kulcsjelölt egyenrangú-e, egyformán választható-e kulcsnak?

Probléma: Látszólag a két megoldás egyenértékes. Azonban a helyezés egyediségének a korlátja az első verzióban nem érvényesíthető (egy tárgyból többen is elérhetnének azonos helyezést). Ezért csak a második verzió tesz eleget az összes feltételezésnek. Ez az eset azt példázza, hogy az *alternáló kulcsok* sokszor csak formailag egyenértékesek: az elsődleges kulcs végső kiválasztásánál további aspektusokra is figyelni kell.

Megjegyzés: A szerző nem kedveli, ha a matematika túlzottan a szemantika hátrányára érvényesül. A fentiek dacára ő bizony a VIZSGA-1 egyedet választaná megoldásnak, mert „a diák tanulja a tárgyat” összefüggést az mutatja jobban. Mivel az adatmodell nemcsak struktúra, hanem korlátokat is tartalmaz, abban kötné ki a Helyezés egyediségét.

## 9.3 példa RENDELÉSTÉTEL (Rendelésszám, Tételsorszám, Cikkszám, Mennyiség)

Magyarázat: Az egyedben két összetett kulcsjelölt van: a *Rendelésszám* + *Tételsorszám* és a *Rendelésszám* + *Cikkszám* páros. A két kulcsjelölt átfed. Viszont azokon kívül nincs más meghatározó az egyedben. Ezért az egyed BCNF alakban van és nem szükséges azt megbontani, csak ki kell választani az elsődleges kulcsot. Itt nem lép fel olyan külön korlát, mint az előző példánál, ezért a választás elvileg szabad: a két kulcs egyenértékes. Azonban a szerző itt is a természetes *Rendelésszám* + *Cikkszám* párost választaná. Azért, mert annak mindkét tagja kapcsolótulajdonság (az itt nem mutatott rendelés illetve cikk egyedek felé), míg a *Tételsorszám* nem kapcsolóselemből áll.

—

*A változatok közül mindig a természetesebbet válasszuk.*

Magyarázat: A mesterséges megoldás felesleges tényezőket igényel (vö. *Tételsorszám*). A nem-strukturális korlátokat azok esetében sem ússzuk meg. Ha a *Cikkszám* nem a kulcs része, akkor korláttal kell ellenőrizni, hogy az kitöltött-e. (A kulcs és részei esetében ez a korlát magából a

szerepből adódik.) A természetes megoldás sok esetben több ismeretet enged meg, mint a mesterséges. A 9.2 példa két megoldása közül az elsőben ki lehet azt is fejezni, hogy valaki tanul egy tárgyat, de nem vizsgázik belőle. A második megoldásban ez a lehetőség nem áll fenn.

#### 9.4 példa AKCIÓ (Akciónkód, Vevőkód, Vevőnév, Kedvezmény)

Magyarázat: Egy akcióban több vevő vehet részt és egy vásárló több akcióban érintett. Ezért a Kedvezményt az akció és a vevő kulcsa csak együttesen határozza meg. A vevők nevei legyenek egyediek. Ekkor az AKCIÓ egyednek két összetett kulcsjelöltje van: az Akciónkód+Vevőkód és az Akciónkód+Vevőnév páros. Az egyed 3NF alakú, mert egyetlen „nem-kulcs” tétele (Kedvezmény) teljes függéssel függ a lehetséges kulcsoktól és másától nem. Ugyanakkor az egyed nem BCNF formájú, mert van benne két olyan meghatározó (Vevőkód, Vevőnév), amely nem ennek az egyednek a kulcsjelöltje (egymástól függenek).

Probléma: Az egyed karbantartási anomáliát mutat. Ugyanaz a Vevőnév érték több előfordulásban is szerepelhet. A 111 kódú vevő X nevét minden további nélkül átírhatjuk Y-ra az AKCIÓ egyik sorában, miközben az ugyancsak 111 Vevőkód értéket tartalmazó másik sorban ezt nem tesszük meg. Arról pedig ne is beszéljünk, hogy az AKCIÓ fizikailag redundáns, hiszen többszörösen tartalmazza az „111 - X” értékpárost. Ezért a szakirodalom azt állítja, hogy az AKCIÓ egyedét BCNF-re kell normalizálni. A Vevőkód és Vevőnév kiemelése után kétféle eredmény adódhat: az AKCIÓ-1 (*Akciónkód+Vevőkód*, Kedvezmény) és az AKCIÓ-2 (*Akciónkód+Vevőnév*, Kedvezmény) egypéldatípus. Ez a kettő a szakírók szerint egyenértékes.

Megoldás: Szerintünk a szakírók tévednek. A kiemeléskor el kell dönteni, hogy maga a VEVŐ egyed miképpen épül fel, vagyis ki kell jelölni annak elsődleges kulcsát. Logikus, hogy a VEVŐ (*Vevőkód*, Vevőnév) egyed mellett döntünk úgy, hogy a Vevőnév *alternáló kulcs*. Mindezek után az AKCIÓ-2 megoldás már nem jöhet számításba, mert korábbi szabályunk szerint az alternáló kulcsokon nem határozunk meg függéseket. Tehát a 9.4 példában egyáltalán nem lép fel normalizálási probléma, mert az AKCIÓ egyedben eleve nem is szerepelhet a Vevőnév tulajdonság.

## 9.4 A normalizálás negyedik lépése

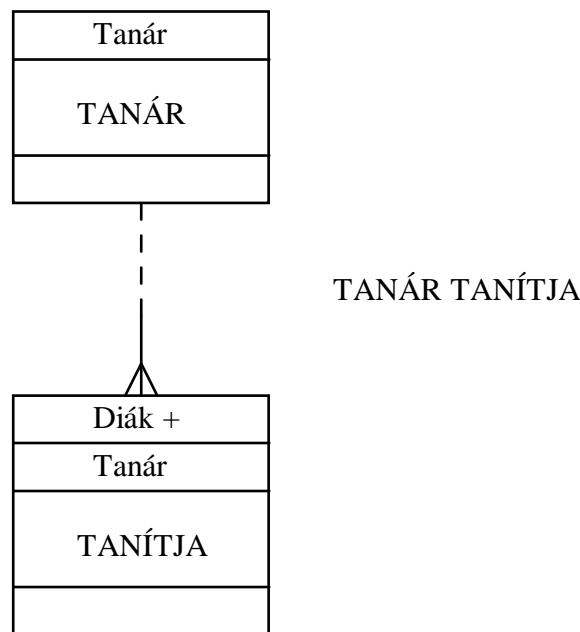
+

**Atomnak nevezzük az egyedet, ha az több kulcsjelöltet tartalmaz, de nem bontható le egymástól független egyedekre.**

Magyarázat: A független lebontásról már volt szó (← *A dekompozíció sajátosságai*). A fenti definíció J. Rissanen [<sup>12</sup>] nevéhez fűződik. Meghatározása értelmében a 9.1 ábra TANÍTJA egyede atomi volt, mert az egyed több kulcsjelöltet tartalmaz és nem találunk rá független megbontást. A szerző - óvatosan - úgy fogalmaz, hogy bár ilyen esetekben is van mód normalizálásra (a 9.2 és 9.3 ábra BCNF alakú egyedeket mutat), lehet, hogy a normalizálás végrehajtása „nemkívánatos”.

Ez a kitétel gyakorlatilag arra utal, hogy a normalizálás negyedik lépése kétféle jellegű lehet. Ha találunk független lebontást, akkor azt végrehajtjuk. Ha nem, akkor döntés előtt állunk. Mérlegelnünk kell azt, hogy az eredeti egyed megbontatlanul hagyása illetve annak különböző

megbontásai közül melyik okozza számunkra *tartósan* is a legsúlyosabb karbantartási gondokat. (Például nem valószínű, hogy huzamosan is érvényes az a korlát, amely szerint a diákokat csak egy tanár taníthatja egy tárgyra vagy amely szerint egy tanár mindig csak egy tárgyat okít.) Ha találunk a megbontások között olyat, ami kevés bajjal jár, akkor végrehajtjuk a dekompozíciót. Egyébként az eredeti egyedet nem bántjuk.



9.4 ábra: A TANÍTÁS modellrészlet diagramja

Magyarázat: Semmiképpen sem javasunk olyan megbontást, amely után az eredmény-egyedek egymással hálós viszonyban állnak (vö. 9.2 ábra). Ezért ha netán a dekompozíció mellett döntünk, akkor mindig egymással kapcsolható egyedekhez jutunk (vö. 9.3 ábra). Ezzel legalább a valósághűség kritériumának a teljesítéséhez jutunk közelebb, mert hiszen a TANÁR egyed mindenképpen tükrözi azt a tényt, hogy egy tanár csak egy tárgyat tanít.

Kiegészítés: Az egymásba skatulyázás elve alapján tudjuk, hogy minden BCNF alakú egyed egyben szükségszerűen 3NF formájú is. Elméletileg az eredeti egyed karbantartási anomáliáinak a megszüntetése miatt a BCNF forma jobb, mint a 3NF. Gyakorlatilag esete válogatja: van, amikor a BCNF több nehézséget okoz, mint amennyit megszüntet. Ezért csakis az elméleti teljesség kedvéért ismertettük.

## 9.5 Többértékű függés

+

**Az E egyed B tulajdonsága akkor és csak akkor többértékű függéssel függ az A tulajdonságtól, ha az adott A-értéknek megfelelő B-értékek készlete csak az A-tól függ és független az egyed C tulajdonságától.**



#### TANFOLYAM

Kurzus	Tanár	Téma
Fizika	Zöld	Mechanika
Fizika	Zöld	Optika
Fizika	Barna	Mechanika
Fizika	Barna	Optika
Fizika	Fekete	Mechanika
Fizika	Fekete	Optika
Matematika	Fehér	Algebra
Matematika	Fehér	Geometria

9.5 ábra: Furcsa ismétlődéseket tartalmazó TANFOLYAM tábla

Magyarázat: Ezt a példát is Date [10] kiadványából vettük. Egy tanfolyamon több tanár több témát tanít. Ha egy tanár a kurzuson részt vesz, akkor annak minden témáját tanítja. Fordítva: Ha egy téma a kurzus része, akkor azt minden tanár tanítja, aki a tanfolyamon egyetlen témát is okít. Zöld tanár úr mechanikát és optikát tanít, mert ez a két téma tartozik a fizika kurzushoz. Az optikát Barna tanár úr is előadja, ha a kurzuson már a mechanika oktatásában is részt vesz.

Probléma: A nem-normalizált egyedekben *ismétlődő csoportok* vagy adatok találhatók. Az ilyen egyedeket 1NF formára kell hozni. Azonban az ismétlődésnek van egy sajátos, rejtett esete is, amely nem oldható fel a korábban tárgyalt normalizálási technikával. A fenti tábla ugyanis már BCNF alakban van (← *Boyce-Codd normálforma*). Nem is lehet másban, mert hiszen ún. *csupakulcs* [all-key] reláció, vagyis minden tétele az összetett kulcs része. Ennek ellenére az ábra relációja karbantartási anomáliákat mutat. Például ha Zöld tanár úr kilép, akkor több sort is törölni kell. Ha Piros tanárnő a fizika kurzuson oktatni kezd, akkor többszörös bevitelre (mechanika, optika) van szükség. Végül ha Fehér tanárnő férjhez megy és nevét Kékre változtatja, többszörös karbantartást kell végezni.

Megoldás: Az eddigi normalizálási eljárások nem alkalmazhatók a karbantartási bajok kiküszöbölésére. Új koncepcióra van szükség. A táblában a Tanár és Téma párosok egy adott rendszer szerint jelennek meg. Ennek mintája az alábbi módon fejezhető ki:

Ha megjelenik a táblában a  $\langle k1, t1, m1 \rangle$  és a  $\langle k1, t2, m2 \rangle$  sor,  
akkor szerepelnie kell abban a  $\langle k1, t1, m2 \rangle$  és a  $\langle k1, t2, m1 \rangle$  soroknak is.

Ha az első kurzuson ( $k1$ ) az első tanár ( $t1$ ) az első témát ( $m1$ ), a második tanár ( $t2$ ) a második témát ( $m2$ ) tanítja, akkor az első tanárnak is kell tanítania a második témát és a másodiknak is az első. A hasonló helyzetek feloldására vezették be a *többértékű függés* [multivalued dependency] fogalmát [13]. Ennek rövid neve *MVD*, jele pedig „ $\twoheadrightarrow$ ”.

Magyarázat: Példánkban a Tanár többértékű függéssel függ a Kurzus tulajdonságon, mert a Kurzus értékhez (Fizika) adott Tanár értékhalmoz (Zöld, Barna, Fekete) tartozik és a Téma nem határozza meg a Tanárt (minden témát több tanár okíthat). Vegyük észre, hogy a többértékű függés mindig párosával jár. Ha fennáll az  $A \twoheadrightarrow B$  függés, akkor léteznie kell az  $A \twoheadrightarrow C$  függésnek is. Ha a Kurzustól többértékűen függ a Tanár, akkor azt kell, hogy tegye a Téma is - és megfordítva. Ezért az MVD-t csak olyan egyeden lehet értelmezni, ami legalább három tulajdonságot tartalmaz.

A többértékű függés nem más, mint generalizált funkcionális függés. Vagyis az FD az MVD olyan speciális esete, amelyben a függő értékészlet egyetlen tételre korlátozódik. Mivel az MVD mindig páros, annak jelölését így is szokták rövidíteni:  $A \twoheadrightarrow B \mid C$ .

Kiegészítés: Ezt a példát a szakirodalom tévesen mutatja be. Ha egy tanár nem oktathat többféle kurzuson és egy téma nem szerepelhet több tanfolyamon - a példa ezt sugallja -, akkor a Tanár és a Téma funkcionálisan meghatározza a tanfolyamot. Ebben az esetben pedig egyáltalán nincs szükség az MVD-re. Tehát a továbbiakhoz azt kell feltételeznünk, hogy vannak olyan tárgyak, amelyeket több kurzuson is oktatnak és olyan tanárok, akik több kurzuson is tanítanak. Ellenkező esetben a reláció nem lenne csupakulcs.

## 9.6 Negyedik normálforma

+

Az E egyedtípus 4NF alakban van akkor és csak akkor, ha az abban lévő bármilyen  $A \twoheadrightarrow B$  MVD egyben  $A \rightarrow B$  FD is.

KURZUS TANÁRAI

<i>Kurzus</i>	<i>Tanár</i>
<i>Fizika</i>	<i>Zöld</i>
<i>Fizika</i>	<i>Barna</i>
<i>Fizika</i>	<i>Fekete</i>
<i>Matek</i>	<i>Fehér</i>

KURZUS TÉMÁI

<i>Kurzus</i>	<i>Téma</i>
<i>Fizika</i>	<i>Optika</i>
<i>Fizika</i>	<i>Mechanika</i>
<i>Matek</i>	<i>Algebra</i>
<i>Matek</i>	<i>Geometria</i>

9.6 ábra: Átalakított TANFOLYAM modellrész

Magyarázat: Az eredeti egyedben két MVD-t fedezhetünk fel. A Kurzus többértékűen meghatározza a Tanárt és attól többértékűen függ a Téma is. Ha kiválasztunk egy Kurzus értéket, akkor meg tudjuk határozni a Tanárok *készletét*, függetlenül a Téma értékétől. Ugyanez vonatkozik fordítva a Témák alhalmazára is. Az MVD karbantartási anomáliát okoz. Ezért a TANFOLYAM egyedet megbontjuk a két MVD mentén, vagyis a többértékű meghatározó és függő párosokat külön egyedtípusokba emeljük ki.

Mivel a 9.5 ábrában a Kurzus  $\twoheadrightarrow$  Tanár MVD nem volt FD is, a TANFOLYAM egyed csak BCNF alakú volt. A 9.6 ábra két egyede viszont már 4NF alakú, mert olyan csupakulcs relációk, amelyekben már nincs többértékű függés. (NB.: A kétszlopos táblák elméletileg sem mutathatnak rossz függési viszonyokat, ezért azokat már nem is szoktuk tovább elemezni.) Az olvasó meggyőződhet arról, hogy a 4NF esetében nem lépnek már fel az alacsonyabb normálalakra jellemző karbantartási anomáliák.

Probléma: A negyedik normálformához számos elméleti és gyakorlati félreértés tapad. Ezek közül néhányat ismertetünk alább.

**Változatok.** A TANFOLYAM egyednek háromféle megbontása is létezhet. Az  $A \rightarrow B$ ,  $B \rightarrow C$  és  $A \rightarrow C$  függések fennállásakor az (A,B) és (B,C) páros adja a jó lebontást. Az (A,B) és (A,C) páros szerinti dekompozíció karbantartási gondokat okoz (vö. 8.3 példa). Ezért egyes szakírók

szerint a 9.6 ábra megoldásánál jobb lebontás a (Kurzus+Tanár) és (Tanár+Téma) vagy a (Kurzus+Téma) és (Téma+Tanár) páros. Ez a vélekedés téves. Pont ezek az eredmények vezetnek karbantartási anomáliára. Például ha beviszünk egy új tételt a Téma+Tanár relációba, akkor be kellene vinnünk a kurzuson tanított összes témához is az adott tanárt. Ezt a korlátot viszont a másik reláció (Kurzus+Téma) tartalmazza. Ezért jelen esetben csak a fenti - vagyis a függések menti - megbontás a jó.

Vegyük észre, hogy igen kivételes esettel állunk szemben. Általában a megbontásokkal hierarchikus szerkezetet kapunk. Itt a megbontás hálóra vezetett, mégis a speciális korlát miatt a két egyednek a Kurzuson való összekapcsolása nem okoz szemantikai problémát.

**Alkalmazás.** Az MVD-t csak csupakulcs táblán kell vizsgálni. Meglepő, hogy egyesek elfeledkeznek erről a szabályról. Tegyük fel, hogy a TANFOLYAM-ban szerepel egy leíró tulajdonság is: Óraszám. Mármint két eset lehetséges. Ha ennek meghatározásához mind a három tétel szükséges (minden kurzuson minden tanár ugyanolyan órázámban tanítja az adott témát), akkor az egyed nem bontható meg. Ha viszont az X és Y tanárok más órázámban okítják ugyanazt a témát, akkor az Óraszám részlegesen függ a kulcstól. Tehát azt ki kell emelni az eredeti relációból, ami után az csupakulcs relációvá válik.

**Tévedés.** Sokan elfeledkeznek arról a tételről, miszerint az MVD mindig „párosával jár” és ott is alkalmazzák, ahol az nem alternáló jellegű. Gyakori hiba, hogy az  $A \twoheadrightarrow B$  funkcionális függetlenséget  $B \twoheadrightarrow A$  (fordított irányú) MVD-nek értékelik. Mert ha az A nem határozza meg a B-t, akkor a B-hez több A érték, az A értékkészlete tartozik. Az MVD definíciójának ( $\leftarrow$  *Többértékű függés*) arról a kitételéről, hogy „a C-től függetlenül” egyszerűen elfeledkeznek. Ha a TANFOLYAM példában nem lenne igaz, hogy minden a kurzuson résztvevő tanár a tanfolyam minden témáját tanítja, akkor is fennállna a Kurzus  $\twoheadrightarrow$  Tanár függetlenség, de szó sem volna a Kurzus  $\twoheadrightarrow$  Tanár és a Kurzus  $\twoheadrightarrow$  Téma MVD-párosról. Nem lépne fel karbantartási anomália és a TANFOLYAM egyedet nem kellene, nem is lehetne megbontani.

**Egyszerűbben.** A 4NF alakot elméleti zsonglörködésnek tartjuk. A 9.5 ábra egyedében explicitté tehetjük az ismétlődéseket a következő módon:

TANFOLYAM		
Kurzus	Tanár	Téma
Fizika	Zöld	Mechanika
	Barna	Optika
	Fekete	
Matek	Fehér	Algebra
		Geometria

9.7 ábra: Ismétlődést tartalmazó TANFOLYAM egyed

Az ilyen táblát úgy bontjuk meg, hogy a független részt levágjuk a kulccsal együtt. Ekkor a KURZUS TANÁRAI (*Kurzus+Tanár*) és a KURZUS TÉMÁI (*Kurzus+Téma*) egyedeket kapjuk meg éppen úgy, mint a 9.6 ábrában. Az eredmény mégis jobb annál. Az ismétlések levágása után megmarad az eredeti egyed is annak kulcsával. Tehát ennél a normalizálásnál lesz egy TANFOLYAM (*Kurzus*, ...) egyedünk is, ami a másik kettő hierarchikus fölérendeltje. Nem vitás, hogy ez a megoldás elvileg és gyakorlatilag is jobb, mint a 4NF dekompozíció.

## 9.7 Kapcsolásfüggés

+

Az E egyedtípus akkor és csak akkor tesz eleget a kapcsolásfüggésnek, ha X, Y, ..., Z kivetítéseinek az összekapcsolásával helyreállítható úgy, hogy X, Y, ..., Z az E egyed tulajdonságainak a részhalmazai.

SZÁLLÍTÁS

SSZ	PSZ	CSZ
S1	P1	C2
S1	P2	C1
S2	P1	C1
S1	P1	C1

9.8 ábra: SZÁLLÍTÁSI modellrészlet

Magyarázat: Az eléggé kacifántos példát ismét Date úrtól vettük [10], aki azt maga is „patologikus”-nak nevezte. Vannak szállítók, projektek és cikkek. Ezek értékkészletei körkörösén egymáshoz kötöttek. Egy szállító meghatározott cikkeket adott projektekre szállít. Ha egy cikket szállít, akkor azt minden a cikket érintő projektre biztosítja és megfordítva: ha egy projekthez kapcsolódik, akkor annak minden cikkét szállítja. Egy cikket meghatározott szállító biztosít adott projektre. Ha a cikket egy szállító szállítja, akkor azt minden projektre teszi, amelyben a cikk szerepel és megfordítva. Ugyanígy a projekt szempontjából is egymáshoz kötődnek a szállítók és a cikkek.

A matematikai formula így hangzik:

ha megjelenik a táblában az  $\langle S1, P1, C2 \rangle$ ,  $\langle S2, P1, C1 \rangle$  és a  $\langle S1, P2, C1 \rangle$  sor, akkor szerepelnie kell abban a  $\langle S1, P1, C1 \rangle$  sornak is.

Probléma: A SZÁLLÍTÁS csupakulcs egyed igen ravasz karbantartási bajokat rejteget. Tekintsük úgy, mintha az ábra csak az első két sort tartalmazná. Most próbáljuk meg beilleszteni a harmadik sort. Ekkor azzal együtt a negyediket is be kell vinni. Az első szállító szállít a P1 projektre (első sor) és szállítja a C1 cikket (második sor). Azonban a beillesztésig a P1-C1 páros nem szerepelt az első két sorban. Vagyis nem kötöttük ki, hogy a P1 projektre a C1 cikket is szállítják. Az S2-P1-C1 sor bevitelével ezt megtesszük. Mivel az S1 szállító részt vesz a P1 projekt ellátásában, arra a C1 cikket is szállítania kell. Tehát a harmadik sor bevitele magával vonja a negyedikét is, viszont a negyedik beadása nem követeli meg a harmadikét is. A törlésekkel is gond van. A harmadik sor önmagában törölhető, mert nem veszik el az az ismeret, hogy a P1 projekthez tartozik a C1 cikk. Viszont a negyedik sor törlése együtt jár másik sor megszüntetésével is. (Melyikével?)

Megoldás: A karbantartási anomáliák miatt a SZÁLLÍTÁS egyedeket meg kell bontani, bár az 4NF alakban van. Ezt három módon tehetnénk: Szállító-Cikk/Cikk-Projekt, Szállító-Cikk/Projekt-Szállító és Cikk-Projekt/Projekt-Szállító egyedpárosokat alkotva.

SP	PC	CS						
<table><tr><td>SSZ</td><td>PSZ</td></tr></table>	SSZ	PSZ	<table><tr><td>PSZ</td><td>CSZ</td></tr></table>	PSZ	CSZ	<table><tr><td>CSZ</td><td>SSZ</td></tr></table>	CSZ	SSZ
SSZ	PSZ							
PSZ	CSZ							
CSZ	SSZ							

<i>SI</i>	<i>P1</i>	<i>P1</i>	<i>C2</i>	<i>C2</i>	<i>SI</i>
<i>SI</i>	<i>P2</i>	<i>P2</i>	<i>C1</i>	<i>C1</i>	<i>SI</i>
<i>S2</i>	<i>P1</i>	<i>P1</i>	<i>C1</i>	<i>C1</i>	<i>S2</i>

SPC		
<i>SSZ</i>	<i>PSZ</i>	<i>CSZ</i>
<i>SI</i>	<i>P1</i>	<i>C2</i>
<i>SI</i>	<i>P2</i>	<i>C1</i>
<i>S2</i>	<i>P1</i>	<i>C1</i>
<i>S2</i>	<i>P1</i>	<i>C2</i>
<i>SI</i>	<i>P1</i>	<i>C1</i>

9.9 ábra: Az első két kivetített egyed összekapcsolásának az eredménye

Probléma: Tegyük fel, hogy a fejlesztő az SP és PC párosokra bontja le a relációt, tehát a CS táblát nem alkalmazza. Vegyük tehát egyelőre úgy, mintha az nem lenne az ábrán. Most kapcsoljuk össze az SP és a PC relációkat. Ezzel az SPC relációt kapjuk, ami nem azonos az eredeti táblával, amiben nem szerepelt az itt kiemelten mutatott sor.

Megoldás: Csak úgy jutunk vissza a volt táblához, ha az SPC relációt összekapcsoljuk a CS-sel. Mivel abban nincs S2-C2 páros, az összekapcsolással az SPC-ből ez a sor kiesik. Tehát az eredeti relációt nem csak két párosra kell lebontani, hanem mind a három párost alkalmazni kell. Emiatt a körkörös viszony és kettős összekapcsolás miatt nevezik ezt a függést **kapcsolásfüggésnek** [join dependency].

A kapcsolásfüggés rövid neve **JD**, jele pedig  $*$  (X, Y, ..., Z). Példánk esetében az SP, a PC illetve a CS a részhalmazok, és ezért fennáll a  $*$  (SP, PC, CS) kapcsolásfüggés.

## 9.8 Ötödik normálforma

+

**Az egyed akkor és csak akkor van 5NF alakban, ha a kulcsjelöltjei között nem áll fenn kapcsolásfüggés. (Ami azt is jelenti, hogy nincsen közöttük többértékű és funkcionális függés sem.)**

Magyarázat: Amint a többértékű függés (MVD) a funkcionális függés (FD) általános esete, úgy a kapcsolásfüggés (JD) az előző kettő általánosítása. Tehát az FD és az MVD a JD speciális esete. A SZÁLLÍTÁS egyedben nehezen áthidalható karbantartási problémák léptek fel. Ennek az az oka, hogy bár az egyedben nincs MVD és FD, viszont a lehetséges három kulcsjelölt páros körkörös kapcsolásfüggésben áll egymással.

**9.5 példa** RENDELÉS (**Rendelésszám**, Vevőkód, Vevőnév)

VEVŐ (**Vevőkód**, Vevőnév)

RENDELÉS-1 (**Rendelésszám**, Vevőkód)

RENDELÉS-2 (**Rendelésszám**, Vevőkód)

### RENDELÉS-3 (**Rendelésszám**, Vevőnév)

Magyarázat: Itt a 8.3 példát ismételtük meg. Az eredeti RENDELÉS egyedet kétféle módon bontjuk meg. Az eredeti egyedben létezett az a \* (Vevőkód, Vevőnév) JD, amely ekkor persze funkcionális függést (Vevőkód  $\rightarrow$  Vevőnév) jelentett. Az első megbontással ez a kapcsolásfüggés nem veszik el, az tehát jó. A második verzióban viszont nem annak mentén bontottuk le az egyedet, a függés elveszett, tehát az a megoldás rossz.

#### 9.6 példa VEVŐ (Vevőkód, Vevőnév, Vevőcím, Vevőtípus)

\* (Vevőkód, Vevőnév, Vevőtípus) - (Vevőkód, Vevőcím)

\* (Vevőkód, Vevőnév) - (Vevőkód, Vevőtípus) - (Vevőnév, Vevőcím)

Magyarázat: Tegyük fel, hogy a vevőnevek egyediek. Ekkor az eredeti egyedben több JD fedezhető fel a két kulcsjelölt (Vevőkód, Vevőnév) miatt. Most mindkét lebontás jó, mert mindkettőben csak a volt kulcsjelölteken alapulnak a kapcsolásfüggések. Ezért mindkét esetben a két illetve három eredményegyed összekapcsolásával visszanyerjük az eredetit.

Kiegészítés: Az ötödik normálformát (5NF) kivetítés-összekapcsolás [projection-join] alaknak is nevezik és ezért PJ/NF módon is jelölik. Bizonyítható [<sup>14</sup>], hogy matematikai értelemben ennél jobb alak nem létezhet. Ez világos, hiszen az eredményrelációkban vagy csak a kulcsjelöltektől függenek a leírók, vagy eleve csupakulcs relációkról van szó. Ezért ne is várja az olvasó, hogy találkozhat majd 6NF, 7NF stb. alakokkal: az egyeden belüli tulajdonságfüggéseknek nincs olyan magasabb formája, amelynek a JD lenne a speciális válfaja. Ezért addig, amíg a normalizálás lényegét kizárólag csak a veszteségmentes, tehát összekapcsolással visszafordítható dekompozícióban keressük, az 5NF valóban a végső, a legtokéletesebb alak. Ennek dacára egyben a legtöbb problémával is jár.

Probléma: A 9.9 ábra három egyedpárosa már nem mutatja a SZÁLLÍTÁS frissítési anomáliáit. Azonban fellép egy karbantartási gond. Bármelyik egyedet is aktualizáljuk, másikhoz is hozzá kell nyúlni. A származtatott egyedek egymástól nem függetlenek, mert az eredeti egyed atomi volt. Erre pedig azt mondtuk, hogy adott esetben egyáltalán nem célszerű megbontani ( $\leftarrow$  A normalizálás negyedik lépése). Mivel a SZÁLLÍTÁS egyed megbontásával nyert egyedek egymás keresztkorlátaiként lépnek fel, ebben az esetben a normalizálással nagyobb gondot okozunk magunknak, mintha azt elhagyjuk.

Következtetés: A gyakorlatban igen ritkán fordulnak elő a 9.8 ábra példájához hasonló „beteges” helyzetetek. Ezeket nagyon nehéz felfedezni. A megbontás nem mindig sikerül jól (lásd a 9.5 példa második megoldását). Ha mégis, akkor vagy nincs értelme (9.6 példa) az alternáló kulcsok miatt, vagy a dekompozícióval az eredetinél is több bajt generálunk magunknak. Szumma-szummárum: Az 5NF dekompozíciónak nincs *gyakorlati* értelme, az olvasó máris elfeledkezhet róla. Két ok miatt foglalkoztunk mégis vele. Részben azért, hogy az előbbi mondat állítását igazoljuk. Részben azért, hogy ne érhesse bennünket az a vád, hogy a normalizálás *elméleti* háttérét nem mutattuk be a maga teljességében.

## 9.9 Ellenőrző kérdések

- 9/1 Az alábbi kijelentések közül melyik igaz (I) és melyik hamis (H)?
- Minden kétszlopos egyed eleve 3NF alakban van.
  - Minden kétszlopos egyed eleve BCNF alakban van.
  - Minden kétszlopos, egyetlen funkcionális függést mutató egyed 5NF alakú.
  - Minden kétszlopos egyed 5NF alakú.
- 9/2 Igaz-e (I) az az állítás, vagy hamis (H), amely szerint az  $E(A,B,C)$  egyed csak akkor egyenlő az  $E_1(A, B)$  és  $E_2(A, C)$  lebontásainak az összekapcsolásával, ha fennáll az  $A \rightarrow B$  függés?
- 9/3 Igaz-e (I) vagy hamis (H) a következő négy kijelentés:
- Ha egy egyedben  $A \rightarrow B$  és  $B \rightarrow C$ , akkor  $A \rightarrow C$ .
  - Ha egy egyedben  $A \rightarrow B$  és  $A \rightarrow C$ , akkor  $A \rightarrow \{B, C\}$ .
  - Ha egy egyedben  $B \rightarrow A$  és  $C \rightarrow A$ , akkor  $\{B, C\} \rightarrow A$ .
  - Ha egy egyedben  $A \rightarrow B$  és  $A \rightarrow C$ , akkor  $\{B, C\} \rightarrow A$ .
- 9/4 Igazak-e (I) vagy hamisak (H) a következő kitételek:
- A kulcstörő függést tartalmazó egyed mindig BCNF alakra hozható.
  - Az ilyen egyed mindig függetlenül karbantartható egyedekre bontható.
  - Az ilyen egyed mindig összekapcsolható egyedekre bontható.
- 9/5 Versenybridzset játszunk több asztalon. Vannak leosztások. Egy leosztásban több csapat vesz részt több asztalon. Minden leosztásban mindegyik csapat mindegyik asztalon részt vesz, amelyik asztalon játszik. Hányadik normálformában van a következő két egyed és mi velük a teendő:

BRIDZS (Leosztás, Csapat, Asztal)

BRIDZS (Leosztás, Csapat, Asztal, Eredmény)

## 10. SZEMANTIKUS NORMALIZÁLÁS

### 10.1 A csoportok csapdái

A hagyományos normalizálásban nem eléggé figyelnek a triviális függésre, vagyis arra, hogy az összetett kulcs mint csoport meghatározza saját tagjait. Ezért nem veszik észre, hogy a részleges és a tranzitív függés lényege azonos (10.2 *Függéstáblázat*). Viszont a külső kulcstörő függés természete teljesen eltér az előző kettőtől annak dacára, hogy ez az eset is az  $A \rightarrow B$ ,  $B \rightarrow C$  és  $A \rightarrow C$  függéshármas képletsorával írható le.

Ha módszeresen - azaz kombinatorikusan - megvizsgáljuk, hogy a három függés közül melyik „normális” és melyik triviális, akkor összesen nyolc esettel kell számolnunk. Ezek közül az egyéb szakirodalom csak hármat tárgyal, kettő esetében nincs semmi teendő, ám még mindig nyitva marad három igen érdekes kombináció (10.3 *Speciális függések*).

Ezekre az jellemző, hogy a meghatározó és/vagy a meghatározott csoportot alkot. A jelleg szerint másként és másféle problémák lépnek fel e kombinációk esetén. Azonban közös bennük, hogy egyik gond sem szüntethető meg az eddig tárgyalt dekompozíciós eljárással (10.4 *Belső kulcstörő függés*, 10.5 *Metszefüggés* és 10.6 *Csoportfüggés*).

*Ennek a fejezetnek két célja van. Az első az, hogy átformálja a tervezők normalizálási gondolkodásmódját. A másik az, hogy ismertesse a tulajdonságok közötti függések olyan speciális eseteit, amelyek a csoportokkal kapcsolatosak és amelyek kizárólag csak szemantikus normalizálással közelíthetők meg.*

Mit is jelent a „szemantikus normalizálás”? Az  $A \rightarrow B$ ,  $B \rightarrow C$  és  $A \rightarrow C$  függéssorban a legtöbb olvasó a harmadikat tartja következménynek és a RENDELÉS (**Rendelésszám**, Vevőkód, Vevőnév) egyedből annak alapján bizonyára a Vevőnevet fogja kiemelni. Pedig nem mindig helyes, ha csak matematikai alapon - azaz mennyiségekben - gondolkodunk. A szemantikus normalizálás is épít a matematikára. Ámde a fenti függéssor esetében nem azonnal a dekompozíciót célozza meg. A tervező először azt a kérdést teszi fel magának, hogy a  $B \rightarrow C$  és  $A \rightarrow C$  függésekben a C valóban azonos lényegre takar-e? Igen sokszor előfordul, hogy nem. Ekkor pedig szemantikai megoldásra - átértelmezésre - van szükség. Meg kell tanulni ebből a szempontból is értékelni modellünket (10.7 *Új szemlélet*).

A normalizálást az irodalom logikai szintű adatbázistervezési részmódszernek tekinti. Olyannak, ami egymástól független egyedtípusok dekompozíciójára szolgál. Emiatt nem figyel a sok galibát okozó csoportokra. A mi szemléletünkben a normalizálás olyan matematikai részmódszerrel támogatott fogalmi szintű adatmodellezési eljárás, amelyben a végső megoldáshoz sokszor éppen csak a fogalmak tisztázásával lehet eljutni. Egy külön tárgyalt összetett eset (10.8 *Pszudo-tranzitivitás*) mindennél jobban bizonyítja, hogy az egyedeket nem lehet egyenként normalizálni: a normalizálás tárgya maga a teljes modell.



## 10.2 Függéstáblázat

**10.1 példa** RENDELÉS (**Rendelésszám**, *Vevőkód*, *Vevőnév*)

RENDELÉSTÉTEL (**Rendelésszám+Cikkszám**, *Cikknév*)

RENDELÉS

RENDELÉSTÉTEL

Rendelésszám → Vevőkód

Vevőkód → Vevőnév

Rendelésszám → Vevőnév

Rendelésszám+Cikkszám → Cikkszám

Cikkszám → Cikknév

Rendelésszám+Cikkszám → Cikknév

Magyarázat: A RENDELÉS egyed tranzitív, a RENDELÉSTÉTEL részleges függést mutat. Ebből következően csak második ill. első normálformában vannak. A két esetben pontosan ugyanazok a tárolási és karbantartási anomáliák lépnek fel. Még a függések sorozata is azonos mintát mutat, ha a RENDELÉSTÉTEL-nél felvesszük a sorba az ún. *triviális függést* is. (Armstrong [9] *projektivitási* axiómája szerint az összetett kulcs - Rendelésszám+Cikkszám - mindig meghatározza a saját összetevőit - Cikkszám. Ez az ún. triviális függés.) A két fenti képletsor csak annyiban tér el egymástól, hogy az első függés baloldalt „normális”, viszont jobboldalt - a kiemelt részben - triviális.

Probléma: Ha a tényezők kulcsrész/nem-kulcsrész szerepe, vagyis normális (Vevőkód) illetve triviális (Cikkszám) függési módja közömbös, akkor a részleges és tranzitív függés, illetve az 1NF és a 2NF probléma között nincs különbség. Tehát felesleges két különböző esetben gondolkodni. Ha viszont a *függési mód* nem közömbös, akkor a szakírók nem gondolták át a függések összes lehetséges esetét [15].

		1	2	3	4	5	6	7	8
A	→ B	T	T	T	T	N	N	N	N
B	→ C	T	T	N	N	T	T	N	N
A	→ C	T	N	T	N	T	N	T	N
Tranzitív függés									X
Részleges függés									
Külső kulcsörés									X

10.1 ábra: A függések döntési táblázata (1)

Magyarázat: A döntési táblázat négy részből áll. A baloldali felső részben (feltételtörzs) három tulajdonság képletsorát mutatjuk. Ez a szokásos tranzitivitási képlet. A jobboldali felső részben (feltételjegyzék) N betűvel jelöltük a normális, T betűvel a triviális függést. A baloldali alsó részben (tevékenységtörzs) a lehetséges következmények találhatók. A jobboldali alsó rész (tevékenységjegyzék) X bejegyzéssel jelzi, hogy a döntési szabályok közül melyik következik be. A jobboldali teljes oszlopokat szabályoknak nevezzük.

A táblázatot így kell olvasni: Ha az  $A \rightarrow B$  függés normális (N) és a  $B \rightarrow C$  függés normális és az  $A \rightarrow C$  függés normális, akkor tranzitív függés áll fenn. Ezt mutatja a táblázat kiemelt nyolcadik oszlopa. Hasonlóképpen leolvasható a táblából a részleges függés esete (negyedik

oszlop), amelyben az első függés triviális (T). A tábla hetedik szabálya fedi le a kulcstörő függés esetét. Ennek képlete az alábbi (vö. 9.1 ábra):

$$\begin{array}{ll} A \{X+Y\} \rightarrow B \{Z\} & \text{Diák+Tárgy} \rightarrow \text{Tanár} \\ B \{Z\} \rightarrow C \{X\} & \text{Tanár} \rightarrow \text{Tárgy} \\ A \{X+Y\} \rightarrow C \{X\} & \text{Diák+Tárgy} \rightarrow \text{Tárgy} \end{array}$$

Magyarázat: A tranzitív és részleges függések *konkrét* képletét a 10.1 példa mutatta a rendelések és rendeltetések vonatkozásában. Az *általános* - azaz tartalomtól független - képletekben  $A\{X+Y\}$  módon jelezzük, hogy az A milyen tagokból összetett. Az elemi tényezőknél is a  $C\{X\}$  jellegű jelölést alkalmazzuk. Ennek két oka van. Egyrészt így lehet látni, hogy a C az A része. Másrészt az X maga is lehet összetett - pl.  $X\{D+E\}$  - , de azt nem bontjuk ki, ha a részeire nézve nincs mondanivalónk. Pl. a tárgy állhat kurzusból és témából, de e két tényezőre nem volt szükségünk a TANÍTÁS-példa esetében.

**10.2 példa** ORSZÁG (Országkód, ...)  
KÖRZET (Országkód+Körzetkód, ...)  
TELEPÜLÉS (Országkód+Körzetkód+Településkód, ...)

$$\begin{array}{ll} A \{X+Y+Z\} \rightarrow B \{X+Y\} \\ B \{X+Y\} \rightarrow C \{X\} \\ A \{X+Y+Z\} \rightarrow C \{X\} \end{array}$$

Magyarázat: Előfordul, hogy a kulcsok hierarchikusan, egymásból épülnek fel. Ekkor természetesen az összetettebb kulcs mindig triviálisan meghatározza a kevésbé összetettet, hiszen az a részét képezi. A települések kulcsának része a körzet azonosítója, annak pedig része az ország kulcsa. Általános képlettel:  $A\{X+Y+Z\} \rightarrow B\{X+Y\}$  és  $B\{X+Y\} \rightarrow C\{X\}$ . Világos, hogy a település kulcsának része az ország azonosítója is:  $A\{X+Y+Z\} \rightarrow C\{X\}$ .

Kiegészítés: Az itteni példa képletét felfedezhetjük függéstáblánkban is. Az első szabály az **abszolút triviális függés** esetét mutatja. Az A a saját részeként meghatározza a B-t, ami szintén triviálisan meghatározza a C-t, ami az A-nak is része. Ebből viszont már az is következik, hogy a második szabály **ellentmondásos**. Az  $A\{X+Y+Z\} \supset B\{X+Y\} \supset C\{X\}$  halmazösszefüggésekből adódik az  $A\{X+Y+Z\} \supset C\{X\}$  viszony. Ezért kizárt, hogy a C ne legyen az A része, vagyis azon normális - és ne triviális - függéssel függjön.

### 10.3 Speciális függések

		1	2	3	4	5	6	7	8
A	→	B	T	T	T	N	N	N	N
B	→	C	T	T	N	N	T	T	N
A	→	C	T	N	T	N	T	N	N

Triviális függés	X		
Ellentmondás	X		
Tranzitív függés			X
Részleges függés		X	
Külső kulcstörés			X

10.2 ábra: A függések döntési táblázata (2)

Magyarázat: Táblánkat kiegészítettük a triviális függés és az ellentmondás szabályaival. Tudjuk, hogy tranzitív és részleges függés esetén (8. és 4. szabályok) a harmadik  $A \rightarrow C$  függés alapján az A kulcsú egyedből ki kell emelni a C redundáns leíró tulajdonságot. A RENDELÉS táblában a Rendelésszám  $\rightarrow$  Vevőnév függés rossz, ezért onnan kiemeljük az utóbbi tételt. Ellentmondásos függés (2. szabály) nem fordulhat elő, ezért azzal az esettel nem kell foglalkozni. Abszolút triviális függés (1. szabály) esetén nincs sem szükség, sem lehetőség a normalizálásra. A TELEPÜLÉS (*Országkód+Körzetkód+Településkód*, ...) egyedből a harmadik Országkód+Körzetkód+Településkód  $\rightarrow$  Településkód tranzitív függésre alapozva nem vehető ki a Településkód, mert hiszen az a kulcs része. Eddig a pontig a függéseken alapuló normalizálás törvényei egyszerűek.

Probléma: Táblázatunkban van még három kitöltetlen szabály. Ezeket fogják kifejteni a következő pontok. Itt egy általánosabb jelenségre kell felhívni a figyelmet. Táblánk 7. oszlopa mutatja a külső kulcstörés esetét. Az idevágó példában a Diák+Tárgy  $\rightarrow$  Tanár, Tanár  $\rightarrow$  Tárgy és Diák+Tárgy  $\rightarrow$  Tárgy függéssorozat harmadik függése triviális. Ezért az nem szüntethető meg úgy, mint a tranzitív illetve a részleges függés esetében. A másik két függés pedig azért nem számolható fel, mert akkor ismeretet veszítenénk.

Kulcstörő függések esetén már nem alkalmazhatjuk azt a mechanikus normalizálást, ami a tranzitivitási képleten alapul ( $\leftarrow$  *A normalizálás negyedik lépése*). A *matematikai* összefüggések alapján vesszük észre - éppen táblázatunk 7. szabálya alapján - azt, hogy Boyce-Codd normálforma megoldásra *lenne* szükség. Az nem biztos, hogy az egyedet egyáltalán meg kell bontani. Az viszont biztos, hogy nem a tranzitivitási szabály szerint fogjuk azt átalakítani, ha mégis alkalmazunk valamilyen dekompozíciót. Itt kezdődik a *szemantikus* normalizálás, aminek során a tervező a jelenségek természetes viszonyait mérlegelve alakítja át a szerkezetet, aminek hibáját a mechanikus normalizálás tárta fel. Ez a kettősség - matematikai alapú feltárás, szemantikai alapú megoldás - jellemzi majd a következő pontokban bemutatott eseteket is.

## 10.4 Belső kulcstörő függés

+

Az E egyed belső kulcstörő függést tartalmaz akkor és csak akkor, ha az A+B azonosító összetett és annak egyik része meghatározza a másik összetevőt.

10.3 példa RENDELÉS-1 (Rendelésszám, Cikkszám, ...)  
RENDELÉS-2 (Rendelésszám+Cikkszám, ...)

Magyarázat: Mi nemcsak az egyeden belüli (attribútum), hanem az egyedek közötti (doméjn) függéseket is számbavesszük. Természetesen egy *egyeden belül* nem fordulhat elő, hogy a Rendelésszám és a Cikkszám egymástól független is (amint azt a második egyed kulcsa sejteti), meg egymástól függ is (amint azt az első egyed mutatja). Azonban két *egyed között* előfordulhat ez a szituáció. A két rendelés egyed külön-külön akár jó is lehetne, de együttesen már látszik ellentmondásos és redundáns voltak. Ezt a problémát a következő képletsorral írhatjuk le (lásd a döntési táblázat 3. szabályát):

$$\begin{array}{ll} A \{X+Y\} \rightarrow B \{X\} & \text{Rendelésszám} + \text{Cikkszám} \rightarrow \text{Rendelésszám} \\ B \{X\} \rightarrow C \{Y\} & \text{Rendelésszám} \rightarrow \text{Cikkszám} \\ A \{X+Y\} \rightarrow C \{Y\} & \text{Rendelésszám} + \text{Cikkszám} \rightarrow \text{Cikkszám} \end{array}$$

Magyarázat: Van egy A összetett kulcsunk, ami triviálisan meghatározza a saját első X részét. Az X kulcsrész meghatározza a kulcs másik Y tagját, ami szintén triviálisan függ az azonosító egészétől. A 10.3 példa tehát belső kulcstörést mutat.

Kérdés: Miért baj az, ha a modellben van belső kulcstörési jelenség? Nem létezhet abban két rendelés egyed a fenti módon, hiszen azok még egymáshoz is kapcsolhatók?

Válasz: Egyrészt a Rendelésszám és a Cikkszám sajátos ciklikus kapcsolatban áll, ami mellesleg redundanciát is okoz. Fellépnek a szokásos karbantartási anomáliák. Átírjuk az egyik egyedben az adott rendelésszámhoz tartozó cikkszámot, de ezt nem tesszük meg a másik egyedben - és máris előáll az integritási hiba. Azonban nem ez a valódi probléma. A súlyosabb gond az, hogy maguk a jelenségek nem kerülnek egyértelmű azonosításra.

A RENDELÉS-1 egyed egytétéles rendeléseket sejtet. A RENDELÉS-2 többtétélesekre enged következtetni. Mármint a *valóságban* a rendelések vagy ilyenek, vagy olyanok. Ha létezik egyetlen többtétéles rendelés is, akkor a Rendelésszám és a Cikkszám egymástól funkcionálisan független és már nem lehet egytétéles rendelésben gondolkodni. A *modell* mindig az általános viszonyokat tükrözi, azok pedig csak egyfélék lehetnek. Tehát a fenti példa ellentmondást tartalmaz, amit a belső kulcstörés képlete alapján fedeztünk fel.

Megoldás: Mivel a harmadik függés triviális, a Cikkszám nem vehető ki az összetett azonosítóból. Csak a második függés normális. Ha azt felszámoljuk, az egyet jelent azzal, hogy megszüntetjük a RENDELÉS-1 egyedet. Abban a Cikkszám és a Rendelésszám viszonya csak 1:N fokú, míg a másik egyedben M:N fokú (az összetett kulcs részei mindig szükségszerűen függetlenek egymástól). A modellezésben pedig mindig az általánosabb viszonyt kell alapul venni (az 1:N fok az M:N fok speciális esete). Ezért a modellezőnek az a döntése, hogy megszünteti a RENDELÉS-1 egyedet, számszerűleg helyesnek tűnhet.

Probléma: Amikor nem a harmadik függés alapján döntünk, akkor már nem az eredeti dekompozíciós normalizálási eljárást alkalmazzuk. A példában mutatott ellentmondás és redundancia már nem oldható fel *matematikai* alapon. Szükség van a *szemantikai* normalizálásra. Mit jelent ez az adott esetben? A Cikkszám redundanciája nem feltétlenül valós ( $\leftarrow$  *nyílt logikai átfedés*): az takarhat homonimát is ( $\leftarrow$  *látszólagos logikai átfedés*). Márpedig ha ahelyett, hogy korrekt neveket felszámoljuk a homonimát, azt egyszerűen kiírtjuk, akkor ismeretet veszíthetünk.

**10.4 példa** VEVŐRENDELÉS (Vevő rendelésszám, Cikkszám, ...)  
SZÁLLÍTÓRENDELÉS (Szállító rendelésszám+Cikkszám, ...)

Magyarázat: Vannak vevői rendeléseink, amik egytételesek (pl. egy hajógyárban) és ezért a Rendelésszám  $\rightarrow$  Cikkszám függés érvényesül bennük (RENDELÉS-1). Viszont léteznek szállítói rendeléseink is, amik többtételesek. A tételeket a Rendelésszám és a Cikkszám párosa azonosítja és így a két adat egymástól független (RENDELÉS-2). Ezért *nem ugyanarról* a Rendelésszámról van szó, tehát ez a tétel homonima. Egyáltalán nincs szükség dekompozícióra. A fogalmak tisztázása után csak a neveket kell rendbe tenni. (NB.: Ha két egyed kulcsa eltérő, akkor sejthető, hogy az adott egyedek lényege sem azonos. Már innen is tudhattuk volna, hogy a nevekkel volt baj.)

Kiegészítés: Az azonosítót nem csak leíró tulajdonság törheti meg ( $\leftarrow$  *külső kulcstörő függés*), hanem - amint látjuk - az azonosító belső része is. A két probléma feloldása teljesen eltérő természetű. Külső kulcstörés esetében mechanikusan is eljárhattunk úgy, hogy a BCNF normálformát alkalmaztuk. Itt viszont csak a szemantikus megoldás segít. Nem lehet és nem is szabad dekompozíciós eljárást alkalmazni.

Tanulság: A 10.3 példa két egyede *egyenként* 5NF alakban volt. *Együttesen* mégsem alkottak jó modellt, mint a 10.4 példa. A dekompozíció - mint neve is mutatja - a relációk megbontására irányul, azok integrálása nélkül. Márpedig az utóbbi is feladatunk, mert az optimumkritériumok nem külön-külön egyedekre, hanem a modell egészére vonatkoznak. Ezért leszögezhetjük, hogy az 5NF mechanikus keresése sohasem vezethet el az optimális adatmodellhez. Matematikai alapon fel kell tárnunk a belső kulcstörés esetét is, hogy azt valamilyen szemantikai alapon számoljuk fel.

## 10.5 Metszefüggés

+

Az E egyed metszefüggést tartalmaz akkor és csak akkor, ha az A+B összetett azonosító olyan csoportot határoz meg, ami tartalmazza az A vagy a B kulcsrészt.

**10.5 példa** RENDELÉSTÉTEL (*Rendelésszám+Cikkszám*, ..., Mennyiség)  
 DISZPOZÍCIÓTÉTEL (*Diszpozíciószám+Cikkszám*, *Rendelésszám*, ..., Mennyiség)

Magyarázat: Rendelések érkeznek hozzánk. Ezek többtételesek. Egy-egy rendeléstételt több részletben is szállíthatunk. Mondjuk 80 valamit kértek tőlünk és mi három (pl. 40, 25, 15) adagban teljesítünk. Ezért egy rendeléstételhez több diszpozícióétel is tartozhat. A diszpozíciónak mindig arra a rendelésre kell utalnia, amire maga a rendeléstétel is hivatkozik. Ezért szerepel a DISZPOZÍCIÓTÉTEL-ben a Rendelésszám. Ezzel szemben a diszpozíció - nem tétel, hanem a teljes szállítás - több rendelés tételeit is tartalmazhatja. Így nem áll fenn a Diszpozíciószám  $\rightarrow$  Rendelésszám függés és a fordítottja sem. Mivel a meghatározó tulajdonságok átfednek, a metszet függés elnevezés találó. Ugyanis az ilyen függéskombináció képlete a következő:

A {X+Y} $\rightarrow$ B {Z+Y}	Diszpozíciószám+Cikkszám $\rightarrow$ Rendelésszám+Cikkszám
B {Z+Y} $\rightarrow$ C {Y}	Rendelésszám+Cikkszám $\rightarrow$ Cikkszám, (Mennyiség)
A {X+Y} $\rightarrow$ C {Y}	Diszpozíciószám+Cikkszám $\rightarrow$ Cikkszám, (Mennyiség)

Probléma: A példa két gondot takar. Egyrészt a második egyed nem utal expliciten a Cikkszám+Rendelésszám összetételre, vagyis az első egyed kulcsára. Ezért nem látszik a két egyed közötti kapcsolat sem. Azért nem, mert a második egyedben a Rendelésszám *tetszőleges* értéket vehet fel, nincs korlátozva arra, hogy csak a Cikkszám+Rendelésszám összetételekben szereplő tartalmat kaphassa meg. Tehát a második egyedben nem tárható fel a fenti függéssorozat első tagja. Másrészt éppen ebből következik a második baj. Az első függés hiányában nem fedezhető fel a Mennyiség tranzitivitása.

Háttér: A relációs modell elismeri az *összetett azonosító* tételt, de nem engedi meg az *összetett leíró* tulajdonságot. Ezért a függés „baloldalára” írható  $A+B$  összetétel, de „jobboldalára” nem. Nem fejezhető ki például ez a függés:  $A+B \rightarrow C+D$ . Az additivitási axióma [<sup>9</sup>] csak azt mondja ki, hogy az  $A \rightarrow B$  és  $A \rightarrow C$  függések esetén mindig igaz az  $A \rightarrow \{B, C\}$  függés. Csakhogy a  $\{B, C\}$  szemantikailag nem azonos a  $B+C$ -vel! Míg az előző esetben a  $B$  és a  $C$  egymástól függetlenül tetszőleges értéket kaphat, addig az utóbbi esetben mindegyikük csak olyan értéket vehet fel, ami már szerepel egy meghatározott és korlátként szolgáló  $B+C$  párosban. Magyarul és egyszerűbben fogalmazva a diszpozíció tételeknek nem külön-külön kell utalniuk egy cikkre és egy rendelésre, hanem együttesen egy már meghatározott rendeltetésre kell mutatniuk.

**10.6 példa** RENDELÉSTÉTEL (*Rendelésszám+Cikkszám*, ..., Mennyiség)  
 DISZPOZÍCIÓTÉTEL (*Diszpozíciószám+Cikkszám*,  
*Rendelésszám+Cikkszám*, ..., Mennyiség)

Magyarázat: Most már minden egy fokkal világosabb. Az átrendezett példa alapján már felállítható a fenti függéssorozat. Az egyedekkel kapcsolatosan egy feladatunk biztosan akad, mert már látszik a Mennyiség normális tranzitivitása. Azt azonban egyáltalán nem szabad felszámolnunk, mert nem redundanciáról, hanem *homonimáról* van szó. Az első egyedben a Rendelt mennyiséget, a másodikban pedig a Diszponált mennyiséget kellene vezetni. Vagyis a két Mennyiség lényege nem azonos és a dekompozícióval csak ismeretet veszítenénk.

**10.7 példa** RENDELÉSTÉTEL (*Rtételek azonosító*, ..., Rendelt mennyiség)  
 DISZPOZÍCIÓTÉTEL (*Dtételek azonosító*, *Rtételek azonosító*, ..., Diszponált mennyiség)  
*Rtételek azonosító* {Rendelésszám+Cikkszám}  
*Dtételek azonosító* {Diszpozíció+Cikkszám}

Magyarázat: Ez a megoldás már tökéletes. Megszüntettük a homonimát és korrekt módon adtuk meg a két egyed kapcsolatát is. Fogalmi szinten az teljesen közömbös, hogy a kezelőrendszer képtelen a csoportok feldolgozására (lásd a két összetételt). A valóságot kell modelleznünk, azt pedig a fenti kép adja vissza élethűen.

Kiegészítés: A metszetfüggés esetét döntési táblánk 5. szabálya mutatja. Az első függés normális, a másik kettő triviális. Nincs mód dekompozícióra. A triviális függéseket nem lehet felszámolni, ha pedig ezt tennénk az első függéssel, akkor elvesztenénk a két egyed kapcsolatát. Viszont minden ilyen esetben a tervezőnek el kell gondolkodnia azon, hogy a metsző rész - esetünkben a Cikkszám - pontosan ugyanazt jelenti-e mindkét egyedben. Ha nem, akkor egyéb megoldást kell találnia.

Tanulság: Ismét csak oda jutottunk, hogy matematikai apparátussal fedeztünk fel egyes problémákat, de azokat szemantikai úton oldottuk meg. Tanulság az is, hogy az összetett meghatározottakat meg nem engedő relációs koncepció elméletileg sem alkalmas arra, hogy segítségével feltárjuk az optimumkritériumokat sértő hibákat. Helyes, hogy ma már egyes adatkezelők használatakor példánk két egyede között kapcsolatot lehet kijelölni. Az viszont helytelen, hogy ezt külön korlátként kell megtenni és a kapcsolat nem a két egyed közös *kapcsolótulajdonságának* az egyébként szokásos természetes logikáján alapul.

## 10.6 Csoportfüggés

+

**Az E egyed csoportfüggést tartalmaz akkor és csak akkor, ha az A+B csoport mellett expliciten tartalmazza annak A vagy B részét is.**

**10.8 példa** DISZPOZÍCIÓ (*Diszpozíciószám*, *Rtélet azonosító*, Cikkszám)  
Rtélet-azonosító {Rendelésszám+Cikkszám}

Magyarázat: Most azt feltételezzük, hogy minden rendeléstételt külön diszponálnak. Ezért a Diszpozíciószám folyamatos sorszám, vagyis nem csoport. Viszont a diszpozíció továbbra is a rendeléstételre kell, hogy mutasson. Ennek kulcsa impliciten tartalmazza a Cikkszámot, miközben az expliciten is megjelenik az egyedben. Az tehát csoportfüggést tartalmaz, aminek a képlete a következő:

A {X}	→ B {Z+Y}	Diszpozíciószám	→ Rendelésszám+Cikkszám
B {Z+Y}	→ C {Y}	Rendelésszám+Cikkszám	→ Cikkszám
A {X}	→ C {Y}	Diszpozíciószám	→ Cikkszám

Probléma: A képlet „transzitivitást”, azon keresztül redundanciát sejtet. Az egyedben két Cikkszám is szerepel. Két eset lehetséges. Ha a két tétel jelentése azonos, akkor valóban redundanciáról van szó és az explicit Cikkszámot ki kell írtani. Ha viszont nem azonos, akkor a homonimát a megfelelő nevekkkel kell kiküszöbölni.

**10.9 példa** DISZPOZÍCIÓ (*Diszpozíciószám*, *Rtélet azonosító*, Helyettesítő cikkszám)  
Rtélet-azonosító {Rendelésszám+Eredeti cikkszám}

Magyarázat: Ez a megoldás már tökéletes. A rendeléstételben szerepel egyféle cikk (az eredeti), amit nem tudunk szállítani, de pótoluk vele egyenértékessel (a helyettesítővel). A helyzetet ekkor a homonima kiküszöbölésével oldottuk meg. Figyelje meg az olvasó, hogy a Cikkszámot a csoportösszetételben is minősítettük. Ennek okaira majd kitérünk. Itt csak annyit kell megjegyezni, hogy ha egy tulajdonság nem maga a kulcs és többször szerepel egy egyedben, akkor mindegyik előfordulását minősíteni kell.

Kiegészítés: Döntési táblázatunk 6. szabálya világítja meg az esetet. A második függés triviális, a harmadik nem az. Ezért az utóbbit akár meg is lehetne szüntetni. Azonban ha mechanikusan járunk el, akkor ismeretet veszíthetünk (helyettesítés). Ezért a matematikai alapon feltárt problémát ismét csak szemantikai megoldással kell feloldani.

## 10.7 Új szemlélet

	1	2	3	4	5	6	7	8
$A \rightarrow B$	T	T	T	T	N	N	N	N
$B \rightarrow C$	T	T	N	N	T	T	N	N
$A \rightarrow C$	T	N	T	N	T	N	T	N
Triviális függés	X							
Ellentmondás		X						
Tranzitív függés								X
Részleges függés				X				
Külső kulcstörés							X	
Belső kulcstörés			X					
Metszefüggés					X			
Csoportfüggés						X		

10.3 ábra: A függések döntési táblázata (3)

Magyarázat: Táblánkat kiegészítettük a normalizálási szakirodalom által nem tárgyalt három esettel. Most már biztos lehet az olvasó abban, hogy a normális és triviális függés valamennyi kombinációja esetére megfelelő megoldást kapott. Pedig nem tettünk semmi mást, mint megengedtük a csoportos meghatározottakat is és egy picit átértelmeztük a kedvezőtlen függések mint következmények esetén végrehajtandó teendőket.

Kérdés: Fel kell-e hagyni az ortodox normalizálási eljárások alkalmazásával?

Válasz: Nem erről van szó. „Csak” annyiról, hogy szemléletet kell váltani. Az irodalom a normalizálást kizárólag a relációs adatbázis matematikai alapú *logikai szintű* tervezési eljárásának tekinti. Ezért nem engedi meg „jobbaldalon” is a csoportokat. Nekünk három teendőnk van. Először is feledkezzünk el erről a korlátról. Másodszor próbáljunk meg *fogalmi szinten* gondolkodni. Harmadszor ismerjük fel és el, hogy a matematikai alapon feltárható problémákat mindig először szemantikai elemzésnek kell alávetni.

Az  $A \rightarrow B$ ,  $B \rightarrow C$  és  $A \rightarrow C$  függéssorban a logika szabályai szerint mindig az utolsó tétel mutat a problémára, hiszen a  $C$  tényező a kettősen meghatározott. Most már tudjuk, hogy emögött a *formailag* kimutatott hiba mögött mélyebb *tartalmi* gondok rejtőzhetnek. Még a legegyszerűbb esetben is meg kell vizsgálni, hogy a  $B \rightarrow C$  és  $A \rightarrow C$  függésekben a redundánsnak mutató  $C$  valójában nem homonima-e? Nagyon sokszor az és ezért szükséges a szemantikai elemzés.

Kiegészítés: Az emberi agy áttekintő képessége korlátos. A tankönyvek mintapéldáiban mindig egyszerű tényezőket alkalmaznak. A valóságban az  $A$  lehet  $\{X+Y+Z+W\}$  és a  $B$  lehet  $\{X+Q+Z+V\}$  összetétel is. A  $C\{X+Z\}$  közös rész által okozott esetleges problémákat nagyon sokszor csak automatával lehet felfedezni. Lásd a következő pontot.



## 10.8 Pszeudo-tranzitivitás

+

Az E egyed pszeudo-tranzitív függést tartalmaz akkor és csak akkor, ha a D tulajdonsága az A+B azonosító mellett függ az implicit A+C csoporttól is és fennáll a  $B \rightarrow C$  vagy a  $C \rightarrow B$  függés.

**10.10 példa** RENDELÉSTÉTEL (*Rendelésszám+Cikkszám*, ..., R-mennyiség)  
 DISZPOZÍCIÓTÉTEL (*Diszpozíciószám+Cikkszám*, D-mennyiség, ..., R-mennyiség)

RENDELÉSTÉTEL

<i>Rendelésszám</i>	<i>Cikkszám</i>	...	R-mennyiség
12345	222222		120
12345	333333		140

DISZPOZÍCIÓTÉTEL

<i>Diszpozíciószám</i>	<i>Cikkszám</i>	...	D-mennyiség	R-mennyiség
A11	222222		60	120
A12	222222		20	120
A13	222222		40	120
A21	333333		40	140

10.4 ábra: DISZPOZÍCIÓ adatbázisrészlet

Magyarázat: Egy diszpozíció mindig csak egy rendelésre vonatkozik. Ezért fennáll a Diszpozíciószám  $\rightarrow$  Rendelésszám függés, ami persze a példában nem látszik. Éppen ez a baj, mert rejtett volta miatt nem bukkanunk rá a rendelt R-mennyiség tranzitív voltára és abból következő redundanciájára. A diszpozíciókételek nem állnak közvetlen kapcsolatban a rendeléstételekkel. (Ha a Rendelésszámot beillesztenénk a második egyedbe, akkor az az előbbi viszony miatt részleges függést eredményezne.)

Probléma: A terv *nyílt logikai átfedést* tartalmaz, amit most láthatunk is, de nincsen rá elméleti/matematikai magyarázat. Ezért az eddigi normalizálási eljárásokkal nem is lehet próbálkozni. Mivel a két egyed nem kapcsolható, nem találjuk azt a függéshármaszt, ami alapján a dekompozíciót el lehetne végezni. Ebben a helyzetben segíthet bennünket a fenti definíciónak megfelelő képlet:

X	$\rightarrow$ Y	Diszpozíciószám	$\rightarrow$ Rendelésszám
A {Y+Z}	$\rightarrow$ C	Rendelésszám+Cikkszám	$\rightarrow$ R-mennyiség
B {X+Z}	$\rightarrow$ C	Diszpozíciószám+Cikkszám	$\rightarrow$ R-mennyiség

Magyarázat: Ha elfeledkezünk a dölten szedett Cikkszámról, akkor tisztán és világosan a tranzitivitás képletével találkozunk. Azaz a pszeudo-tranzitivitás valódinak mutatkozik. Csak annyiról van szó, hogy a képletbe belekerült egy olyan „konstans” (a Cikkszám) is, ami a függéskombináció szempontjából lényegtelen. Viszont arra alkalmas, hogy elrejtse előlünk a jó megoldást.

**10.11 példa** RENDELÉSTÉTEL (*Rendelésszám+Cikkszám*, ..., R-mennyiség)  
DISZPOZÍCIÓTÉTEL (*Diszpozíciószám+Cikkszám*, D-mennyiség, ...)

Magyarázat: A pszeudo-tranzitivitást ugyanúgy küszöböljük ki, mint a normálisat. A rendelt mennyiséget már tartalmazza az első egyed, ezért azt felesleges többszörösen, azaz redundánsan megismételni a második egyedben is. A pszeudo-tranzitivitást *elvileg* könnyű felfedezni. Van egy kétszeresen függő tulajdonság (R-mennyiség). Meghatározói (Rendelésszám+Cikkszám és Diszpozíciószám+Cikkszám) összetettek úgy, hogy van egy közös részük (Cikkszám). A nemközös részükhöz (Diszpozíciószám  $\rightarrow$  Rendelésszám) pedig függés áll fenn. *Gyakorlatilag* a hasonló szituációk feltárása nem egyszerű, mert mindegyik tényező maga is összetett lehet.

Tanulság: Az adatmodell egyedeinek az összefüggéseit sohasem szabad önmagukban vagy korlátosan - például egyedpárosokra vonatkozóan - vizsgálni. Bár a 10.10 példában mutatkozó redundancia itt és most látványosan kiütözik, egy összetett modell esetében nem várható, hogy a bajok megoldásra kínálják magukat. A hagyományos normalizálási eljárás nem nyújt megfelelő elméletet és gyakorlatot a problémák kiküszöbölésére, mert két átfedő egyedünk között nem is létezik közvetlen funkcionális függési viszony.

Ismét eljutottunk oda, hogy összetett modelleknél már a matematikai összefüggéseknek a korrekt feltárásához is automatára van szükség. Olyanra, ami csak rámutat a vélhető bajokra, de nem akarja azokat automatikusan megoldani. A helyzetek értékelése és azok feloldása csak értelmezéssel történhet, ezért mindig is emberi feladat marad. Az alapvető szabályt pedig így lehet megfogalmazni:

— ***Nem relációkat kell logikai szinten normalizálni, hanem a modellt kell fogalmi szinten optimalizálni.***

Ennek az elvnek a megvalósításáról szól majd a következő fejezet.

## 10.9 Ellenőrző kérdések

10/1 Melyik állítás igaz (I) és melyik hamis (H) a következő modellrészletre nézve:

SZÁMLA (*Számlaszám*, Cikkszám, Érték)

SZÁMLA (*Számlaszám+Cikkszám*, Érték)

- Feltételezhető, hogy a Számlaszám homonima.
- A két egyed külön-külön 5NF alakban van.
- Két egyednek nem illik azonos nevet adni.
- A modellrész rossz, de azt dekompozícióval nem lehet megjavítani.
- A belső kulcstörés technikai jelensége hívja fel a figyelmet a valós gondra.

10/2 Adott a következő két egyedtípus. A terméket egy bizonyos dolgozó a megfelelő gépen állítja elő. Miért rossz a terv és miként nézne ki helyesen?

HASZNÁLJA (*Dolgozó azonosító+Gép azonosító, ..., Műszak*)  
TERMÉK (*Termék azonosító, Dolgozó azonosító, Gép azonosító*)

- 10/3 Miért hibás a következő terv és mi a helyes megoldás? Kovács beteg lett és helyette - az előzetes tervvel szemben - Szabó kezelte a gépet.

HASZNÁLJA (*Használja azonosító, ..., Műszak*)  
TERMÉK (*Termék azonosító, Használja azonosító, Dolgozó azonosító*)  
Használja azonosító {Dolgozó azonosító+Gép azonosító}

- 10/4 Milyen problémát rejt a következő terv? Egy számlán több rendelés tételei is megjelenhetnek, de egy rendelés tételei mindig egy számlán szerepelnek.

SZÁMLATÉTEL (*Számlaszám+Cikkszám, ..., Érték*)  
RENDELÉSTÉTEL (*Rendelésszám+Cikkszám, ..., Érték*)

- 10/5 Adott a Szerződésszám tulajdonság. Ennek első pár jele a Vevőkód. További jelei az előzőn belül egyedi Kötésszámot jelentenek. Vagyis két vevőnek lehet egyező a Kötésszám értéke, de egy vevő minden egyes szerződésének más-más a kötésszáma. A relációs rendszerek nem ismerik a csoportokat. Milyen gondok fakadnak ebből az Ön véleménye szerint?

## 11. NORMALIZÁLÁSI ELJÁRÁSOK

### 11.1 A normalizálás mint feldolgozás

Ha egy egyed típus nincs 2NF alakban, akkor ki kell szűrni a részleges függést. Ha nem 3NF alakú, akkor fel kell számolni a tranzitivitást. A normalizálás *elmélete* nem igazán bonyolult, ámde *gyakorlati* végrehajtása összetettebb feladatok esetében nem gyerekjáték.

A tervezők a legtöbb esetben még azt sem tudják, hogy miképpen fogjanak hozzá az elemzéshez. Ennek az az oka, hogy a szakkönyvek csak az elveket ismertetik, azonban a normalizálást nem mutatják be gyakorlati eljárásként is.

*Ennek a fejezetnek az a célja, hogy tájékoztatást adjon az ismert normalizálási eljárások alapjairól, eljárásairól és eredményeiről. Ki fogja mutatni, hogy az utóbbiak sokszor nem kielégítőek. Ezért ismertetni fog egy új normalizálási megközelítést.*

A normalizálás maga is számítógéppel támogatható *feldolgozás*. Ezért van bemenete, amit normalizálási *alapnak* nevezünk. Ezt feltehetőleg egy normalizálási adatbázisban tároljuk, ami a legtöbb esetben az egyedeket, a tulajdonságokat és a függési viszonyokat őrzi. A tényezőket a megfelelő algoritmussal, általában több iteráció során átalakítjuk, vagyis fokozatosan kedvezőbb alakra hozzuk. A normalizálási *eredmény* az adatmodell meghatározott részlete. Nem maga a modell, mert az olyan tényezőket (pl. kapcsolatok) is tartalmaz, amik nem vesznek részt a normalizálásban.

A normalizálási alap, eljárás és eredmény jellege a választott módszertől függ. Másból indul ki és más eredményt ad a két alapvető eljárás (11.2 *Normálforma analízis* és 11.3 *Normálforma szintézis*). A hagyományos dekompozíciós eljárással az előző fejezetekben részletesen foglalkoztunk. Ezért annak lényegét most csak röviden fogjuk összegezni. A szintézisről ki fogjuk mutatni, hogy a kitűnő alapgondolat dacára elvileg is, gyakorlatilag is tökéletlen eljárás. A fejezet végén az általunk kifejlesztett normalizálási módszert (11.12 *Fonal*) fogjuk ismertetni. Azt a következő érdekes helyzetek, szerkezetek és résztechnikák ismertetésével alapozzuk meg:

- 11.4 *Kapcsolathiány*
- 11.5 *Ellentmondó függések*
- 11.6 *Evidens függések*
- 11.7 *Kulcsmátrix*
- 11.8 *Egyedek közötti függés*
- 11.9 *Lineáris szerkezet*
- 11.10 *Ciklikus függés*
- 11.11 *Közvetett tranzitivitás*

### 11.2 Normálforma analízis

Definíció: A normálforma analízis az a normalizálási eljárás, amelyben sorra véve az egyedeket egyenként megvizsgáljuk azok tulajdonságainak a *funkcionális*, *többértékű* és *kapcsolási függéseit*, és a kedvezőtlen (részleges, tranzitív stb.) függésű tulajdonságokat kiemeljük az egyedből. A

megbontás miatt az eljárást normálforma dekompozíciónak is hívják. A kiemelt tulajdonság vagy új egyednek ad alapot, vagy egy már meglévő egyed tételévé válik, vagy - ha már szerepel a megfelelő egyedben - egyszerűen csak elhagyjuk.

Tényezők: A normalizálás **alapja** az egyed-tulajdonság-viszonyok halmaza és az azok között egyedenként feltárt függések készlete. A normalizálás **eljárása** két mozzanatból áll. Az egyedek a fentebb leírt módon megbontjuk, de ezzel párhuzamosan a közös kulcsú egyedeket a **minimalitás** jegyében egymással összevonjuk. Tehát például a következő két egyedből a harmadik fog születni:

**11.1 példa**      átmeneti: VEVŐ (**Vevőkód**, Vevőcím)  
                      átmeneti: VEVŐ (**Vevőkód**, Vevőtípus, Vevőnév)  
                      VEVŐ (**Vevőkód**, Vevőcím, Vevőtípus, Vevőnév)

A normalizálás **eredménye** az egyenként optimális alakú (5NF) egyedek együttese. Ez az eredmény nagyon sok esetben nem teljes. Lásd a következő pontot. Ezért az analízis elméletileg tökéletlen modellezési eljárás.

Az eljárás lépéseit a 7-9. fejezetek részletesen leírták.

## 11.3 Kapcsolathiány

Definíció: A csúnya szóval kapcsolhatatlanságnak [inconnectivity] is nevezett jelenség lényege az, hogy két fogalmilag összefüggő egyednek nincsen olyan közös tulajdonsága, ami alkalmas volna a kapcsolat megteremtésére (← Logikai átfedés hiánya).

**11.2 példa**      VEVŐ (**Vevőkód**, Vevőcím)  
                      RENDELÉS (**Rendelészám**, Vevőnév)  
  
                      VEVŐ (**Vevőkód**, Vevőcím, Vevőnév)  
                      RENDELÉS (**Rendelészám**, Vevőnév)

Magyarázat: Az első két egyedben nincs közös tulajdonság, így nem juthatunk el a VEVŐ egyedből a RENDELÉS egyedbe és megfordítva. A második két egyedben van ugyan közös tétel, de mivel a Vevőnév értéke nem egyedi, az nem alkalmas a kapcsolat egyértelmű kifejezésére.

Probléma: A négy egyed mindegyike külön-külön tökéletes alakban (5NF) van, viszont mindkét modellrészlet együttesen rossz, mert hiányos: megsérti a **teljesség** követelményét. Emiatt nem teljesül az a korlát, ami szerint minden rendelés egy jól meghatározott vevőhöz kell, hogy kapcsolódjon és minden vevőnek lehet 0, 1, ..., N darab megrendelése. Ezért a fenti példák szerint felépített modell nem lesz integráns.

Megoldás: A kapcsolathiány nem fedezhető fel a normálforma analízis eljárásával, mert annak során az egyedeket egyenként normalizáljuk és nem vizsgáljuk az **egyedek közötti függéseket**. A fenti példák célszerű megoldása a következő:

**11.3 példa** VEVŐ (Vevőkód, Vevőcím, Vevőnév)  
 átmeneti: RENDELÉS (Rendelésszám, Vevőkód, Vevőnév)  
 RENDELÉS (Rendelésszám, Vevőkód)

A megoldás a következő szabályon alapul:

— ***Ha egy tulajdonságtípus közvetlenül függ egy másiktól, akkor a másik által azonosított egyed típusban van a helye.***

Mivel a Vevőkód közvetlenül függ a Rendelésszámtól, azt a RENDELÉS egyedbe kell tenni. Ekkor fény derül a Vevőnév tranzitivitására, amit kiküszöbölve jutunk el a végső eredményhez. A helyes megoldást elméletileg a normálforma szintézis eljárása alapozza meg. Gyakorlatilag annak egy korlátozott változatát célszerű alkalmazni (→ kulcsmátrix).

## 11.4 Normálforma szintézis

Segédfogalom: Az **univerzális reláció** [universal relation - <sup>16</sup>] az alkalmazás összes egyértelműen meghatározott - tehát nem homonim és nem szinonim - tulajdonságtípusát felölelő egyetlen egy (óriási) általános „egyed”. Mivel itt a valódi egyedektől független tulajdonságokról van szó, azok relációs értelemben tulajdonképpen doméjnek.

Definíció: A normálforma szintézis [<sup>17</sup>] az a normalizálási eljárás, aminek során az egyetlen univerzális relációt alkotó tételek közötti függéseket vizsgáljuk, és a függő illetve meghatározott párosokból immár valós relációkat építünk fel. Mivel ebben az esetben egyedektől független tulajdonságokról van szó, a vizsgált viszonyok **tartományfüggések**. A szintézis felfogható úgyis, mint az univerzális reláció dekompozíciója.

Tényezők: A normalizálás **alapja** az univerzális reláció és a tételek között felfedezett doméjn-függések halmaza. Csak a közvetlen - nem-részleges, nem-tranzitív stb. - függések vizsgálándók. A normalizálási **eljárás** szabályait a következő példán világítjuk meg:

**11.4 példa** EGYETEMES (Vevőkód, Vevőnév, Vevőcím, Rendelésszám)  
 Vevőkód => Vevőnév  
 Vevőkód => Vevőcím  
 Rendelésszám => Vevőkód  
 átmeneti: Rendelésszám => Vevőnév  
 átmeneti: Rendelésszám => Vevőcím

Az univerzális reláció négy tételén öt függést fedeztünk fel. Ezek közül a két utolsó nem jöhet számításba, mert nem közvetlen, hanem (a Vevőkódon át) tranzitív. Az adatmodellt alkotó három tényező (egyed-, tulajdonság- és kapcsolattípus) meghatározására három egyszerű szabály szolgál:

- Hozz létre annyi egyedtípust, ahány különböző meghatározó van a „baloldalon”. Minden ilyen egyedtípusnak a baloldali tulajdonságtípus lesz az azonosítója.  
(VEVŐ - Vevőkód, RENDELÉS - Rendelésszám)
- Az így létrehozott egyedtípusokhoz a baloldali meghatározók szerint kapcsold hozzá leíró tulajdonságokként a megfelelő „jobboldali” tételeket.  
(VEVŐ - Vevőnév, Vevőcím, RENDELÉS - Vevőkód)
- Az egyedek között annyi kapcsolatod lesz, ahányszor a „baloldali” tényező más függésben megegyezik a „jobboldali” tétellel.  
(VEVŐ-ben Vevőkód baloldalt, RENDELÉS-ben Vevőkód jobboldalt)

**11.5 példa** VEVŐ (**Vevőkód**, Vevőcím, Vevőnév)  
 RENDELÉS (**Rendelésszám**, *Vevőkód*)  
 VEVŐ-RENDELÉS kapcsolat a Vevőkód alapján

A normalizálás *eredménye* elvileg egy teljes, kapcsolható modell és ezért a leírt eljárás elméletileg jobb eredményt ad, mint a normálforma analízis. Azonban a valóságban a szintézis gyakorlatilag nem végrehajtható és elméletileg is tökéletlen.

Probléma: Ha az univerzális reláció 500 tételt tartalmazna, akkor  $500 \cdot 499 / 2 = 124750$  elemi függés vizsgálatát kellene elvégezni. Ekkor pedig még nincs is szó az  $A+B+\dots \rightarrow C$  jellegű összetett függésekről, amelyek száma megjósolhatatlan. Ezért jelenthető ki, hogy a normálforma szintézis eredeti eljárása *gyakorlatilag* végrehajthatatlan.

Éppen az összetett tényezők miatt a módszer *elméletileg* is tökéletlen. Léteznek ugyanis úgynevezett csupakulcs relációk - például E (**A+B**) - úgy, hogy azokban nincs függő, azaz meghatározott tétel. Világos, hogy ilyen reláció a fenti három szabállyal nem kreálható. A szintézis egy másik problémával sem tud megbirkózni. Lásd a következő pontot.

## 11.5 Ellentmondó függések

Lényeg: Az egy adott egyeden belüli funkcionális függéseken alapuló szerkezetek néha ellentmondhatnak a tulajdonságok közötti doménfüggésekből adódó struktúráknak.

**11.6 példa** KÖZPONT (**Központ azonosító**, ...)  
 TELEPHELY (**Telephely azonosító**, ..., *Központ azonosító*)  
 RENDELÉS (**Rendelésszám**, *Központ azonosító*, *Telephely azonosító*)

Magyarázat: A rendeléseket hol a központ, hol a telephely adja fel. Ha az előbbi teszi, akkor a Telephely azonosítónak nincs értéke. Ezért a RENDELÉS egyedben nem áll fenn a Telephely azonosító  $\rightarrow$  Központ azonosító funkcionális függés. Nincs tranzitivitás, tehát a példa modell-részlete jól meghatározott.

Probléma: Normálforma *szintézisnél* a tervező a nevezett két tulajdonság között csak egyféle viszonyt jelölhet ki. Tehát vagy feltételezi a Telephely azonosító  $\Rightarrow$  Központ azonosító függést, vagy úgy tartja, hogy az nem létezik. Ugyanis az univerzális reláció két tétele között vagy van, vagy nincs függés. Az első esetben a TELEPHELY egyedbe tudja vinni a Központ azonosítót, de

az már nem kerülhet a RENDELÉS-be. (Ott ugyanis nem áll fenn a függés.) A második esetben a RENDELÉS egyedbe kerül a Központ azonosító, de a függés hiánya miatt a TELEPHELY-ből marad ki.

Végeredményben a szintézis kizárja azt, hogy két tulajdonság több egyedben is együtt forduljon elő, ha létezik olyan egyed, amiben az egyik meghatározza a másikat. Mivel pedig a 11.6 példa megoldása jó, de nem érhető el szintézissel, ez az eljárás elvileg hibás.

## 11.6 Evidens függések

Probléma: A *normálforma analízis* nem garancia a teljes modellre (← *kapcsolathiány*). A *normálforma szintézis* gyakorlatilag végrehajthatatlan és elméletileg is tökéletlen. Az előbbi esetben próbálunk ragaszkodni az előre elképzelt egyedekhez és ezért nem látjuk a kellő össze-függéseket. Az utóbbi esetben pedig elveszünk a függések óriási halmazában.

Megoldás: Változtatni kell azon a szokáson, hogy a függéseket mindig a meghatározó felől szemléljük. Azaz például a DISZPOZÍCIÓTÉTEL (**Diszpotételszám**, Rendelésszám, Cikkszám, Rendelt mennyiség) egyedben azt vizsgáljuk, hogy a kulcs meghatározza-e a Rendelt mennyiséget. A célravezető megoldás kulcsa a következő szabály:

— ***A normalizálásból kizárhatók a leíró abszolút szerepű tulajdonságok.***

A Rendelt-mennyiség abszolút szerepe biztosan leíró, mert ez a tulajdonság nem lesz se kulcs, se kulcsrész. Az ilyen tételnél nem azt kell adott egyedhez kötöten vizsgálni, hogy meghatározza-e őt például a Diszpotételszám. Hanem a kérdést meg kell fordítani és arra kell választ kapni, hogy melyik az a tulajdonság(együttes), amiktől ő közvetlenül függ. A válasz a példa esetében evidens: Rendelésszám+Cikkszám → Rendelt mennyiség. Ezért a meghatározottnak a meghatározó által azonosított egyedben és *csak ott* (!) van a helye.

Definíció: Egy függést akkor nevezünk evidensnek (nyilvánvalónak), ha a függő tétel egy és csakis egy közvetlen meghatározóval rendelkezhet.

Magyarázat: A Személy azonosító → Személy név függés evidens. Ha tehát létezik egy SZEMÉLY egyedtípusunk (a generalizációról most feledkezzünk el), akkor evidens, hogy ott és csak ott van a Személy név leíró tulajdonság helye. Ez a tétel minden más egyedben csak redundáns lehet. Visszatérve a fenti szabályra annak értelme most már világos: nem kell a normalizálás során törődnünk a Személy név tényezővel. Azt csak egyetlen egy egyedben szabad elképzelni, minden más egyedből el kell távolítani. Pontosabban szólva nem szabad felvenni semmilyen másik egyedbe sem.

Megjegyzés: Természetesen nem kis gyakorlati tapasztalat szükséges ahhoz, hogy egy tulajdonságról eleve ki merjük jelteni, hogy az csak leíró szerepű lehet. Ezért a kezdő tervező kétes esetben jobban teszi, ha nem él ezzel a feltételezéssel.



## 11.7 Kulcsmátrix

Definíció: A kulcsmátrix egy olyan trianguláris táblázat, ami a feltételezett **azonosító abszolút szerepű** tulajdonságok függéseit tartalmazza. Vagyis azt mutatja, hogy egy elemi vagy összetett kulcs függési viszonyban áll-e egyenként az összes többivel.

	Vevő	Telephely	Rendelés	Cikk	Rendeléstétel
Vevő	-	T	T	*	
Telephely		-	T	*	
Rendelés			-	*	
Cikk				-	Ó
Rendeléstétel					-

11.1 ábra: Kulcsmátrix

Magyarázat: Az ábra a meghatározó (Ó) és a meghatározott (T) kulcsokat illetve a függéshányt (\*) mutatja. A táblába a tulajdonságok helyett az egyedek neveit írtuk, de ez nem hiba, mivel az egyed és kulcsa lényegében ugyanaz a dolog. A Vevő azonosítója a Telephelyé által *közvetlenül* meghatározott (T). A Rendeléstétel kulcsa a Cikkének közvetlen meghatározója (Ó). A *közvetett* meghatározásokat dőltbetűs szedet jelzi.

Eljárás: A kulcsmátrix arra való, hogy segítségével feltárjuk a **kapcsolathiányokat**. A már meglévő egyedeink (illetve feltételezett azonosítók) alapján a táblába bevezetjük az általunk ismert közvetlen összefüggéseket. Eközben mindjárt meghatározhatjuk illetve kiszűrhetjük a közvetetteket is. A Rendelés meghatározza a Telephelyet, az pedig a Vevőt, tehát a Rendelés és a Vevő között létezik egy közvetett meghatározás. Ezt azért vezetjük be a táblába, hogy vizsgálni tudjuk a fennmaradó üres helyeket.

A Rendeléstétel alatt van három ilyen ismeretlen tartalmú bejegyzés. Feltesszük azt a kérdést, hogy a sor határozza-e meg az oszlopot, fordítva, vagy nincsen függés? Látjuk, hogy a Rendeléstétel meghatározza a Rendelést (oda T-t kell írunk), amiből az is adódik, hogy közvetve (T bejegyzés) meghatározza a Telephelyet és a Vevőt is. Ezek után nem marad további vizsgálandó összefüggésünk: előttünk áll a modell kapcsolati rendszere.

Megjegyzés: A normálforma szintézis a nagy darabszámok miatt gyakorlatilag nem alkalmazható eljárás. Elméletileg nem engedi meg két jelenség egynél többféle viszonyát. A kulcsmátrix a szintézisnek egy sajátosan átalakított változata. Egyrészt mivel a kulcsok száma nagyságrenddel kisebb, mint az összes tulajdonságé, a fenti eljárás gyakorlatilag végrehajtható. Másrészt a kulcsmátrix nem zárja ki a többféle viszonyt. Itt a Rendelés és a Vevő kapcsolatát tranzitívnak tekintettük (T bejegyzés), de a mátrixban azt közvetlenként is (T bejegyzés) meghatározhattuk volna (← *Ellentmondó függések*).

## 11.8 Egyedek közötti függés

Definíció: Egyedek közötti függésről beszélünk akkor, ha az egyik egyed azonosítója úgy határoz meg egy másik tulajdonságot, hogy az egy másik egyednek a kulcsa. Ekkor az egyeden belüli [intra-entity] függés egyben egyedek közöttivé [inter-entity] is válik.

Példa: A RENDELÉS *egyeden belüli* Rendelésszám → Vevőkód függés egyben az adott és a VEVŐ *egyed közötti* függést is jelenti.

Probléma: A normálforma analízis arra ad szabályokat, hogy az egyedeket miképpen kell lebontani a függéseknek megfelelően. Arra viszont nem nyújt megoldást, hogy milyen alapon kell meghatározni az egyedek közötti szerkezeteket. Bár a külső kulcs erre nézve ad némi támpontot, az nem épül a kölcsönösség elvére. Ennek az az oka, hogy a relációs adatbázisban a kapcsolatot nem is tekintik önálló modellezési tényezőnek.

Magyarázat: A tulajdonságok közötti funkcionális függés nem más, mint az értékek *hierarchikus* (1:N fokú) viszonya. A Rendelésszám → Vevőkód függés annyit jelent, hogy minden Rendelésszám értékhez csakis egyetlen (1) Vevőkód érték tartozhat, viszont minden Vevőkód érték több (N) Rendelésszám tartalomhoz is kapcsolódhat. A nevezett két fogalom kulcsként szolgál. A RENDELÉS egyedelőfordulások és a rendelésszámok, a VEVŐ egyedelőfordulások és vevőkódok kölcsönös és egyértelmű 1:1 viszonyban állnak egymással. A szóbanforgó függés ezért azt is jelenti, hogy a két egyedtípus előfordulásai között is 1:N fokú viszony áll fenn. Vagyis végeredményben a Rendelésszám → Vevőkód egyedek közötti függés nem más, mint a VEVŐ-RENDELÉS kapcsolat.

Miért fontos ez a felismerés? Azért, mert megoldódik a normálforma analízis gondja. Nemcsak az egyedeken belül, hanem az egyedek között is lehetőség van a normalizálásra. Ráadásul a tervező kétféle egymást kiegészítő, egymást mintegy ellenőrző módszert is választhat. Ha abból indul ki, hogy létezik egy egyedek közötti függés, akkor feltétlenül meg kell határoznia (ha addig még nem tette volna) a két egyed közötti kapcsolatot. Ha pedig abból indul ki, hogy létezik egy kapcsolat, akkor gondoskodnia kell az azt hordozó kapcsolótulajdonságról. Vagyis ha a RENDELÉS egyedből kifejejtette volna valamiért a Vevőkódot, akkor azt pótlólag oda kell helyeznie.

Végeredményben a függések alulról kiinduló hálója és a kapcsolatok felülről szemlélt hálója ugyanannak az egyetlen lényegnek a két oldala. A „rendelés meghatározza a vevőt” ugyanazt jelenti, mint a „vevőnek vannak megrendelései” állítás. A szemantikai összefüggések (kapcsolatok) így teljes harmóniában vannak a matematikai viszonyokkal (egyedek közötti függések). A kivételes problémákat lásd a következő pontban.

## 11.9 Lineáris szerkezet

Definíció: Lineáris szerkezetről akkor beszélünk, ha az egyedtípusok között úgy áll fenn 1:1 fokú kapcsolat, hogy azok nem alkotnak altípus hierarchiát. A partner-személy-férfi specializáció altípus hierarchiát alkot, ezért ezt a szerkezetet nem tekintjük lineárisnak.

**11.6 példa**      VEVŐ (**Vevőkód**, Vevőcím, Vevőnév)  
                    RENDELÉS (**Rendelésszám**, *Vevőkód*)  
  
                    RENDELÉS (**Rendelésszám**, ...)  
                    CIKK (**Cikkszám**, ...)  
                    RENDELÉSTÉTEL (**Rendelésszám**+**Cikkszám**, ...Rendelt mennyiség)

KOCSI (**Rendszám**, ...)  
CASCO (**Cascoszám**, ...)

Magyarázat: A fenti példasorozatban az első két egyed viszonya *hierarchikus*, hiszen egy (1) vevőhöz több (N) rendelés tartozhat. A következő három egyed közül az első kettő viszonya *hálós*, mert egy rendelésben kérhetnek több cikket (M) és egy cikket tetszőleges számú (N) rendelésben igényelhetnek. Az adatmodellezésben a hálós viszonyokat mindig egy olyan harmadik egyeddel oldjuk fel, ami a másik kettővel hierarchikus kapcsolatban áll. A rendelések és a cikkek a rendeléstételeken át kapcsolódnak egymáshoz.

A harmadik együttes szándékosan ennyire üres. Egyelőre csak annyit tudunk, hogy egy kocsinak legfeljebb csak egy (1) Casco-ja lehet és megfordítva, egy adott Casco-biztosítás legfeljebb csak egy (1) kocsira vonatkozhat. Mivel egyik egyed sem altípusa a másiknak, együtt tipikus lineáris szerkezetet alkotnak.

Probléma: Háromféle gond merül fel. Mivel a két egyed egymással kapcsolatban áll, ezt a viszonyt kapcsoló tulajdonsággal kellene tükrözni. A kölcsönösség miatt valaki esetleg a következő kettős megoldást alkalmazná:

**11.7 példa** KOCSI (**Rendszám**, *Cascoszám*)  
CASCO (*Cascoszám*, **Rendszám**)

A redundancia és a karbantartási anomália ekkor nyilvánvaló. Például az első egyedben adott rendszám mellett átírjuk a Cascoszám értéket, de ezt nem tesszük meg a másikban. Így a redundancia óriásira duzzadhat és teljesen lehetetlenné válik a normalizálás is:

**11.8 példa** KOCSI (**Rendszám**, *Cascoszám*, Kocsitípus, ...)  
CASCO (*Cascoszám*, *Rendszám*, Kocsitípus, ...)

Ha hagyományosan normalizálnánk, akkor a Kocsitípust a tranzitív függés miatt a második egyedből át kellene tennünk az elsőbe, onnan a másodikba, onnan az elsőbe stb. A normalizálási eljárás végtelen ciklusba torkolna abból az egyszerű okból kifolyólag, hogy a fenti szerkezet valóban *ciklikus függést* tartalmaz.

Ha nem alkalmazunk kapcsolótulajdonságot az is baj, ha alkalmazunk az is gond. A harmadik felvetésre sem könnyű a válasz. Egyes tervezők úgy vélhetnék, hogy a Casco nem is igazán külön egyedtípus, hanem a kocsi egyed néhány sajátos tulajdonságtípusa. Azonban ez a megközelítés sem állja meg a helyét. Elméletileg azért nem, mert a Casco nem azonos a kocsival, hanem önálló ismeretekkel leírandó jelenség. Gyakorlatilag pedig azért nem, mert minden ikszedik Casco-biztosítással nem rendelkező kocsinál üres értékek mutatkoznának. Ezért nem az a kiút, hogy a CASCO egyed jogosságát vitatjuk.

Megoldás: Ha két egyed kölcsönösen és kötelezően 1:1 fokú viszonyban áll, akkor azokat feltétlenül össze kell vonni. Lásd normálforma analízis 11.6 példa. Ha a viszony kölcsönös, de nem mindkét oldalról kötelező, akkor fölé/alárendeltséget kell feltételezni. Ilyenkor az alárendeltet *kiegészítő egyednek* hívtuk.

**11.9 példa** KOCSI (**Rendszám**, ..., Kocsitípus)  
CASCO (*Cascoszám*, ..., *Rendszám*)

A megoldásban a két egyed egymáshoz kapcsolható. Nem redundánsak. Normalizálási probléma nem lép fel. A CASCO a KOCSI kiegészítő egyede. Ezzel a minősítéssel adjuk meg azt a korlátot, hogy egy kocsihoz minden időben csak nulla vagy egy Casco tartozhat. Mindez pedig a *függésről* alapul. A Rendszám - Cascoszám függés nem kölcsönös, mert csak az utóbbi oldalról kötelező.

**Figyelmeztetés!** A mai adatkezelő rendszerekben nincs lehetőség az egyedek olyasféle minősítésére, mint a fenti „kiegészítő” jelző. Ugyanakkor nincs akadálya a 11.9 példa által mutatott szerkezet létrehozásának. Ami pedig azzal jár, hogy az alapelveket megszegve a két egyed viszonyára vonatkozó korlátot procedurálisan kell betartatni vagy legalábbis nem lehet a modellben a kiegészítés/altípusra bontás lényegét szétválasztani.

## 11.10 Ciklikus függés

Definíció: Ciklikus függésről beszélünk akkor, ha egy egyed valamelyik tulajdonsága az egyedek közötti függéseken át közvetlenül vagy közvetve saját magát meghatározza.

**11.10 példa** KOCSI (**Rendszám**, Cascoszám)  
CASCO (**Cascoszám**, Rendszám)

MEGYE (**Megyekód**, ..., Városkód)  
JÁRÁS (**Járáskód**, ..., Megyekód)  
VÁROS (**Városkód**, ..., Járáskód, X)

Magyarázat: Mivel a Rendszám az első egyed szerint meghatározza a Cascoszámot, a második egyedben pedig a fordított függés áll fenn, közvetlen ciklikus helyzet áll elő. A másik részletben a ciklus közvetett: Városkód → Járáskód → Megyekód → Városkód.

Probléma: A bajok kettősek. Ha mindegyik függést kötelezőnek feltételezzük, akkor az adatbázisba nem tudunk egyetlen ismeretet sem beilleszteni. Mert a városhoz ott kell már lennie a járásnak, ahhoz a megyének, ahhoz pedig a beillesztendő városnak. A másik gond a normalizálási eljárással kapcsolatos. Melyik egyed jellemzi az „X” tulajdonság? Mivel a Városkódot meghatározza a Megyekód, a MEGYÉ-be kellene átmozgatnunk. A Megyekódot meghatározza a Járáskód, ezért a JÁRÁS-ban volna a helye. Majd a végén visszakerül a VÁROS-ba, hogy onnan áttegyük ismét a MEGYÉ-be stb.

—

**Nem normalizálható az olyan modell, amely ciklikus függést tartalmaz.**

Megoldás: Ebből a szabályból az következik, hogy a normalizálás előtt a közvetlen és a közvetett ciklusokat meg kell szüntetni. A ciklikus struktúra feloldása általában opcionális tulajdonságokkal történik. A közigazgatási probléma egyszerűen megoldható úgy, hogy a megfelelő városnál utalunk arra a megyére, amelynek az adott település a központja:

**11.11 példa** MEGYE (Megyekód, ...)  
JÁRÁS (Járáskód, ..., Megyekód)  
VÁROS (Városkód, ..., Járáskód, Megyekód, X)

**Figyelmeztetés!** A ciklusok feltárására a modellezési segédletek egyike sem képes.

## 11.11 Közvetett tranzitivitás

Definíció: Közvetett tranzitivitásról beszélünk akkor, ha az egyed kulcsa egy olyan tulajdonságon át határoz meg egy további tételt, ami annak nem közvetlen azonosítója.

**11.12 példa** VEVŐ (Vevőkód, Vevőnév)  
SZERZŐDÉS (Szerződésszám, Vevőkód)  
RENDELÉS (Rendelésszám, Szerződésszám, Vevőnév)

Magyarázat: A Rendelésszám a Szerződésszámon át meghatározza a Vevőnevet, ami ezek szerint a RENDELÉS egyedben tranzitív. Azonban a Szerződésszám a Vevőnévnek nem közvetlen azonosítója, mert hiszen csakis a Vevőkód tekinthető annak.

Probléma: Az igazi gondot a *normalizálási sorrend* jelenti. A normálforma analízisben semmiféle szabály nem létezik arra nézve, hogy a modell melyik egyedénél kell elkezdni a normalizálást. Ha például az eljárást a SZERZŐDÉS egyednél kezdjük, akkor azt persze teljesen „normálisnak” találjuk. Amikor tovább lépünk a RENDELÉS-re, akkor látjuk meg a Vevőnév tranzitivitását. Az egyedet megbontva átmozgatjuk a Vevőnevet az azt meghatározó Szerződésszám egyedébe:

SZERZŐDÉS (Szerződésszám, Vevőkód, Vevőnév)

Ekkor itt is felfedezzük a tranzitivitást. Jóllehet az egyedet egyszer már normalizáltuk, azt újból elő kell vennünk és meg kell bontanunk, hogy megkapjuk a jó végeredményt:

**11.13 példa** VEVŐ (Vevőkód, ..., Vevőnév)  
SZERZŐDÉS (Szerződésszám, ..., Vevőkód)  
RENDELÉS (Rendelésszám, ..., Szerződésszám)

Most képzeljen el az olvasó két dolgot! Egyrészt egy sokkal *mélyebb* láncot, másrészt pedig egy sokkal *szélesebb* hálót. Vajon hány iterációt fog igényelni, hogy minden tétel a maga célszerű, végső helyére kerüljön?

Megoldás: Az egyedenkénti normalizálás csak akkor ad jó eredményt, ha ki tudja hány iteráció során egyszer sem tévedünk és a dekompozíciókat végigvisszük. Ennél elvileg és gyakorlatilag is sokkal jobb megoldás a *fonalak* alkalmazása. Lásd a következő pontot.

**Figyelmeztetés!** A közvetett tranzitivitások feltárására a mai modellezési segédletek legtöbbje nem képes. Az ilyen problémák kiküszöbölése a tervező személyes feladata.

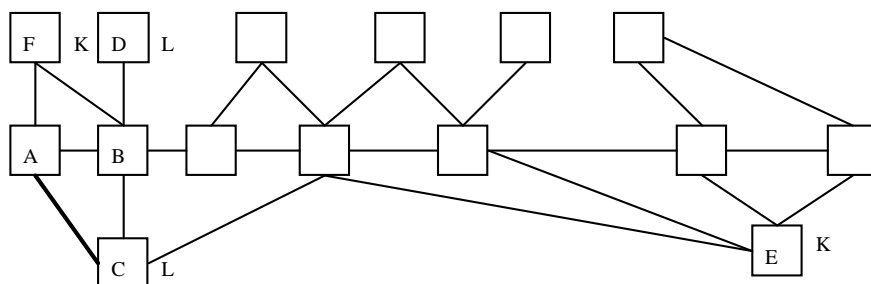
## 11.12 Fonalak

Segéddefiníciók: Az adatmodellben **kezdőpontnak** tekintjük azt az egyedet, aminek a kulcsát már nem határozza meg egy másik egyed azonosítója, vagyis aminek már nincs alárendeltje. Az adatmodellben **végpontnak** hívjuk az olyan egyedet, aminek a kulcsa már nem határozza meg egy másik egyed azonosítóját, tehát aminek már nincs fölrendeltje. Képszerűen a kezdőpont az ábra legalsó, a végpont a legfelső szintjén látható.

+

**A fonal [thread] az adatmodell egy adott kezdőpontjától egy adott végpontjáig húzódó, az azonosítók közötti függések által kapcsolható egyedek egyirányú sorozata.**

Példa: A 11.2 ábra egy sematikus adatmodellt mutat. A vízszintes vonalak balról-jobbra mutató kapcsolatokat és ellentétes irányú függéseket jelentenek. Tehát például az F egyed és a B egyed egyaránt az A tétel alárendeltje, vagyis tartalmazza annak kulcsát.



11.2 ábra: Egy viszonylag egyszerű adatmodell váza

Magyarázat: Az ábrán a C és az E egyed kezdőpont, az F és a D végpont. A C-B-A-F, C-A-F, C-B-F és C-B-D együttesek alkotnak fonalakat, nem beszélve most az E egyedből kiindulókról. A modell hálójá akkor korrekt, ha megszakításmentes ( $\leftarrow$  *kapcsolathiány*), ha nincs benne kör-körösség ( $\leftarrow$  *ciklikus függés*) és közvetett vagy közvetlen tranzitivitás. A háló csak az azonosítók által tükrözött egyedekre vonatkozik. Abban nem tüntetjük fel a leíró tulajdonságokat. Mi ezt csak a későbbi magyarázat kedvéért tettük (lásd K és L).

Probléma: A normálforma **analízis** eljárásánál a dekompozíció „vándoroltatást” igényel. A C egyedben felfedezzük a  $C \rightarrow B \rightarrow L$  tranzitivitást. Ezért az L leíró tulajdonságot a B egyedbe tesszük át. Ott felfedezzük a  $B \rightarrow D \rightarrow L$  függéssort, ezért az L tétel a D egyedbe kerülne, ha már nem lenne ott. Az  $E \dots \rightarrow F \rightarrow K$  függés esetében a traverzió még több lépést igényel. További gondként merül fel, hogy egyáltalán felismerjük-e az  $E \dots \rightarrow F$  igen távoli függést. Ezért a hagyományos normalizálási eljárás egyrészt nehézkes, mert sok lépést igényel, másrészt megbízhatatlan.

Megoldás: A fonalak alapján történő elemzés <sup>[15]</sup> gyors, megbízható és abban a ciklus feltárása illetve a helytelen (pl. tranzitív) függések kiszűrése egy algoritmussal történik. A fonalakat alulról, a kezdőpontoktól építjük fel. Világos, hogy ha egy fonalban megjelenik egy olyan elem, ami abban már korábban is szerepelt, akkor ciklusba botlunk. Így például a  $Q - X - Y - Z - X$  fonal nem is építendő tovább, mert az az XYZX ciklust tartalmazza.

Amikor a fonalak azonos elemétől hosszabb és rövidebb út is vezet egy másik közös elembe (C-B-A és C-A részlet) és a függések erősek, akkor felfedeztünk egy tranzitív utat. Lásd a vastagon kiemelt vonalat. Az ilyen ösvényt meg kell szüntetni, azaz a C egyedből ki kell venni az A kulcsát. Amint látható, ebben az eljárásban nincs vándoroltatás, mert a **CBA/CA** fonalakból azonnal kitetszik a CA tranzitivitás. Az algoritmus szempontjából pedig közömbös, hogy a fonal mennyire hosszú. Az **E(K)...BF(K)** fonalból - a K-t zárójelbe tettük, mert leíróként nem a fonal része, bár ahhoz a mutatott pontokon kötődik - azonnal kiderül, hogy a K leíró tulajdonság az E egyedben redundáns.

(NB.: Mivel a kapcsolatok lefelé irányultán 1:N fokúak, a redundáns tulajdonságok mindig a fonal kezdete felé lépnek fel, hiszen ott található az alárendelt egyedek. A K nem az F egyedben, hanem az E-ben - ami az előbbinek áttételes alárendeltje - többszörös. Azonban a probléma nem mindig a fonal legelején található. Például a CBAF és a CBF fonalak esetében a BF részfonal tranzitív.)

Szemben a szintézissel, a fonal eljárás erősen lecsökkenti a vizsgálatok számát. Mivel a fenti ábrán nincs olyan fonal, ami együtt tartalmazná a C és az E egyedtypust, az ezek közötti függéseket teljesen felesleges vizsgálni. A párhuzamos szálakon nem léphet fel probléma. Ciklus és redundancia csak a fonalak mentén alakulhat ki, ott is csak akkor, ha a szálak összefutnak és eltérő hosszúságúak. Például bár a Q-X-Y-Z és a Q-V-W-Z láncok összefutnak, de azonos hosszúak és nem is fedezhető fel bennük tranzitivitás.

Figyeljük meg az azonosító és a leíró szerepű tulajdonságok normalizálásának eltérését. A kulcsokkal akkor van baj, ha két szál összefut (**CBA/CA**) és nem egyenlő hosszúak. A leírók akkor okoznak gondot, ha egy fonálon többször is megtalálhatók (**E(K)...BF(K)**).

Kiegészítés: Az egymástól független - lásd C és E - egyedekben lévő esetleges közös tulajdonságok sohasem okozhatnak bajokat. Ha felfedezzük a függetlenségüket, akkor azt érdemes gyorsan megvizsgálni, hogy esetleg nem feledkeztünk el a kapcsolatukról. Tehát azt kell nézni, hogy a C kulcs meghatározza-e az E azonosítóját vagy fordítva. Ezzel a gyorsselelemzéssel esetleg kiszűrhetők a **kapcsolathiányok**.

Ezzel az alapvető függési viszonyok, a normálformák és a normalizálás tárgyalását lezárjuk és áttérjünk az adatmodell szerkezeti finomságainak az ismertetésére.

## 12. SAJÁTOS SZERKEZETI TÉNYEZŐK

### 12.1 Minőségi adatmodellezés

A mai adatbáziskezelő rendszerek többsége „rekord-orientáltan” dolgozik. Ez két dolgot jelent. Egyrészt csak a tulajdonságok meglétére illetve egyszerű összefüggéseire (tehát a mennyiségekre) figyelnek, azok sajátos értelmezéseire (tehát a minőségekre) már nem. Például a szerepneveket is csak közönséges tulajdonságoknak tekintik és így nem tudnak mit kezdeni azok viszonyaival (12.3 *Szerepnevek függései*). Ezért a tervezőnek a logikai tervezés szintjén kell megadnia olyan tényezőket (12.2 *Többszörös inhomogén kapcsolat* és 12.4 *Homogén viszonyok*) is, amelyek a fogalmi modellből automatikusan adódnának. Másrészt éppen e korlátok miatt a tervezők egy része úgy érzi, hogy nincs is sok értelme törődni a tényezők minőségével, hiszen a kezelő ugyanúgy bánik például egy családfával, mint bármilyen más minden jellegzetesség nélküli egyeddel. A relációs rendszer képtelen felismerni és sajátosan kezelni az egyedaltípust, aminek pedig komoly matematikai alapja van (12.5 *Feltételes függés* és 12.6 *Generalizáció és specializáció*).

*Ebben a fejezetben az adatmodell egyelőre még mindig különlegesnek tartott sajátos szerkezeti és minősítési aspektusait fogjuk ismertetni.*

Az adatmodellezés legfontosabb feladata a valóság hű tükörképének a megkeresése. A modell nem teljesen azonos az adatbázistervvel. A tervezés pedig nemcsak a struktúrának a felépítését jelenti, hanem abba beleértendő a korlátok meghatározása is. Mármost ha a rendszer nem képes például a visszamutató kapcsolat kezelésére, tehát az ehhez a sajátos szerkezethez tartozó speciális korlátok automatikus betartatására, akkor azt magunknak kell megtennünk programmal. A kezelő gyengesége nem jogosít fel bennünket arra, hogy az integritási feltételeket elhanyagoljuk. Ezért fogalmi szinten akkor is törekednünk kell a minőségi modellezésre, ha logikai/fizikai szinten a minőség „kiveszik” a tervünkből.

Fejezetünk két sajátos konstrukciót is bemutat (12.7 *Unáris egyed* és 12.9 *Szinguláris egyed*). Az előbbi lehetőséggel arra hívjuk fel a figyelmet, hogy nem mindig kell a régi korszakban megszokott módon *kódokat* alkalmazni. A másik ritkán használt szerkezet. A nevezett konstrukció egyes speciális származtatott adatok következetes modellezését segíti elő (12.8 *Származtatott tulajdonságok*).

### 12.2 Többszörös inhomogén kapcsolat

+

Két egyedtípus akkor és csak akkor áll többszörös kapcsolatban, ha az egyik a másik azonosítójának több szerepnevét tartalmazza.

12.1 példa CÍM (Címkód, ...)

SZEMÉLY-E (Személy azonosító, Címkód, Címkód, ...)

SZEMÉLY-V (Személy azonosító, Lakcímkód, Levelezési címkód, ...)



Magyarázat: Néha előfordul, hogy két egyedet egynél több kapcsolattal kell egymáshoz kötni. Mivel a viszonyt **kapcsolótulajdonság** (Címkód) hordozza, az alárendelt egyedben többször kellene felvenni a kapcsoló szerepű tételt (SZEMÉLY-E). Azonban ezt kétszer nem tehetjük meg ugyanazzal a névvel, de még ha megtehetnénk akkor is inkább olyan neveket alkalmaznánk, amelyek a kapcsolatok minőségéről árulkodnak (SZEMÉLY-V). Tehát az alárendelt egyedben nem a fölérendelt kulcsa, hanem annak szerepnevei lesznek a kapcsolótulajdonságok, amiken a két egyed többszörös kapcsolata alapul.

**12.2 példa** VEVŐ (Vevőkód, ...)  
RENDELÉS (Rendelésszám, Rendelés vevőkód, ...)

Magyarázat: Ha a RENDELÉS egyedben két vevőre (pl. rendelő, fizető) kellene utalni, akkor kapcsolóként szerepnevet kellene alkalmazni. Itt azonban nem erről van szó. Sok tervezőnek rossz szokása, hogy a nevet akkor is minősíti (Rendelés-), ha az szükségtelen, ráadásul úgy, hogy a tervben nem is rögzíti, hogy szerepnévről van szó. A logikai szintű adatbázistervben nem kizárt a 12.2 példa megoldása. Viszont fogalmi szinten a kapcsolók felesleges minősítése csak megnehezíti az áttekintést és az elemzést, ezért kerülendő.

**12.3 példa** CÍM (Címkód, ...)  
SZEMÉLY (Személy azonosító, ...)  
SZEMÉLY/CÍM (Személy azonosító+Címkód, (Dátumtól, Dátumig))

Magyarázat: A többszörös kapcsolat olyan M:N fokú viszony, amiben az M értéke eleve korlátozott (a 12.1 példa esetében maximum két cím tartozhatott egy személyhez) és a viszonyt nem egy hálóban, hanem M darab hierarchiában tükrözzük. Ez két esetben nem jó megoldás. Ha az M értéke nem stabil - holnap lehet, hogy kettő helyett három viszonyt kell tükrözni -, akkor a többszörös kapcsolat helyett a viszonyt explicit hálóban, tehát egy harmadik kapcsolóegyedben át kell megvalósítani, amint azt a 12.3 példa mutatja. Az új struktúrában az M értéke már szabadon, szerkezeti változtatás nélkül módosítható, amit nem tehattünk meg a többszörös kapcsolat esetében.

A másik eset az ismétlődésekkel függ össze. Modellezési szempontból nagy eltérést jelent, hogy ismétlődő **csoportról** vagy csak ismétlődő **elemi adatról** van-e szó. A 12.1 példa esetében semmit sem akartunk elmondani a személyek és a címek viszonyáról, a cím hivatkozása csak elemi ismétlődő adat volt. Ámde abban a pillanatban, hogy arra is kíváncsiak vagyunk, milyen dátumtól fogva illetve milyen dátumig érvényes a cím, már ismétlődő csoportról van szó (Címkód, Dátumtól, Dátumig). Az ismétlődő adatokat pedig ki kell venni az egyedből (← Nem-normalizált egyed). Az előnormalizálással a 12.3 példa megoldásához jutunk el.

Kiegészítés: A dolgot másként is fel lehet fogni. Definíciónk szerint ha egy jelenséget ismeretekkel akarunk leírni, akkor azt egyeddel tükrözzük. Ha tehát ismeretekkel kell leírni a címek és a személyek viszonyát, akkor nem kapcsolatról, hanem egyedről van szó (SZEMÉLY/CÍM).

Probléma: A legtöbb esetben a tervező nem tudhatja előre, hogy a többszörözés mértéke nem változik-e az idők során illetve a viszonyt nem kell-e majd saját adatokkal ellátni. Ezért a többszörös kapcsolatot csak nagyon ritkán alkalmazzuk és többnyire egy harmadik kapcsolóegyeddel váltjuk ki.

Kiegészítés: Több oka van annak, hogy ezt a sajátos struktúrát mégis ismertettük. Egyrészt hasonló elveken alapul a már tárgyalt *családfa*, amire még ebben a fejezetben is vissza fogunk térni (→ *Homogén kapcsolat*). Másrészt előfordulnak esetek, amelyekben a többszörös kapcsolat sikerrel alkalmazható (több mértékegység, több településre történő hivatkozás a viszony részletezése nélkül stb.). Végül a jó modellező gyakran alkalmaz általános célú egyedeket is. Ilyen például a DÁTUM (*Dátum*) tábla. Evidens, hogy sok olyan egyéb egyedünk lesz, amiben több keltezés is szerepel. Tehát ezek az egyedek többszörös kapcsolatokkal fűződnek a DÁTUM egyedhez (vö. SZEMÉLY/CÍM).

## 12.3 Szerepnevek függései

+ **Ha az A' tulajdonság az A tulajdonság korlátozó szerepneve, akkor azt alkalmazfüggéssel határozza meg.**

**12.4 példa** CÍM (*Címkód*, ..., *Helységnév*)  
SZEMÉLY (*Személy azonosító*, *Címkód*, *Levelezési címkód*)

Magyarázat: Korlátozó szerepnévről akkor beszélünk, ha a szerepnév az értéktartomány valós részhalmazát jelenti. Nem minden Címkód érték egyben Levelezési címkód érték is, ezért az utóbbi tétel az előbbi korlátozó szerepneve.

Vigyázat! Figyeljünk az azonos értelmezésre. Ha a lakcím utca szerinti, a levelezési cím viszont postafiókot jelent, akkor a kettő nem származhat azonos értéktartományból, azaz nem lehetnek egy közös cím-lényeg szerepnevei!

Probléma: A SZEMÉLY egyedben fennáll a Személy azonosító → Levelezési címkód → Címkód függéssor. Az utóbbi függés definíciónk szerint alkalmaz jellegű, de ez nem változtat a tranzitivitás tényén. Tehát az egyed - látszólag - meg kellene bontani. Ámde valójában csak arról van szó, hogy a tervező „slampos” tervet készített. Elhagyva a minősítést, a Lakcímkód helyett Címkódot írt. A Lakcímkód maga is korlátozó szerepnév, ami már nem áll függésben a Levelezési címkóddal és nem okoz tranzitivitást. Az ilyen tévedések elkerülése érdekében be kell tartani a következő szabályt:

— **Az egyed típusban csak akkor szerepelhet egy elsődleges tulajdonság és annak szerepneve, ha a kettő közül az előbbi az egyed azonosítója.**

Magyarázat: A következő pontban majd kimutatjuk, hogy ha a szabályban lévő feltétel fennáll, akkor nem lép fel normalizálási probléma.

**12.5 példa** CÍM (*Címkód*, ..., *Helységnév*)  
SZEMÉLY (*Személy azonosító*, *Lakcímkód*, *Levelezési címkód*, ..., *Helységnév*)  
- a Helységnév a lakcímmel kapcsolatos

Probléma: A többszörös kapcsolat szintézises normalizálási eljárással nem hozható létre, hiszen a tartományok (Személy azonosító és Címkód) között csak egyféle viszony határozható meg (van

függés - nincs függés) és az nem minősíthető. A dekompozíció sem tud mit kezdeni a 12.5 példával, vagyis nem tudja feltárni a Helységnév tranzitivitását.

Megoldás: Modellezési módszerünkben a szerepneveket tudatosan alkalmazzuk. A mi normalizálási eljárásunkban ezért felfedezzük a függéseket a szerepnevek esetében is. Így például azt, hogy a Lakcímkód  $\rightarrow$  Címkód  $\rightarrow$  Helységnév függéssor miatt a SZEMÉLY egyedből ki kell emelni a Helységnév tulajdonságot.

Példák: A RENDELÉS (**Rendelésszám**, *Rendelés vevőkód*, Vevőnév) egyedben éppúgy fennáll a tranzitivitás, mintha a vevő kulcsa nem szerepnévként, hanem eredeti Vevőkód alakjában jelenne meg. A RENDELÉSTÉTEL (**Rendelésszám**+**Cikkszám**, Tétel cikknév) egyedben is látható a részleges függés, noha abban a Cikknév szerepnevét alkalmaztuk.

#### 12.6 példa BÍRÁLAT (Bírálat azonosító, ..., Bizottsági tag kód, Bírálokód)

Magyarázat: A bizottsági tag és a bíráló egyaránt személy. Tegyük fel, hogy minden bíráló csak bizottsági tag lehet, de nem minden bizottsági tag lehet bíráló. Látszólag úgy néz ki, hogy a példa a Személy azonosító két leíró szerepnevét tartalmazza, tehát az egyed - legalábbis formálisan - jól tervezett.

Probléma: Az egyedbeli redundancia szembeötlő. A Bírálat azonosító  $\rightarrow$  Bírálokód  $\rightarrow$  Bizottsági tag kód tranzitivitás miatt az utóbbi tételt el kell hagynunk az egyedből. (NB.: Ezt persze csak azután tehetjük meg, miután meggyőződünk arról, hogy a két tétel értéke mindig azonos. Tehát nem arról van szó, hogy pl. egyéb bizottsági tagok valamiféle külön - nem bíráló - szerepet játszanak a bírálatban.) A megoldás a szerepnevek hierarchiáján alapul. Egy alaptulajdonságnak (Személy azonosító) létezhet olyan szűkítő szerepneve (Bizottsági tag kód), aminek van további korlátozó szerepneve (Bírálokód). Az utóbbi alkalmazfüggéssel határozza meg az előbbi.

## 12.4 Homogén viszonyok

+

Az egyed akkor és csak akkor áll visszamutató viszonyban önmagával, ha azonosítója funkcionálisan meghatározza saját szerepnevét.

- 12.7 példa SZEMÉLY-R (Személy azonosító, ..., Személy azonosító)  
SZEMÉLY-1 (Személy azonosító, ..., Házastárs azonosító)  
SZEMÉLY-2 (Személy azonosító, ..., Főnök azonosító)

Magyarázat: A rekurzív viszonyról már volt szó ( $\leftarrow$  *Visszamutató kapcsolat*). Ebben a pontban már csak a szerepnevek fontosságát kell kiemelni. A különböző egyedek közti inhomogén hierarchikus viszonyokat kapcsolótulajdonságok hordozzák. Az lenne jó, ha így lehetne megalapozni a homogén hierarchikus viszonyokat is (SZEMÉLY-R). Ámde egy egyedben nem adható meg kétszer ugyanaz a tulajdonságnév, no meg az nem is fejezi ki a viszony minőségét (első egyed). Ezért hasznos kényszerrel szerepneveket fogunk bevetni, amint azt a másik két egyedünk mutatja.

Jellemzők: Természetesen ez a két egyed csak akkor jelent korrekt megoldást, ha a modellben expliciten rögzítjük, hogy a Házastárs azonosító és a Főnök azonosító maga is Személy azonosító, vagyis az utóbbi tulajdonságnak a szerepneve. Ha európai házasságot feltételezünk, akkor a SZEMÉLY-1 egyed 1:1 fokú lineáris viszonyt implikál. Azt külön korlátban kell rögzíteni, hogy a viszony kötelező-e (csak házasságokról vezetjük a nyilvántartást). A SZEMÉLY-2 egyed 1:N fokú hierarchikus viszonyt von maga után úgy, hogy az csak kétoldaltól opcionális lehet.

#### 12.8 példa HÁZASSÁG (Személy azonosító+Házastárs azonosító)

Probléma: Az előző pontban kijelentettük, hogy egy egyedben csak akkor szerepelhet együtt egy tulajdonság és annak szerepneve, ha az előbbi az egyed kulcsa. Az E ( $A+A'$ ) egyed, ahol  $A'$  az A szerepneve, rosszul tervezett. Azért az, mert az összetett kulcs tagjai mindig hálós viszonyban kell, hogy álljanak, azaz nem lehet közöttük függés. Márpedig a szerepnév meghatározza elsődlegesét. Ezért a példa egyede *belső kulcstörő függést* mutat, vagyis normalizálandó.

#### 12.9 példa HÁZASSÁG (Házastárs azonosító férj+Házastárs azonosító feleség)

Magyarázat: Most a két kulcsrész mindkettője szűkítő szerepnév. Két eset lehetséges. Ha csak az aktuális házasságokat vezetjük, akkor a kettő között létezik függési viszony (kölsönös függés) és elvileg más megoldást kell alkalmazni. Ha viszont a mindenkori házasságokat tartjuk nyilván, akkor nincs formális probléma sem. Az előbbi határesetben még a szerző is engedményt tenne a gyakorlat kedvéért az elmélet rovására, vagyis nem kényszerítene ki más megoldást. Főleg akkor nem, ha a Házassági dátum tulajdonságot is felvesszük. Ugyanis abban az esetben már lebonthatatlan relációt kapnánk.

#### 12.10 példa TÉTEL (Tétel azonosító, ...)

CSALÁDFA-T (Családfa az, Fölé tétel az, Alá tétel az)

CSALÁDFA-D (Fölé tétel az+Alá tétel az+Dátum)

Magyarázat: A családfa szerkezetről már beszéltünk. A visszamutató M:N fokú hálós viszonyt egy sajátos kapcsolóegyeddel - a családfával - tükrözzük. Azt feltételeztük, hogy az ilyen egyed kulcsa az alapegyed azonosítójának két szerepnevéből alkotott csoport. A szerkezethez más korlátokat is kötöttünk. Ezért bár a viszony átalakítható többszörös kapcsolattá is (CSALÁDFA-T), abban az esetben a korlátokat külön-külön kell megadni, ezért általában az ilyen megoldás nem kívánatos. Az viszont sokszor előfordulhat, hogy dinamikus családfát (CSALÁDFA-D) kell alkalmaznunk, mert a fölérendelt tétel ma még ezekből, holnap már azokból az alárendelt tételekből épül fel. Ilyenkor az érvényesség kezdetét vagy végét jelző dátumot/időpontot is az összetett kulcs részévé kell tenni.

#### 12.11 példa SZEMÉLY (Személy azonosító, ..., Főnök azonosító)

FŐNÖK (Főnök azonosító, ..., Személy azonosító)

Probléma: A példán azonnal látható, hogy ciklust tartalmaz. Ha egy egyednek a kulcsa szerepnév, akkor abban nem szerepelhet az elsődleges tulajdonsága is. Mert nem minden személy főnök, de minden főnök személy. Tehát nem minden Személy azonosító egyben Főnök azonosító is, de fordítva ez a helyzet. Vagyis ha a második egyedben a személy kulcsa a főnökre utal, akkor az durva redundancia (azonos értékek). Ha pedig nem, akkor a terv azért rossz, mert nem minősíti az egyetlen (!) a főnökhöz kapcsolódó személyt.

+

***A modellezésben mindig az általánosabb eset a mérvadó.***

**12.12 példa** SZERZŐDŐ (Szerződkód, ...)  
 SZERZŐDÉS-1 (Szerződésszám, ..., Szerződkód-1, Szerződkód-2)  
 SZERZŐDÉS-2 (Szerződésszám, ...)  
 SZERZŐDÉS/SZERZŐDŐ (Szerződésszám+Szerződkód)

Magyarázat: Természetesen a szerződő a partner egyik altípusa, de ez most mellékes. A példa arról szól, hogy vannak szerződéseink, amikben az esetek 95%-ában csak egyetlen szerződő partner szerepel. Igen speciális esetben a szerződést egy házaspár köti, teljesen azonos jogokkal és kötelezettségekkel. A szerződésért együtt felelnek és annak előnyeit együtt élvezik. Sohasem fordul elő, hogy a szerződés kettőnél több személyt érintene.

Megoldás: Valaki úgy gondolkodik, hogy mivel a szerződők száma maximum kettő, a SZERZŐDÉS-1 jelenti a jó megoldást ( $\leftarrow$  *Többszörös kapcsolat*). Mások szerint mindig a fenti szabályt kell szem előtt tartani. Ha két jelenség viszonya „többnyire” 1:N fokú, de kivételes helyzetben M:N fokú, akkor a szabály értelmében az utóbbit kell mérvadónak tekinteni. Ezért a SZERZŐDÉS-2 és a külön kapcsolóegyed jelenti a helyes megoldást. Ez akkor is megállja a helyét, ha az érintettek száma csak kettő.

Kiegészítés: *Technikai* értelemben a példa mindkét megoldása helyes. Tekintettel arra, hogy a dolgok változnak, a többszörös kapcsolat kevésbé ajánlott megoldás. Azért az, mert ma még nem kell adatokkal leírni a szerződés/szerződő viszonyát, viszont holnap már lehet, hogy erre szükség lesz és akkor csak a második megoldás állja meg a helyét. Ebben a verzióban méltatandó az is, hogy a tervező figyelembe vette a vonatkozó szabályt. Ennek dacára *szemantikai* értelemben az a megoldás sem jó.

+

***Több expliciten kifejezhető tényt nem szabad impliciten egy tényezőben tükrözni.***

**12.13 példa** SZERZŐDŐ (Szerződkód, ...)  
 SZERZŐDÉS (Szerződésszám, ..., Szerződkód)  
 PÁROK (Szerződkód-1+Szerződkód-2)

Magyarázat: Ha a szerződést házaspár köti, akkor nem a két személy, hanem a valós helyzetnek megfelelően maga a házaspár a szerződő fél! A fenti megoldások egyike sem tükrözi magának a házasságnak a tényét. (Az első esetben nyugodtan bevihető például két férfi azonosítója a SZERZŐDÉS-1 egyedbe.) Ezért azok a tervek szemantikai értelemben hiányosak illetve pontatlanok. Ugyanis valójában három partnerről van szó. Magában a szerződésben a házaspár szerepel. A pár két tagját pedig két okból külön is felvesszük. Egyrészt azért, hogy kifejezhessük kapcsolatukat (PÁROK). Másrészt azért, mert világos, hogy Kovács nemcsak Kovácsnéval Kovács házaspárként köthet szerződést, hanem ezt megteheti - más szerződésben - egyedi, egyéni szerződként is.

Kiegészítés: Azok számára, akiknek netán visszásnak tűnik a példa megoldása, legyen szabad csak így szövegesen egy másik esetet is bemutatnunk. Az autógyárak nemcsak kész gépkocsikat adnak el, hanem szerelvényeket - motorokat, sebváltókat stb. - is. Az eladási szerződés a motorra

(vagyis egy termelési tételre) vonatkozik és nem azt írják a szerződésbe, hogy eladnak x darab szelepet, y darab cilindert stb. Azt, hogy miből épül fel a motor, a darabjegyzék tükrözi (vö. PÁROK). Az, hogy eladunk valamit egy tény. Az, hogy az eladott dolog miként tevődik össze másokból, egy másik tény. A 12.13 példa a két expliciten kifejezhető tény két explicit tényezőben adta meg. Szemben a 12.12 példa két megoldásával, amiben a felépülés csak impliciten illetve még úgy sincs jelen.

## 12.5 Feltételes függés

+

**Az E egyed A kulcsa akkor és csak akkor határozza meg feltételes függéssel az egyed B tulajdonságát, ha létezik olyan (implicit) C tétel, aminek adott tartalma(i) esetében a B mindig NA-értéket vesz fel.**

**12.14 példa** SZEMÉLY (Személy azonosító, Foglalkozás, Orvoscód, Rendelőkörzet)

Magyarázat: A korábbiakban már volt szó az *üres értékről*. Ha valakinek a foglalkozása nem orvos, akkor nála az Orvoscód értéke nem üres vagy ismeretlen, hanem értelmetlen, nem alkalmazható [not applicable]. Az angol rövidítés nyomán ezt hívjuk *NA-értéknek*. Ezt - szemben az összes egyéb modellezési eljárással - mi valós értéknek tekintjük, mert csak így van módunk bizonyos sajátos függések vizsgálatára.

Kiegészítés: A feltételes függés jele  $A \rightarrow C(i) \rightarrow B$ , ahol C a feltételtényező és i maga a konkrét feltétel. Esetünkben fennáll a Személy azonosító  $\rightarrow$  Foglalkozás(nem orvos)  $\rightarrow$  Orvoscód feltételes függés. (Impliciten akkor is létezne, ha nem volna jelen a Foglalkozás tulajdonság expliciten is. Hiszen csak az orvosoknak lehet Orvoscód értékük.) A feltételes függés a tranzitív függés speciális esete. Ha a foglalkozás értéke orvos, akkor nincs gond. Ha viszont nem orvos, akkor tudjuk, hogy az Orvoscód és a Rendelőkörzet NA-értékű.

Probléma: A feltételes függés nem okoz súlyos fogalmi szintű modellezési gondokat. Az úgynevezett *üresredundanciát* többnyire hatékonysági/kezelési okokból - logikai/fizikai szinten - tartják kívánatosnak feloldani. Az üresredundancia a fizikai átfedésnek az a sajátos esete, amiben egy meghatározott feltétel esetén az egyed tulajdonságsorának adott részsora mindig kitöltetlen. Ez adott kezelőrendszer esetén tárolási anomáliát okoz.

**12.15 példa** SZEMÉLY (Személy azonosító, Foglalkozás)  
 ORVOS-J (Személy azonosító orvos, Orvoscód, Rendelőkörzet)  
 ORVOS-R (Orvoscód, Személy azonosító, Rendelőkörzet)

Magyarázat: Ha a feltételes függés alternáló, akkor specializációt alkalmazunk. Itt erre nincs szükség, mert nem beszéltünk az egyéb foglalkozásokról. Ennek ellenére az eredeti egyedet megbonthatjuk a feltételes függés mentén és akkor úgynevezett *kiegészítő egyedet* kapunk. Az ORVOS a SZEMÉLY kiegészítő egyede, mert foglalkozás = orvos értéknél többletismereteket ad számunkra a személyről.

Probléma: Az orvos is személy. Ezért az ORVOS egyed kulcsa is a személy szerepneve kell, hogy legyen. A fenti kétféle megbontás közül az első jó, a második pedig nem az. Az ORVOS-R

egyeddel nemcsak az a baj, hogy Orvoscód kulcsa a személytől függetlenül is módosítható (elveszik a pecsét és újat adnak ki), hanem az is, hogy 1:M fokú kapcsolatot mímel. Ami pedig azzal jár, hogy több orvoscód mellé is megadható ugyanaz a Személy azonosító. Márpedig evidens, hogy egy személy nem lehet több orvos.

## 12.6 Generalizáció és specializáció

+

**Specializációnak nevezzük azt a műveletet, aminek során az eredeti egyed előfordulás (rész)sorait az alternáló feltételes függés mentén altípus egyedekbe helyezzük.**

**12.15 példa** ERŐFORRÁS (Erőforrás azonosító, Személy, Anyag, Eszköz)

SZEMÉLY (Személy azonosító, Személy)

ANYAG (Anyag azonosító, Anyag)

ESZKÖZ (Eszköz azonosító, Eszköz)

Magyarázat: Az eredeti egyed 5NF alakban van, mivel leíró tulajdonságai egymástól függetlenek. (Mindegyik leíró valójában több tulajdonságot takar.) Mégis érezzük, hogy az ERŐFORRÁS végletesen rosszul tervezett egyed. Azért, mert ha az erőforrás személy jellegű, akkor az anyag és az eszköz adatai NA-értékűek. Ha anyag, akkor a személy és az eszköz adatai nem értelmezhetőek. Az NA-értékek azt mutatják, hogy a valós jelenségeket nem helyesen tártuk fel.

Megoldás: A látható szemantikai problémának matematikai alapja is van. Létezik egy implicit Erőforrástípus tulajdonság, amitől feltételesen függ az összes többi, még hozzá alternáló módon. Ha az Erőforrástípus = „személy”, akkor a Személy adatok kitöltöttek és a többi nem és így tovább. Ezért az eredeti egyed a három mutatott részre lehet bontani, vagyis el lehet végezni annak specializációját.

**12.17 példa** SZEMÉLY (Személy azonosító, Státus, ...K, T, D, E)

SZEMÉLY (Személy azonosító, Státus, ...K)

TANÁR (T-személy azonosító, T)

DIÁK (D-személy azonosító, D)

EGYÉB (E-személy azonosító, E)

Magyarázat: Most a T a tanár, a D a diák, az E az egyéb személyek olyan adatai helyett áll, amelyek a Státus értékétől feltételesen függve vesznek fel NA-értékeket. A K a közös, feltételesen nem függő tulajdonságok részsora. A specializációt most ugyanúgy hajtjuk végre, mint az előbbi példánál, de az eredményben van némi eltérés. A specializáció lehet *teljes*, amikor az eredeti egyed megszűnik, feloldódik az utódjaiban (lásd előző példa). Azonban a specializáció a legtöbb esetben *részleges*, vagyis megmarad az eredeti egyed is a feltételesen nem függő tulajdonságokkal.

Kiegészítés: Specializációval születnek az eredeti egyed *altípusai*. Például a TANÁR a SZEMÉLY főtípus altípusa. Az altípust elvileg a főtípus kulcsa azonosítja. Viszont két egyednek nem lehet azonos a kulcsa. Ezért ilyenkor az eredeti azonosító korlátozó szerepnevét használjuk. Minden tanár személy, de nem minden személy tanár. Ez a szerepnév - feladata miatt - nem tévesztendő össze a pusztán minősítő feladatúval. A tulajdonképpeni specializáció mindig *alternáló*. Ez azt jelenti, hogy az eredeti egyedtípus megbontásával több, egymást előfordulásaikban kizáró specializált egyedaltípust kapunk. A tanár nem lehet diák és a diák nem lehet tanár, különben fizikai redundancia lépne fel, ami karbantartási anomáliákat okozna. Ha tehát netán van olyan diák, aki egyben tanár is, akkor a T és a D ismeretsorok tételei közül azokat, amik egyszerre jellemezhetnek egy tanár-diák személyt, nem szabad kiemelni az eredeti egyedből.

A részleges specializációval nyert altípusok a főtípussal (SZEMÉLY) alulról kötelező, felülről opcionális 1:1 fokú „is-egy” jellegű kapcsolatban állnak a kulcsukon keresztül. A tanár is egy személy, a diák is egy személy. Az altípus a főtípus tulajdonságait „örökli”. Ebből következik, hogy az altípusnak és a főtípusnak nem lehet közös tulajdonsága. Az altípusok között is kizárt az ilyen redundancia. Ha több altípusnak közös tulajdonsága van, akkor azt az eredeti egyedben - esetünkben a K ismerethalmazban - kell hagyni.

**12.18 példa** TANÁR (T-személy azonosító, T, K)  
 DIÁK (D-személy azonosító, D, K)  
 EGYÉB (E-személy azonosító, E, K)

SZEMÉLY (Személy azonosító, Státus, ...K)

Magyarázat: A specializáció fordított művelete az általánosítás, a *generalizáció*. Ez is lehet teljes illetve részleges. Példánk az utóbbira mutat esetet. Ha az eredetileg tervezett három egyednek létezik egy közös (K) ismerethalmaza, továbbá az egyedek kulcsai vagy egy közös tulajdonság (Személy azonosító) szerepnevei vagy azzá tehetők, akkor van mód az általánosítás végrehajtására. A (részlegesen) generalizált új egyedben általában explicit módon fel szoktuk venni az eredeti egyedekre - immár altípusokra - utaló tételt (Státus).

Kiegészítés: Egy egyednek számos NA-értékű tulajdonsága lehet. Például a szögletes alkatrészek esetében az Átmérő tartalma nem nulla, hanem NA-értékű. Ettől még nem fogunk kerek és nemkerek alkatrész egyedaltípusokat kreálni. Nincs általános szabály arra nézve, hogy mikor *kell* specializálni, mert az üresredundancia kiküszöbölése *lehetőség* és nem kényszer. Gyakorlati elv az, hogy ha egy egyedtípus tulajdonságainak a harmada NA-értékű, akkor már érdemes a specializáción gondolkodni.

Megjegyzés: A korábbiakban már futólag volt szó a generalizációnak egy igen sajátos, kizárólag modellezéstechnikai célú alkalmazásáról (← *Álegyek*).

## 12.7 Unáris egyed

Meghatározás: Az unáris egyed olyan csupakulcs reláció, aminek a kulcsa elemi. Vagyis az ilyen egyed egyetlen tulajdonságából - az elemi azonosítóból - áll.



**12.19 példa** SZEMÉLY (Személy azonosító, ...)  
 NYELV (Nyelvkód, Nyelv)  
 NYELVTUDÁS (Személy azonosító+Nyelvkód, ...)

Magyarázat: Tulajdonképpen ezzel a példával nincsen semmi baj. Korrekten tükrözi a személyek és a nyelvek M:N fokú hálós viszonyát. Azonban véleményünk szerint az eset mégis tartalmaz egy sajátos redundanciát. Nevezetesen a Nyelvkód és a Nyelv kölcsönös függésben áll egymással és ezért mindkettő alkalmazása feleslegesnek tűnik.

**12.20 példa** SZEMÉLY (Személy azonosító, ...)  
 NYELV (Nyelv)  
 NYELVTUDÁS (Személy azonosító+Nyelv, ...)

Magyarázat: Most a NYELV unáris egyed. Ez a megoldás *fogalmi* szinten pontosan olyan jó, mint az előbbi. Azért az, mert a nyelv mint természetes megnevezés sohasem változik. Ezért a Nyelvkód/Nyelv páros nem hasonlít a Vevőkód/Vevőnév párosra, hiszen az utóbbiban a Vevőnév változó. Tehát a 12.19 megoldás használatának csak a méret az oka, az pedig *fizikai* szintű tényező. Ezért jóllehet gyakorlatilag nem zárjuk ki azt a verziót sem, elméletileg nem látjuk megalapozottnak.

Kiegészítés: Az unáris egyed valójában nem más, mint egy *értéktartomány*. Ha magát a nyelvet nem akarjuk valós ismeretekkel leírni, akkor a NYELV egyed látszólag csak az értékek validálására szolgál. Valójában azonban van egy további célja is.

**12.21 példa** SZEMÉLY (Személy azonosító, ..., Költséghely)  
 GÉP (Gép azonosító, ..., Költséghely)

Magyarázat: Tegyük fel, hogy a költséghelyeket nem akarjuk külön ismeretekkel leírni. Az adott példa ekkor korrektnek látszik, hiszen a Költséghely értékét doméjn alapján is lehet validálni. Azonban így egy kapcsolhatatlan modellt kapunk, hiszen - mint tudjuk - két egyed között a leíró tulajdonságaikon nem lehet kapcsolatot meghatározni. A modell formailag teljessé és elemezhetővé válik (ne feledjük, hogy nem csak erről a két egyedről lehet szó), ha bevezetjük a KÖLTSÉGHELY (Költséghely) egyedet is.

Kiegészítés: Unáris egyedekre gyakrabban van szükség, mint azt gondolnánk. Például a MUNKANAP vagy akár maga a DÁTUM - az előbbi főtípusa - is sokat segíthet az egyedek kapcsolatainak a tisztázásában, éppen úgy, mint a kapcsolat szerinti validálásban (vö. hivatkozási integritás).

## 12.8 Származtatott tulajdonság

+

A közlés vagy megfigyelés útján nyert ismeretet alaptulajdonságnak hívjuk. A származtatott tulajdonság az előbbieken, más származtatott tulajdonságokon és konstansokon végrehajtott matematikai és/vagy logikai műveletek eredménye.

Magyarázat: Az alap- és származtatott tulajdonságot a modellezésben nem egy adott eljáráshoz képest (relatívan), hanem az eljárásoktól függetlenül (abszolútan) értelmezzük. Például a rendelés összértékét a tételértékeinek a szummájaként nyerjük. A Rendelésérték =  $\Sigma$  Tételérték műveletben (relatívan) a Tételérték „alaptényezőnek” tűnik, de számunkra modellezési szinten (abszolútan) mégsem az, mert maga is származtatott: a Mennyiség és az Ár szorzata. A Mennyiség közlés útján nyert alaptulajdonság. Nem kizárt, hogy az Ár tulajdonságot valahol másutt úgy számítják ki. Ha ezt nem a mi rendszerünkben tesszük, akkor az Ár is közlés útján nyert alaptulajdonság.

Kérdés: Szabad-e a származtatott tulajdonságokat modellezni?

Válasz: A szerzőnek mindig az előbbi kérdést vetik fel, de valójában arra gondolnak, hogy szabad-e a származtatott tulajdonságokat tárolni? Ez utóbbi hatékonysági, tehát nem fogalmi, hanem logikai/fizikai szintű probléma. Ezzel szemben minden az ismeretről szóló tudásunkat (metaismeret) rögzítenünk kell már a fogalmi szinten. Tehát a válasz az, hogy nemhogy szabad, hanem kell. Tulajdonságstruktúrákkal ( $\leftarrow$  *Metastruktúra*) fejezzük ki, hogy a származtatott tulajdonságok miképpen épülnek fel más tulajdonságokból. Ezen túlmenően viszont arra is kíváncsiak vagyunk, hogy *ha tárolni akarnánk majd* az ilyen tényezőket, akkor azokat melyik egyedhez kellene - ha lehetne - kapcsolni?

Esetek: Van ún. intenzionális és extenzionális származtatás. (A kettő kombinációjára nem térünk ki könyvünkben.) Az **intenzionális származtatás** a tulajdonságokon alapul. Például a RENDELÉSTÉTEL Tételérték adatának a tartalmát a Tételérték = Mennyiség \* Ár képlet alapján számoljuk ki. Az ilyen művelet sajátossága az, hogy a származtatott tulajdonság a tényezőket szolgáltató egyedek valamelyikét jellemzi. Pontosan meg tudjuk mondani azt is, hogy melyiket. Azt aminek kulcsa a tényezőket tartalmazó egyedtípusok kulcsainak a **legkisebb közös többszöröse**. Ezt a fogalmat most nem a szoros matematikai értelemben használjuk. Hanem úgy, hogy a meghatározó kulcs a származtatási tényezők kulcsainak a minimális konkatenációja.

Ha az Ár rendeléstételenként változhat, akkor mindkét tényező a RENDELÉSTÉTEL-ből származik, tehát azt fogja jellemezni a Tételérték is. Ha egységárról van szó, akkor az Ár a CIKK-ből származik. A RENDELÉSTÉTEL kulcsa a Rendelésszám+Cikkszám, a CIKK kulcsa a Cikkszám, a közös többszörös az előbbi. Tehát a Tételérték logikus helye ekkor is a RENDELÉSTÉTEL-ben van.

Az **extenzionális származtatás** az egyedelőfordulásokon alapul. A Rendelésérték =  $\Sigma$  Tételérték képlet alapján számoljuk ki a rendelések értékeit. Tehát olyan halmazműveletet alkalmazunk, amiben a „halmaz” tagjai a (szelektíven) kiválasztott egyedelőfordulások adott tulajdonságra vonatkozó értékei. Az ilyen származtatásban az eredménytulajdonság kulcsa a tényezők kulcsainak a **legnagyobb közös osztója**. A tényezők közé számít maga a kiválasztási ismérv is (esetünkben ez a Rendelésszám).

A kiinduló RENDELÉS egyed kulcsa a Rendelésszám. A RENDELÉSTÉTEL kulcsa a Rendelésszám+Cikkszám konkatenáció. Ezért a Rendelésértéknek az előbbi egyedben van a megfelelő helye. Ha azonban nem egyetlen, hanem az összes rendelés együttes értékére vagyunk kíváncsiak, akkor az Összérték =  $\Sigma$  Rendelésérték képletet kell alkalmaznunk. Ebben az esetben nincsen olyan kiinduló egyedünk, ami tartalmazná a szelekciós tételt, hiszen nem is történik kiválasztás, mert az összes előfordulásról van szó. Ennek a sajátos helyzetnek a megoldását a következő pontban adjuk meg.

— **Származtatás mentén sohasem kreálunk egyedet és nem normalizálunk.**

**12.22 példa** RENDELÉSTÉTEL (**Rendeléstétel az, ..., Mennyiség, Ár, Tételérték**)  
TÉTELÉRTÉK (**Mennyiség+Ár, Tételérték**)

Magyarázat: A származtatások sok tervezőt becsapnak. Ők örömmel fedezik fel, hogy a származtatás funkcionális függés jelleget ölt: Mennyiség+Ár  $\rightarrow$  Érték, és annak alapján létrehoznak a példához hasonló „egyedeket”. Ráadásul ha mindhárom tényező egyaránt a RENDELÉSTÉTEL-ben található, akkor annak mechanikus dekompozíciójával valóban a fenti eredményre jutnak.

Probléma: A normalizálás nem cél, hanem eszköz. Nem azért adatmodellezünk, hogy normalizáljunk, hanem azért normalizálunk, hogy javítsuk az adatmodellt. A 12.22 példa megoldása több ponton is ellentmond az adatmodellezési elveknek. (Arról nem is szólva, hogy a TÉTELÉRTÉK reláció végtelen nagyméretű lenne a kombinációk miatt). Egyrészt csak az ismeretekkel leírandó jelenségeket tekintjük egyedeknek. A matematikus számára egy képlet valóban lehet ismeretekkel leírandó jelenség, az adatmodellező részére viszont semmiképpen nem az. Másrészt az adatbázis egyik legfontosabb követelménye, hogy a kulcs - illetve értelemszerűen annak bármelyik része - sohasem változhat. Ha a fenti „normalizálást” végrehajtánánk a RENDELÉSTÉTEL-en és megváltozna például az Ár, akkor ezernyi helyen kellene a kulcsot módosítani, tehát karbantartási anomália lépne fel.

## 12.9 Szinguláris egyed

Meghatározás: A szinguláris egyed olyan technikai egyed típus, aminek egyetlen egy előfordulása van és kulcsa mesterséges fogalom.

**12.22 példa** ÖSSZVEVŐ (Összvevőkód, ..., Össztartozás)

Magyarázat: A valóságban léteznek olyan ismeretek, amik nem egy egyed típusnak az egyes előfordulásaihoz kapcsolódnak (mint például a Vevőkód, Vevőnév stb.), hanem magához az egyed-halmazhoz kötődnek. Minden egyes vevőnek lehet egyéni tartozása, amiknek a szummája az Össztartozás. Az ilyen halmazműveleten alapuló *származtatott tulajdonságok* nem köthetők az alapegyedhez (VEVŐ), hanem - ha egyáltalán modellezni akarjuk őket - alkalmazni kell egy származtatott egyedet (ÖSSZVEVŐ). Mivel ennek csak egy előfordulása van, nem tekinthető valójában típusnak és nincs szükség valódi - eltérő értékeket felvevő - kulcsra. Ezért az azonosító (Összvevőkód) teljesen mesterséges.

**12.21 példa** VÁLLALATUNK (Sajátkód, ..., Sajátnév, Sajátcím)

Magyarázat: Értelemszerűen saját vállalatunkból csak egy van. Ezért ha modellezni akarjuk, akkor azt *megoldhatnánk* szinguláris egyeddel is. A modellezésre pedig szükség lehet, mert például egy ügyletnek kapcsoltnak utalni kell a partner és a mi feladatainkra. A példa megoldását mégsem tartjuk célszerűnek. Azért nem, mert a partnerek ismereteinek a modellezésénél úgyis fel kell vennünk egy SZERVEZETI EGYSÉG egyedet a partner altípusaként, hogy a belső ügyletek ismereteit tükrözhessük. A külön szinguláris egyed helyett tehát célszerűbb megoldás, ha vállalatunkat is egy szervezeti egységnek tekintjük.

Kiegészítés: A szinguláris egyedet szinte kizárólag a halmazműveletek származtatott eredményeinek a modellezésére használjuk. Mivel az ilyen egyed normalizálása és egyéb elemzése nemcsak felesleges, hanem lehetetlen is, azt ajánljuk, hogy alkalmazzuk bátran ezt a konstrukciót, de ha elemzési segédeszközt használunk, annál ne vegyük fel az ilyen tényezőket az elemzendők közé.

## 13. AZ IDŐ MODELLEZÉSE

### 13.1 Dinamika az adatmodellben

Az eddigi fejezetekben az adatmodellt *statikusan*, az egyed, tulajdonság és kapcsolat háromszor kettő (típus és előfordulás) dimenziója mentén mutattuk be. Nem szenteltünk figyelmet az időtényezőnek, mint a *dinamika* alapjának.

Sokan úgy vélik, hogy az idő éppen úgy negyedik dimenzió a modellezésben is, mint a valós életben. Hamis nyomon járnak. Az időfaktor kérdésköre két részproblémára oszlik. Az egyik - a könnyebb - az, hogy miképpen lehet magát az időt modelltényezőként megfogni. A másik - a jóval nehezebb - az, hogy mit is jelent valójában az idő fogalma (13.3 *Az idő mint modell-tényező*).

Mielőtt ezeket a kérdéseket tisztáznánk, rá kell mutatnunk arra, hogy az idődimenzió bevezetésével vagy az egyedelőfordulások, vagy a tulajdonságtípusok köre bővül. A kétféle eset olykor nem zárja ki egymást (13.2 *Extenzió és intenzió*). Ennek dacára a modellben kerülni kell a dinamikus redundanciákat, ezért az idő kétféle (13.4 *Állapot és esemény*) modellezési lehetősége közül az egyik mellett kell letenni a voksot.

*Ebben a fejezetben az idő modellezésének számos kérdését vetjük fel. Példákon keresztül mutatjuk be a különböző megoldások előnyeit és hátrányait.*

Két lehetőséget ismertetünk (13.5 *Állapotmodellezés* és 13.6 *Eseménymodellezés*). Kimutatjuk, hogy bár elméletileg mindkettő megvalósítható, gyakorlatilag nem létezik problémamentes megoldás. Jelentősen segíthet a helyzeten a tulajdonság-általánosítás (13.7 *Inverzió*). Ez viszont azzal jár, hogy fel kell adnunk eddigi modellfilozófiánkat.

Mindaddig úgy tudtuk, hogy a modell egyedek, tulajdonságok és kapcsolatok valamint az ezekre vonatkozó korlátok szervezett együttese. Ez az alapkép nem is változik. Viszont revideálni kell azt a szemléletet, ami szerint az egyedek és a relációs táblák kölcsönösen megfelelnek egymásnak. Már a *specializáció* kapcsán is sejtettük, hogy ez nem igaz. A valós egyed (egy tanár személy) ismeretei egy egyedfőtípusban és egy egyedaltípusban foglalnak helyet úgy, hogy az altípus nem reprezentál külön valós jelenséget. Tehát ismét visszakanyarodunk az alapvető tételhez, ami szerint a valós egyed és annak modellbeli reprezentánsa - amit szintén egyednek hívunk - nem pontosan ugyanaz a dolog. A térkép nem azonos az ábrázolt földterülettel...

Fejezetünk sajátos dinamikus megoldások rövid ismertetésével zárul (13.8 *Idősorok*). Az alkalmat ott ragadjuk meg, hogy a dinamikus korlátokra is bemutassunk egy példát.

### 13.2 Extenzió és intenzió

+

Az extenzió és az intenzió fogalmi kiterjedést jelent. Az extenzió az egyed típus előfordulásainak, az intenzió a tulajdonságtípusoknak a fogalmi kiterjedése. A két kiterjedés szoros viszonyban áll egymással.

**13.1 példa** SZERVEZET (**Szervezet az**, Név, Cégbejegyzés száma)  
BANK (**Bank jel**, Név, ...)

SZERVEZET (**Szervezet az**, Név, Cégbejegyzés száma, Bank jel)

Magyarázat: Az egyedtípus a bennünket érdeklő jelenségek osztálya, aminek határait a tervező a modellezés szabályain belül a józan belátása szerint húzza meg. A határok kijelölése olyan értelmezést jelent, ami eldönti, hogy milyen valós jelenségeket ért a tervező a kérdéses osztályba és melyeket nem ért oda. Az egyedelőfordulások halmazának a fogalmi kiterjedése az *extenzió*. Ez elsősorban minőséget takar, de persze mennyiségi vonzattal is jár. Példánkban az első tervező a bankot nem sorolta a szervezetek közé. Az most lényegtelen, hogy megoldása nem jó, mert elszalasztotta a generalizáció lehetőségét. A lényeg az, hogy ő a második tervezővel szemben nem érti a SZERVEZET extenziójába a bankokat (minőség). Ennek következtében ennek az egyednek kevesebb (mennyiség) az előfordulása az első megoldásban, mint a másodikban.

Az extenzió szoros kölcsönhatásban áll a tulajdonságok fogalmi kiterjedésével, vagyis az *intenzióval*. Az intenzió az értelmezendő tulajdonságtípusok sora, ami ismét lényegileg minőség, de van mennyiségi vonzata is. Ha a bankokat kizárjuk a szervezetek közül, akkor az utóbbiakat nem jellemezheti a Bank jel. Ha odaértjük őket, akkor ezzel a tétellel bővül a SZERVEZET egyed tulajdonságsora.

— **A modellezésben háromféle absztrakciót alkalmazunk: normalizálást, specializációt és aggregálást.**

**13.2 példa** RENDELÉS (**Rendelésszám**, Vevőkód, Vevőnév)

Magyarázat: A modellezés során végrehajtott absztrakciók illetve azok ellentétpárjai (a denormalizálás, generalizáció és dezaggregálás) az extenziók és az intenziók változásait okozzák. A tudatos tervezőnek tisztában kell lennie az absztrakciók kihatásaival, illetve már azzal is, hogy mikor melyiket célszerű alkalmaznia. Ha a fenti egyedet megbontjuk a Vevőkód  $\rightarrow$  Vevőnév függés mentén, azaz normalizáljuk, attól az egyed extenziója - azaz a „rendelés” fogalom jelentése - semmit sem változik. Viszont módosul az intenziója, mert a Vevőnév kikerül a tulajdonságsorból.

**13.3 példa** SZEMÉLY-E (**Személy azonosító**, Státus, ...K, T, D, E)  
SZEMÉLY (**Személy azonosító**, Státus, ...K)  
TANÁR (**T-személy azonosító**, T)

Magyarázat: Itt a *specializáció* tárgyalásánál már bemutatott példát vettük elő. Ha az eredeti SZEMÉLY-E egyedet specializáljuk, attól a végső SZEMÉLY egyed extenziója nem változik meg, viszont intenziója lényegesen módosul. Az inverz művelet esetében már más a helyzet. Ha a 13.1 példa első megoldását átalakítjuk, vagyis a BANK egyedet is a SZERVEZET-be generalizáljuk, akkor az extenzió mindenképpen kibővül. Vigyázat! A generalizálás nem tévesztendő össze az aggregálással.

+ **Azt a műveletet, aminek során több azonos lényegű jelenségnek a tulajdonságait egy közös egyedben egyesítjük, aggregálásnak hívjuk.**

**13.4 példa** SZERVEZET-1 (**Szervezet az**, Név, Cégbejegyzés száma)  
SZERVEZET-2 (**Belső kulcs**, Szervezetnév, Szervezet címe)

Magyarázat: Az informatikában az aggregálás fogalmát többféleképpen értelmezik. Az adatmodellezésben az aggregálás valójában nem más, mint korrekció. Tegyük fel, hogy egy cégnél „történelmi okok” miatt eddig kétféle szervezet „egyedet” alkalmaztak. (Azért tettük idézőjelbe az „egyed” szót, mert ahol nem végeztek tudatos modellezést, ott persze nem lehet fogalmi szintű tényezőkről beszélni.) Az is elképzelhető, hogy most kezdjük a fejlesztést és két tervezőnk egymástól külön-külön dolgozva kétféle jövődöbeli szervezet egyedet képzelt el. A lényeg az, hogy a minimalitás axiomatikus szabályának értelmében egy jelenség nem tükrözhető több egyed típusban.

Megoldás: A két tábla tulajdonságsorait mintegy össze kell tolni, természetesen miután kiküszöböltük a homonimákat és szinonimákat. Ezzel a művelettel általában az egyednek az intenziója változik. Extenzionális változásra csak akkor kerül sor, ha az aggregálást általánosítással párosítjuk. Vagyis ha a két szervezet fogalmi köre eredetileg nem azonos, de az absztrakció után a tágabb szervezetfogalmat vesszük alapul.

Kiegészítés: Nem véletlen, hogy az extenzió és az intenzió tárgyalására éppen most, az idő modellezése kapcsán kerül sor. Az idő új szempont bevezetését jelenti a modellbe. A lényegi kérdés pedig az, hogy melyik tényezővel valósítsuk meg ezt az új aspektust? A fentiekből már sejthető, hogy két alapvető megoldás van. Az idő kifejezhető extenzionális módon, vagyis egyedelőfordulásokkal (→ *Állapotmodellezés*). Mód van arra is, hogy az időfaktort intenzionálisan, tulajdonságtípusokkal tükrözzük (→ *Eseménymodellezés*).

A két megoldás közti választás elsősorban *elméleti* alapokon történik, ami azt jelenti, hogy a minőségre kell koncentrálnunk. Azonban a választásnak *gyakorlati* mennyiségi következményei is vannak. Az extenzionális megoldással a tábla mélysége (terjedelme - range), az intenzionálissal annak szélessége (foka - degree) nő. Ez pedig nem közömbös.

### 13.3 Az idő mint modelltényező

Háttér: Az idő modellezése kétféle problémakörbe tartozik. Az egyik kérdés az, hogy az idők során változó *egyéb jelenségeket* miképpen kell megragadni. Erre majd a következő pontokban kapunk választ. A másik felvetés az, hogy maga az *idő mint jelenség* milyen modelltényezőkben testesül meg. Ez viszonylag egyszerűen lezárható kérdés. Ugyanakkor több gondot okoz magának az időnek az értelmezése.

Magyarázat: Az adatmodell egyed-, tulajdonság- és kapcsolattípusok valamint az ezekre vonatkozó korlátok szervezett együttese. A *metastruktúra* írja le a modell tényezőit. Annak megfelelően az idő modellezésében is ötféle részmegoldásra kell gondolnunk.

Az idő tükrözhető *egyed típusként*. Természetesen az IDŐ vagy a DÁTUM *szinguláris egyed* nem mindig kerül megvalósításra konkrét, az időpontokat egyedelőfordulásokként tartalmazó táblaként. Van előnye annak, ha mégis úgy alkalmazzák. Az IDŐ (**Idő**) egyed minden olyan másikkal a fölérendeltje, amiben megtalálható az Idő doméjn minősítő vagy korlátozó szerepneve.

Például az IDŐ a SZERZŐDÉS kétszeres fölérendeltje, ha az utóbbi egyedben található a Kötés dátum és a Lejárat dátum szerepnév-páros.

Amint látjuk az idő *kapcsolattípusként* is funkcionál. Ilyen megoldásra főleg akkor van szükség, ha valamilyen korlátos időfogalommal kell dolgozni, amilyen pl. a munkanap. Persze a munkanap megfogalmazható lenne egyszerű *értéktartományként* is. Azonban a doméjn csak tulajdonságvalidálásra szolgál. Tehát például arra, hogy az X egyedben lévő Munkanap tartalmát ellenőrizzük. Azonban a munkanapok pl. nem azonos hosszúak. Ha kíváncsiak vagyunk a hosszra, akkor ismeretekkel akarjuk leírni a munkanapot, tehát azt egyedként kell tükrözni. Ebben az esetben a MUNKANAP az X egyed fölérendeltje lesz és Munkanap kulcsa már a hivatkozási integritás eszközévé is válik.

Természetesen az idő számos egyedben jelenik meg relatív *tulajdonságtípusként*, azaz attribútumként. Ha egyedbeli szerepe leíró, akkor nincs különösebb mondanivaló róla. Ha viszont azonosítórész, akkor sajátos funkciót lát el. Egyelőre ne foglalkozzunk az idővel mint abszolút tulajdonsággal. Viszont azt feltételezve kell elmondanunk, hogy az idő *tulajdonságstruktúrák* tényezője is lehet.

A *származtatási* struktúráról nincs sokat meditatálni: az idő maga is lehet *származtatások* tényezője. Például ki akarjuk számítani az átlagos rendelésátfutási időt, ami a szállítási és a rendelési dátumok különbségeiből adódik.

A következő alponthan térünk ki a konkatenációra, mint *csoport* struktúrára. Az pedig evidens, hogy az idő maga is csoportot alkot(hat). Majd a későbbiekben rá fogunk mutatni arra is, hogy az idő a modellben sokszor sajátos szerkezetként tükrözendő.

Minden munkanap dátum, de nem minden dátum munkanap. Vagyis a munkanap az általánosabb dátum szűkítő *szerepneve*. Egy modellben számos hasonlóra lehet szükség. Gondoljunk csak az évszakhoz kötött termékkampányokra, az időszakos kedvezményekre, a (kezdő- és záró) határidőkre stb.

Probléma: Természetét tekintve az idő végtelen, folyamatos (kontinuum) és végtelen kis részekre osztható. Ezért ha nem kötetlenül, pontos megjelölések nélkül (tartós, rövid stb.) beszélünk róla, hanem tényszerűen is meg akarjuk ragadni, akkor ki kell választanunk a céljainknak megfelelő *időegységet* (évezred, hónap, másodperc stb.). Ez már csak azért is elkerülhetetlen, mert az adatmodellben az időt elsősorban attribútumként kell megfogni (az egyed és a kapcsolat is azon alapul). A relatív tulajdonság kijelöléséhez pedig az is szükséges, hogy behatároljuk az abszolút, vagyis az időt doméjnként fogalmazzuk meg. Márpedig éppen az idő megengedett értékészletének a behatárolása okozhat fejtöréseket.

A végtelenség és a folyamatosság modellezésére nincs lehetőség. Ezért először is azt kell eldönteni, hogy számunkra melyek az *idő diszkrét mértékegységei*. Léteznek olyan abszolút támpontjaink (pl. éjfél), amikhez képest relatívan (délelőtt 9 óra) jelöljük meg a mértékegységek segítségével az *időpontokat*. A két időpont közötti időmennyiség az *időköz* (időintervallum), ami lehet nyílt és zárt. (Ha felöleli a kezdő/záróidőpontot is akkor zárt, egyébként nyílt.) Mindez annyira köznapi, hogy az olvasó most még talán nem is érti, hogy miért kell beszélni róla. Nos, vegyük sorra a részproblémákat.

**Forma.** A dátumoknak, időpontoknak számos ábrázolási formája létezik. Jóllehet itt látszólag fizikai szintű - tehát nem modellezési - mozzanatról van szó, a részlet mellett mégsem mehetünk el szótlannul. A nem egységes forma ugyanis értelmezési zavarokra vezet. (Egy bank nem akarta elfogadni a szerző csekkjét, mondván, hogy lejárt. Mármint ha a bank nem ismeri az amerikai dátumformát és azt angolnak feltételezi, akkor vajon mire jut vele egy átlagpolgár?)



**Alapok.** Az abszolút támpontok nem azonosak. A Gergely-naptár szerint most már kétezret írunk, más naptárak szerint többet vagy kevesebbet. A nap beosztása sem azonos az összes népnel. Ezért még az is előfordulhat, hogy egy modellben feltétlenül be kell vezetni egy olyan tulajdonságstruktúrát, ami az időpontok konverziójára szolgál.

**Pontosság.** A mai adatkezelők többsége nem tud mit kezdeni azzal a ténnyel, hogy az idő - csoport. Még hozzá olyan, amelynek nemcsak az egésze, hanem a részei is lehetnek opcionálisak (ismeretlenek, inszignifikánsak, értelmezhetetlenek). Az egyik író születési dátuma napra ismert. A másiknál csak az év ismeretes. Vegyünk fel kétféle dátum adatot? Bontsuk szét a csoportot részeire? Csacsiság. Meg kellene engedni, hogy a dátumban a kitöltött év mellett a hónap és a nap vagy csak a nap „üres” is lehessen.

**Kettős arc.** 1999. december 18-a a felületes szemlélő számára egy dolog, egy nap, tehát egy diszkrét egység (napban mért időpont). Az is, de valójában a hajnali 0 órától az éjféle 12 óráig számított 24 órás időtartam (időintervallum). Ez a kettősség pedig nagyon fontos a modellezésben, amit egy példa mutat.

**13.5 példa**    TERV-1 (*Év+Terv azonosító*, ...)  
                  TERV-2 (*Terv azonosító+Hónap*, ...)  
                  TERV-3 (*Terv azonosító*, ..., Terv időszak)

Magyarázat: Mindegyik megoldás többé-kevésbé rossz! Bár nem írjuk le szabályként, az olvasó jegyezze meg, hogy ha az idő az összetett azonosító része, akkor az sohasem lehet az első tag (TERV-1). Bár igaz, hogy fogalmi szinten a kulcsrészek sorrendje közömbös, a megoldás egy szemléleti hibára utal és ezért kerülendő. Ha a tervek nem húzódnának át az évek között, akkor az Év nem lehet a kulcs része. Ha áthúzódnak, akkor pedig nyilván maga a terv az alapvető modellezendő jelenség. (NB.: Általános részsabály, hogy ha a kulcs valamelyik tagja nem azonosító másik egyedben, akkor azt helyezzük az összetétel végére, mert az a kevésbé fontos elem.)

A második megoldás formailag jó. Tartalmilag egy esetben rossz. A szerző találkozott olyan adatbázisokkal, amikben volt ÉVES TERV, HAVI TERV stb. egyed úgy, hogy azok leíró tulajdonságai megegyeztek egymással. Nyilvánvaló, hogy a szerző ilyenkor az általánosítás lehetőségével élve egyetlen Terv azonosító+Időszak kulcsú egyedet tervezne. Ám ebben nem úgy értelmezné az időszakot, mint a TERV-3 egyedben tették.

A harmadik példarész impliciten azt feltételezi, hogy a tervek nem húzódnak át egy hallgatólagosan feltételezett (éves?) határon, ezért az időszak nem része a kulcsnak. Ez a megoldás nemcsak azért hibás, mert valamit elrejt, hanem azért is, mert következtelen. Ugyanis két eset lehetséges. Ha az időszak mindig azonos egység (pl. hónap), akkor még egy további lényegyet titkol. Ha pedig nem az, akkor a szokásos baj lép fel.

Probléma: A tulajdonságok homogén tartalmúak kell, hogy legyenek. Ha az időszak lehet év is, nap is, dekád is, a nyári szezon is (a konkrét esetben ilyen volt), akkor az elv nem érvényesül. Ebből pedig az a gond fakad, hogy az *időre épített mutatók* nem lesznek egymáshoz hasonlíthatók, egymással összedolgozhatók. (NB.: Az a probléma nemcsak az idővel, hanem más „egységekkel” - pl. mértékegység - kapcsolatosan is felmerül. A szerző látott olyan „adatbázist”, amiben a mennyiség hol literben, hol kannában volt megadva. És még csodálkoztak, hogy nem tudják kiszámítani az eladott üdítő összmennyiségét!)

Megoldás: Ha időszakok adatait kell összevetni, akkor két megoldást lehet alkalmazni:

**13.6 példa** IDŐEGYSÉG (**Időegység**)  
IDŐSTRUKTÚRA (**Időegység-1+Időegység-2**, Időmennyiség)  
TERV-1 (... , Időegység)  
  
TERV-2 (... , Kezdőidőpont+Záróidőpont)

Magyarázat: Az első tervező úgy gondolkozik, hogy a teljesen vegyes időegység legyen tárolva magában a tervben. Ha szükséges, akkor az *időcsaládfa* segítségével konvertálja át az időket azonos diszkrét egységekre. A megoldás hátránya, hogy az időkonverziót az időre vonatkozó lekérdezéseknél is végre kell hajtani. A másik tervező megállapította, hogy a legkisebb szóbanforgó egység a nap. Ő a bevitelnél alkalmaz egyetlen konverziót, ha dekádról, hónapról stb. van szó. Vagyis a tárolt terhelte a feldolgozási idő kedvéért. Bár itt a döntés hatékonysági, a szerző a még további egyedek felé mutató kapcsolatok miatt a második megoldást részesítené előnyben. (NB.: A két terv táblában nem jelöltük a szerepet, mert az a helyzet függvényében kulcsrész vagy leíró.)

## 13.4 Állapot és esemény

Háttér: Eddig *statikus* szemléletet követtünk. A statikus adatmodell mindig csak a legaktuálisabb jelenségek tükrözését teszi lehetővé. Ha pl. a partner címe megváltozik, akkor a régi tartalmat az újjal fölülrva az előbbi eltűnik az adatbázisból. A kérdés tehát az, hogy miként lehet olyan modellt építeni, amely *dinamikusan* követi a változásokat?

— **Az esemény jellegű ismeretek sohasem változnak.**

Magyarázat: Az előző alponthan volt szó arról, hogy az idővel kétféle értelemben kell foglalkozni. Úgy is, mint más jelenségekhez kötött tényezővel és úgy is, mint önállóval. A modellben vannak olyan egyedek, amelyekre események vonatkoznak (pl. PARTNER) és vannak olyanok, amelyek éppen az események ismereteit tükrözik (pl. KÁR esemény). A kétféle jellegű egyed másként viselkedik.

A modellezésben nagy szerepe van az egyedek ún. *életciklusának*. Új partner egyed születik (hozzáadás), a régi lakcíme megváltozik (módosítás) és egy idő után a partner megszűnik (törlés). Az esemény jellegű egyedek életciklusa nem teljes. Az egyszer és mindenkorra egy tény, hogy valahol egy káresemény történt. Ez a tény elvileg már nem változhat, tehát nem vonatkozhat rá módosítás. Viszont az esemény jellegű egyedeknek mindig ab ovo tulajdonsága kell, hogy legyen valamilyen idődimenzió.

Fontos: A hibajavítást illetve a nem végleges ismeretek pontosítását - pl. kár összege - elvileg nem tekinthetjük tényszerű módosításoknak. Ezt fontos tudomásul venni éppen az alább elmondandók miatt. Ha elütjük a partner címét és korrigálunk, akkor nem történik valós esemény és nincs szükség az esemény/állapot vezetésére. Ezért éppen elég baj, hogy a programozók ugyanúgy kezelik az érdemi és a technikai változásokat.

Az „esemény jellegű ismeret” a fenti szabályban nemcsak az egyedekre, hanem a többi modell-tényezőre is vonatkozik. A nem-esemény jellegű egyedeknek is vannak változatlan tulajdonságai. Jó adatbázis esetében feltétlenül ilyen az azonosító. Mondhat bárki bármit, a szerző ilyennek feltételezi a személy nemét is. Egy ingatlanak a címe megváltozhat, de a földrajzi koordinátája aligha. Eléggyé sajnálatos, hogy a mai adatkezelőkben nem lehet a tényezőket *instabil* illetve *stabil* módon minősíteni és annak megfelelően megengedni vagy tiltani a módosításukat. Gondoljon csak az olvasó arra, hogy egy szerződéshez több feltétel tartozhat. Új feltételt meg lehet határozni, régit lehet törölni, de a feltételnek mint alárendeltnek a szerződéshez mint fölrendelt-hez való kapcsolatát nem lehet módosítani.

— ***Az időt valós tényezőként és nem technikai adatként modellezzük.***

Magyarázat: A modellezés az on-line adatbázisra vonatkozik. Mivel az *archiválás* nem modellezési, hanem technikai megoldás, azzal itt nem foglalkozunk. A kezelőrendszerek számos idővel kapcsolatos technikai adatok is vezetnek. Ilyen például az *időbélyegző* [time-stamp]. Ezt még a programozó sem kezelheti és ezért nem is tekintendő modellezési tényezőnek. Most pedig térjünk át a valódi dinamikai kérdésekre.

— ***Sohasem modellezzünk instabil időtartamot stabil időpontok helyett.***

**13.7 példa** SZEMÉLY (Személy az, ..., Életkor )  
FELMÉRÉS (Felmérés az, ..., Személy az, Életkor )

Magyarázat: A példa trükkös. Ha személyekről vezetünk nyilvántartást, akkor abban nem az évente változó (instabil) életkort, hanem a változatlan (stabil) születési dátumot fogjuk felvenni. (Íme még egy adat, ami sohasem módosítható, legfeljebb korrigálható.) A SZEMÉLY egyed tehát rossz. Viszont egyes felmérésekben nem kíváncsiak a pontos születési dátumunkra. A felmérés esemény jellegű egyed. Abban az Életkor azt a sohasem változó (stabil) tényrt rögzíti, hogy a felmérés időpontjában hány évesek voltunk. (Ezért a FELMÉRÉS egyed csak azért rossz, mert abban nem illenék utalni konkrét személyekre. Ez viszont erkölcsi és nem modellezési kérdés.)

— ***Ha dinamikusan modellezzük, azt sohasem tesszük egyszeresen.***

**13.8 példa** SZERZŐDÉS (Szerződésszám, ..., Utolsó változás kelte, Változás oka)

Magyarázat: Számos helyen találkozunk hasonló megoldással, ami több szempontból is rossz. Egyrészt az utolsó változás kelte gyakorlatilag semmitmondó ismeret. Megint csak mennyiségi adat, ami minőséget nem hordoz. Délután kettőkor a szerződést totálisan átdolgozzák. Kettő óra egy perckor valaki módosítja az ügyintéző elütött nevét, tehát az utolsó dátum nem a minőségileg lényeges változásra fog vonatkozni. Másrészt a példa egy tipikus következtetlenséget is mutat. Világos, hogy abban az „Utolsó változás oka” nevet kellett volna alkalmazni, mert az összetartozó ismereteket konzekvensen kell minősíteni. Harmadrészt az „utolsó”, „előző”, „maximális” stb jelzők mindig azt sejtetik, hogy az ismeret többszörös. Hacsak egy változás lenne lehetséges, akkor értelmetlen volna a jelző.

Probléma: A példa nem tartalmaz dinamikus elemet, mert valójában nem modellezi a változásokat. Azonban a két leíró adat mögött egy *dinamikus ismétlődő csoport* húzódik meg. Ha az

egyedet ezen implicit ismétlődés mentén normalizáljuk, akkor lehetővé válik a többszörös változás vezetése, vagyis valóban dinamikus modellt kapunk. Mielőtt ezt a megoldást kifejténénk, rá kell mutatnunk, hogy a változás kétféle módon modellezhető.

— ***A változást vagy állapottal, vagy eseménnyel tükrözzük. Azonban együtt sohasem alkalmazzuk a kétféle megoldást ugyanarra a tényezőre.***

Magyarázat: Dinamikus modellezésre azért van szükség, mert a valós jelenségekre vonatkozó ismeretek megváltozhatnak. A változást **esemény** váltja ki (Kovács elköltözik volt lakóhelyéről) és a változás új **állapotot** eredményez (Kovácsnak más lesz a lakcíme). A költözés és a lakcím-változás egyazon éremnek a két oldala. Ha azt is rögzítenénk az adatbázisban, hogy Kovács hová költözött (esemény), meg azt is, hogy a költözés után hol lakik (állapot), akkor adatbázisunk hemzsegne a redundanciáktól.

## 13.5 Állapotmodellezés

Háttér: El kell döntenünk, hogy a dinamikát az állapot vagy az esemény tényezőjével vezettjük-e be modellünkbe. Ehhez a döntéshez nyújtunk némi támpontot az alábbiakban. Didaktikai okokból előbb az állapotmodellezést kell mérlegelnünk.

— ***A totális állapotmodellezés sohasem célszerű megoldás.***

**13.9 példa**    **SZEMÉLY** (**Személy** az, Név, Nem, Születési dátum, ...)   
                    **SZEMÉLY-T** (**Személy** az+**Időszak**, Név, Nem, Születési dátum, ...)

Háttér: Az adatmodell minden tényezője tulajdonságon alapul. Az egyedet annak kulcsa képviseli, a kapcsolat pedig pontosan erre az azonosítóra épül. Az egyed bevitele annyit jelent, hogy új azonosítóérték születik, és analóg módon értelmezhető az egyed törlése is. A kapcsolat létrehozása, módosítása, megszüntetése egyet jelent a **kapcsolótulajdonság** értékadásával, az érték változtatásával illetve ürítésével. Ha egy tulajdonságnak az értéke bármelyik értelemben is módosul, akkor a vonatkozó egyed korábbi állapotát egy újabb váltja fel. (Létrehozáskor a nem-létező állapot változik létezőre). Az **állapot** tehát az az értéksor, amit az egyed tulajdonságsora két változás közötti időszakban felvesz. (NB.: A táblákban sokszor alkalmazott Állapotkód nem pontosan fedi ezt az állapotfogalmat, mert abban - sajnos - keverednek a valós és a technikai eseményekre utaló értékek.)

Magyarázat: Az állapot modellezése akkor **totális**, ha a tulajdonságsornak minden tagja tartalmazza az adott állapotnak megfelelő értéket. A **SZEMÉLY** egyed csak egy állapot - az aktuális - rögzítésére alkalmas. A **SZEMÉLY-T** egyed tulajdonságsora teljesen azonos az előzőével (innen a totális jelző). Természetesen azzal az eltéréssel, hogy a kulcsban utalni kell az állapot érvényességi időszakára. Ennek módjára majd később térünk ki.

Probléma: Az egyedben vannak olyan tulajdonságok (Nem, Születési dátum), amelyek **stabilak**, azaz értékük sohasem változik. Totális állapotmodellezés esetében ezen tételek *gyakor-*

*latilag* komoly fizikai redundanciát okoznak. Hiszen ha ötven változás vonatkozik Kovácsra, akkor ötvenszer tároljuk születési dátumát. A második egyed *elméletileg* is hibás, hiszen abban világosan látható a **részleges függés**: a stabil tulajdonságok csakis a Személy az kulcsrésztől függenek, az Időszaktól nem.

Kiegészítés: Sokan azt a megoldást választják, hogy az alapegyedben (SZEMÉLY) a legfrissebb adatokat tartják és ahhoz kapcsolnak egy alárendeltet (SZEMÉLY-T), ami a korábbi állapotokat őrzi, de továbbra is totálisan. Itt nyilvánvalóan egy hatékonysági - azaz logikai szintű - döntésről van szó. *Gyakorlatilag* ez a verzió pontosan ugyanazokat a problémákat mutatja, mint az előbbi. *Elméletileg* pedig még annál is rosszabb. Azért az, mert totális a rejtett logikai átfedés. Például a Név nem pontosan ugyanazt jelenti a két egyedben és egyikben sem tükrözi a valódi tartalmat. Az első egyedben Aktuális név, a másodikban Régi név vagy hasonló megnevezést kellett volna alkalmazni.

**13.10 példa** SZEMÉLY (Személy az, Név, Lakcím, Nem, Születési dátum, ...)  
SZEMÉLY-R (Személy az+Időszak, Régi név, Régi lakcím, ...)

Magyarázat: A példa *részleges* állapotmodellezést mutat. A két egyed most együtt kell értelmezni. A két egyed között dinamikus 1:N fokú viszony van, amit a fölérendelt kulcsa hordoz kapcsolóadatként. Az alárendeltbe csak a fölérendelt *instabil*, tehát változékonny tulajdonságai kerülnek. A Régi név a Név minősítő, nemkorlátozó szerepneve.

Probléma: A részleges megoldás minden értelemben jobb a totálisnál. Példánk most már nem tartalmaz sem részleges függést, sem helytelen logikai átfedést. Emiatt komoly mértékben lecsökkent a fizikai redundancia. Ámde nem szűnt meg teljesen, hiszen a név és a lakcím nem egyszerre változik. Ezért a cím változásakor a Név és a Régi név tartalma ugyanaz lesz, ami miatt a „rég” minősítés sem tökéletesen kifejező.

Kiegészítés: Az időszak elvileg az érvényességi időintervallumot mutatja. Vegyük észre, hogy ekként egy sajátos *előzményredundanciát* hordoz magában. Ez azt a fizikai átfedést jelenti, ami az időhatárokból következik. A T időszak záró időpontja megegyezik a T+1 időszak nyitó időpontjával. A tervező ezért döntés előtt áll. Az időszakban vezetheti mind a két időpontot (érvényesség kezdete és vége), vagy dönthet az egyik mellett. Az előbbi esetben apró fizikai átfedéssel kell számolnia. Ez megengedhető. Például mérlegeknél a nyitóegyenleg azonos az előző időszak záróegyenlegével, de azért mind a két adatot felvesszük a táblába az érthetőség és homogenitás kedvéért. Ha csak a kezdetet vagy csak a véget választjuk, akkor viszont ismeret veszítünk, mert az előbbi esetben nem vezethető a törlés kelte, az utóbbiban pedig a bevitel dátuma!

—

**A logikai kapcsolhatóság nem jelent egyben fizikai konnektivitást is.**

**13.11 példa** SZEMÉLY-R (Személy az+Időszak, ...)  
BÁRMI (Bármilyen az, ..., Személy az(+Időszak))

Probléma: A szabályban rögzített tétel nemcsak az idő modellezésében érvényes, hanem általában is. Azért kerítettünk sort most a kifejtésére, mert itt igen különleges helyzetek állhatnak elő. Az érdemi jelenség kulcsa (Személy az) vagy szükségszerűen mindig (totális állapot), vagy olykor (részleges állapot) kiegészül az időfaktorra utaló adattal (Időszak). Mindez pedig az alárendelteknek okoz nem kis fejtöréseket.

Ha a BÁRMI az *aktuális* személy alárendeltje, akkor kapcsolónak elegendő lenne a Személy az tulajdonság is. Ha nem, akkor az Időszak is szükséges a hivatkozáshoz. Ez a kettősség az egyik gond. A másik probléma az, hogy a BÁRMI-nek lehetnek olyan egyéb fölrendeltjei is, amik felé szintén az időszakkal teremtünk kapcsolatot, de az időszak értelmezése nem pontosan ugyanaz, mint a személynél. Ezért elvileg, a modell síkján a SZEMÉLY-R és a BÁRMI egymáshoz kapcsolódik, ami viszont nem jelenti azt, hogy gyakorlatilag, az adatbázis síkján is ezt teszi. A kapcsolótulajdonság vagy alakilag nem egyezik meg a személy kulcsával, vagy egyéb okokból nem veszi fel annak értékeit. Ilyen esetekben a procedurális ártértelezés elkerülhetetlen.

## 13.6 Eseménymodellezés

— **Az eseménymodellezés során kerülni kell a tartalmi átfedéseket.**

**13.12 példa** SZEMÉLY (Személy az, ..., Lakcím)  
LAKCÍMVÁLTOZÁS (Személy az+Dátum, Régi lakcím, Új lakcím)

Magyarázat: A második egyedben a két leíró adat a Lakcím nemkorlátozó szerepneve. A tervező úgy gondolta, hogy ő a teljes valóságot akarja modellezni. Vagyis azt, hogy egy személynek egy adott napon erről-arra változott a lakcíme. Ez egy valós tény, amit egy egyedelőfordulás kell, hogy tükrözzön.

Probléma: Mivel a Régi lakcím és a Lakcím tartalma azonos, a két egyed fizikailag - és rejtetten logikailag is - redundáns. Ezt a Személy az → Régi lakcím részleges függés jelzi a számunkra, mert a meghatározott *nem függ a változás dátumától*. Ezért azt ki kell emelni a második egyedből. (NB.: A régi köteget feldolgozásoknál szokták megadni a volt értéket is technikai ellenőrzés céljából. Erre ma már nincs semmi szükség.)

— **Az események egyenkénti modellezése sérti az azonosítás szabályait.**

**13.13 példa** SZEMÉLY (Személy az, ..., Név, Lakcím)  
LAKCÍMVÁLTOZÁS (Személy az+Dátum, Új lakcím)  
NÉVVÁLTOZÁS (Személy az+Dátum, Új név)

Magyarázat: Az állapotmodellezés extenzionális dinamikus tükrözést jelent. Vagyis az idődimenzió bevezetésével az eredeti egyed típus előfordulásainak a körét bővítjük. Ezzel szemben az eseménymodellezés intenzionális megoldás. Az eredeti egyed típusnak azon tulajdonságait, amelyekre változások vonatkozhatnak (instabilak) mintegy megkettőzzük. A fenti két változás egyed az alapegyed dinamikus kiegészítésére szolgál.

Probléma: Technikai-kezelési gondot jelent, ha egy egyednek hús instabil adata van. Ekkor húsféle változásegyedre van szükség, ami a kezelést elbonyolítja. Szintén kezelési gond, hogy a személy legfrissebb adatainak a megismeréséhez ki kell keresni mindegyik változásegyedből az utolsó tételt. Ezen a bajon könnyű segíteni: nem a *mai* eseményeket, hanem a *régieket* modellez-

zük. Az igazi probléma elvi: Az azonosítás szabályai szerint két egyednek nem lehet azonos a kulcsa. Ez a gond nem hidalható át átnevezésekkel:

**13.14 példa**    SZEMÉLY (*Személy az, ..., Név, Lakcím*)  
                  LAKCÍMVÁLTOZÁS (*Személy az+Lakcím dátum, Régi lakcím*)  
                  NÉVVÁLTOZÁS (*Személy az+Név dátum, Régi név*)

Magyarázat: Az aktualitás problémáját kiküszöböltük: mindig a fölérendelt tartalmazza a legfrissebb ismereteket, a régieket a változások őrzik. A felszínen a kulcsok egyezésének a gondját és elsimítottuk a szerepnevek alkalmazása által. Azonban ez csak álmegoldás. Mivel nem korlátozó szerepnevekről van szó, a dolgot mindig visszafordíthatjuk. Amit azért is meg kellene tennünk, mert ha több változás (pl. lakcím, foglalkozás) ugyanazon a napon következik be egy valós esemény (költözés) kapcsán, azok nem kapcsolhatók össze.

**13.15 példa**    SZEMÉLYVÁLTOZÁS (*Személy az+Dátum, Régi lakcím, Régi név*)

Magyarázat: Ha két egyed kulcsa megegyezik, akkor őket azonosaknak kell tekinteni. Ez pedig azt jelenti, hogy leíró tulajdonságaikból egy tulajdonságsort kell alkotni. Tehát a 13.14 példa modellrészletét a jelenlegire kellene átalakítani.

Probléma: A figyelmes olvasó nyilván észreveszi, hogy ha betartjuk a modellezés összes szabályát, akkor az efféle eseménymodellezéssel visszatérünk az állapotmodellezésre. Ez olyan huszonkettes csapdája, amelyből eddigi módszereinkkel nem juthatunk ki. Ezért az események modellezésére teljesen új filozófiát kell keresnünk.

## 13.7 Inverzió

+

**Inverzióról akkor beszélünk, ha a leíró tulajdonságokhoz rendeljük hozzá az azonosítót és nem megfordítva (mint az szokásos).**

**13.16 példa**    KOCSI (**Rendszám**, Kocsitípus)  
                  INVERTÁLT SZÍN (Szín, Rendszám)  
                  INVERTÁLT SÚLY (Súly, Rendszám)

Magyarázat: Normális esetben az egyedek úgy épülnek fel, hogy kiválasztjuk a kulcsot (Rendszám) és a tulajdonságsorba illesztjük a tőle függő leíró tulajdonságokat (Szín). Az ismeretek tükrözésének létezik egy másféle filozófiája is. Nem az egyedhez - illetve az azt képviselő kulcs-hoz - rendeljük a tulajdonságokat, hanem megfordítva, a tulajdonsághoz az egyedeket (reprezentáló kulcsokat). Az utóbbi esetben egy ismérvhez (Szín) tetszőleges számú kulcs (Rendszám) tartozhat, ezért az ilyen ismeretegyüttesnek nincs hagyományos értelemben vett azonosítója.

Kiegészítés: A normális és az invertált [inverted] tükrözést együtt is alkalmazhatjuk, de nem ugyanarra a tulajdonságra, mert az redundanciát okozna. Ha egy egyed valamennyi tulajdonságát invertáltan tükrözzük, akkor *teljes inverzióról* beszélünk. Ekkor az eredeti egyed eltűnik, fel-

oldódik az inverz táblákban. Példánk *részleges inverziót* mutat, mert a Kocsitípusra nem készült inverz tábla.

Probléma: Az inverz megoldás előnyeiről majd a későbbiekben győzzük meg az olvasót. A fenti módon végrehajtott invertálás hátránya, hogy sok táblából kell összeszedni az egy adott jelenségre vonatkozó ismereteket. Ezért azt így csak ritkán szokták alkalmazni.

+

**Tulajdonság-általánosításról beszélünk akkor, ha különböző jelentésű ismereteket egy tulajdonságba összevontan modellezzük.**

**13.17 példa** KOCSI (**Rendszám**, Kocsitípus)  
INVERTÁLT KOCSI (Kocsijellemző, Jellemző érték, Rendszám)

Magyarázat: Ha a tervező ad-hoc, implicit módon egy tulajdonságban eltérő tartalmakat fog össze, akkor az nem tulajdonság-generalizáció, hanem rossz logikai szintű tervezés. A valódi általánosítás mindig a példa módjára történik. Egyetlen táblába vonjuk össze a volt invertált egyedeket. A táblában egy külön tétel (Kocsijellemző) utal az eredeti (Szín, Súly) tulajdonságokra, amelyek tartalmait a generalizált közös tétel (Jellemző érték) hordozza. Az INVERTÁLT KOCSI-ban „Szín - piros”, „Súly - 1020” párosok fognak megjelenni.

Kiegészítés: Mivel a tulajdonságnevek tárolása több itt nem részletezendő ok miatt nem kívánatos, általában fel szoktak venni egy KOCSIJELLEMZŐ (**Kocsijellemző kód**, -név) egyedet. Magában az invertált táblában a kódot alkalmazzák.

**13.18 példa** INVERTÁLT KOCSI (**Rendszám**+**Kocsijellemző kód**, Jellemző érték)  
KOCSIJELLEMZŐ (**Kocsijellemző kód**, Kocsijellemző név)

Magyarázat: A 13.17 példa megoldása elvileg tökéletes. Azonban gyakorlatilag a mai kezelőrendszerek - néhány régítől eltérően - nem támogatják annak kezelését. Feltétlenül igénylik a kulcs megjelölését, ezért az eredeti invertált megoldást át kell alakítani. Ha a kocsinak egy színe és egy súlya van, amint azt feltételezzük, akkor azt az egyed kulcsának és a jellemző kódjának az együttese funkcionálisan meghatározza. Ennek alapján építhető fel az INVERTÁLT KOCSI (immár valójában nem is igazából inverz) táblája.

Ez a tábla nem tekinthető külön egyednek, mert az egyed továbbra is maga a kocsi. Itt a KOCSI egyed intenzionális kiegészítéséről van szó. A KOCSI tulajdonságsorát az inverz táblában lévő kocsijellemzők mintegy meghosszabbítják.

**13.19 példa** KOCSIVÁLTOZÁS (**Rendszám**+**Kocsijellemző kód**+**Dátum**, Jellemző érték)

Magyarázat: Mint már tudjuk, ha bevezetjük az idődimenziót egy tábla kulcsába, akkor olyan táblát kapunk, ami az eredeti tábla 1:N fokú alárendeltje. Nem kell kifejteni, hogy a hármas összetételű kulcs háromféle keresést támogat azok kombinációival együtt. Lehet keresni az egyed, a tulajdonság és az idődimenzió mentén. (NB.: Nem kell aggódni a KOCSI adatainak az összekeresése miatt. Méréseink szerint a művelet nem időigényes.)

A fenti megoldás hallatlan előnye, hogy szerkezetfüggetlen. Ha tisztán állapotot vagy esemény modellezzük, akkor a programok a szerkezettől függenek. Ha felvesszünk egy új tulajdonságot a KOCSI egyedbe, akkor azt meg kell tennünk a KOCSIÁLLAPOT-ban is illetve új X-VÁLTO-



ZÁS táblát kell létrehozni. Az invertált megoldás esetén csak annyi a teendő, hogy felveszünk egy új KOCSIJELLEMZŐ előfordulást.

**13.20 példa** ESEMÉNY (*Táblajel+Táblakulcs+Jellemző kód+Dátum*, Régi érték)  
JELLEMZŐ (*Jellemző kód*, Jellemző név)

Magyarázat: Az előbbi példa megoldása általánosítható ún. *történeti* [history] táblába. Ekkor nem magát a konkrét kulcsot (Rendszám) jelöljük meg, hanem azt, hogy melyik alaptábla (Táblajel) melyik azonosítójáról (Táblakulcs) van szó. Például Táblajel=KOCSI és Táblakulcs=Rendszám. Ha sok változékony egyedünk van, akkor egyedi invertáláskor (13.19 példa) sok változástáblára van szükség. Ilyenkor megfontolható a generalizálás.

## 13.8 Idősor

**13.21 példa** TERMELEÉS (*Létesítmény az+Időszak*, Termelt mennyiség)  
TERV (*Terv az*, Január, Február, ...)

Magyarázat: Léteznek olyan egyedek, amik több időszakra vonatkozó terveket illetve tényeket rögzítő tulajdonságokat tartalmaznak. Természetüket tekintve ezek igen sokfélék lehetnek. A TERMELEÉS egyed *expliciten* utal az időszakra és nem korlátozza határral az időszakok számát. A TERV egyed csak *impliciten* tartalmaz időszakot (az ember tudja, hogy a Január egy hónap, de ez a modellben nem került rögzítésre) és az időszakok száma tizenkettőre korlátozott. Ráadásul az egyed statikus, mert nem utal az évre. A tervezőnek kell eldöntenie, hogy a kétféle megoldás közül melyiket fogja választani a körülmények függvényében. Az első rugalmasabb, a második hatékonyabb.

Kiegészítés: Az idősorok a legtöbb esetben nem igazi egyedek, hanem inkább egyedként megfogalmazott kimeneti táblák. Azért azok, mert sokszor származtatott adatok alkotják őket. A modellezésben nem játszanak meghatározó szerepet. Egy adatmodell mindig igen könnyen - a meglévő struktúra változtatása nélkül - kiegészíthető olyan egyedekkel, amik a struktúra legalsó szintjén foglalnak helyet. Az idősorok pedig értelemszerűen ilyenek.

**13.22 példa** CSALÁDI ÁLLAPOT (*Személy az+Időszak*, Családi állapot)

Magyarázat: Az idő olykor speciális korlátként is működik. Az itteni egyed látszólag ugyanolyan jellegű, mint a fenti TERMELEÉS. Azonban erre az egyedre egy igen speciális úgynevezett *átmenetkorlát* [traversing] vonatkozik. Senki sem lehet özvegy, ha előtte nem volt házas. A férfi házas állapotból már sohasem válhat agglegénnyé. Mindez azt jelenti, hogy az időszak-állapot párosok egy sajátos idősor-korlátot alkotnak.

## 14. KRITIKUS TERVEZÉSI TÉNYEZŐK

### 14.1 Hibák - más szemszögből

A járatlan informatikus nem is gondolná, hogy egy adatbázis tervének az elkészítésekor milyen rengeteg és rengetegféle hibát lehet elkövetni. Ráadásul a felszínen csak aprónak tűnő probléma a mélyben igen komoly problémák forrása lehet. Ha elírtunk vagy rosszul minősítettünk egy nevet, üsse kő, úgyis tudjuk, hogy miről van szó - gondoljuk. Majd kiderül, hogy a modell - az elírás miatt - kapcsolathányos, nem egyértelmű, nem érthető.

*Ebben a fejezetben a tipikus tervezési hibákat fogjuk megtárgyalni. A hibák három jellegzetes kategóriába esnek. Mindegyikükkel külön pont foglalkozik.*

A fogalmi adatmodell nevének megfelelően fogalmakat rendszerez. Sajnálatos módon a mai tervezési segédletek többsége egyáltalán nem támogatja a fogalmi szintű modellezést, azokban csak „fogalminak” titulálják a valójában logikai szintű terveket. Használatukkor a tervező arra kap ösztönzést, hogy gyorsan összeállítson egy **formálisan** megfelelőnek tűnő elképzelést. A tervezési formulák kitöltési kényszere oda vezet, hogy a fejlesztő nem igazán gondolkodik el a tényezők **tartalmi** lényegén. Emiatt azok fogalmilag nem eléggé tisztázottak, a nevek nem (jól) tükrözik a jelentéseket és a tervező nem támogatja meg az értelmezést jól megfogalmazott leírásokkal (14.2 **Fogalomalkotás**).

Fejezetünk harmadik pontja a „**CSAKiS**” címet viseli. A tervezők igen nagy fantáziával képesek rossz azonosítókat (A) és célszerűtlen kódolási megoldásokat (K) választani. A CASE-ek minimális mértékben (sem) támogatják a csoportok (CS) és a szerepnevek (S) tudatos és ésszerű alkalmazását. Mivel pedig ezekre sajátos szerkezeti elemek épülnek (altípusok, visszamutató viszonyok), a tervezőnek nincs igazán lehetősége szemantikailag gazdag és kifejező modell összeállítására akkor sem, ha erre törekszik. Bár az indexek (i) nem fogalmi szintű tényezők, pár rövid megjegyzés erejéig azokra is kitérünk. Ugyanis az a baj, hogy néhány tervező már előre indexekben gondolkodva azokhoz igazítja az érdemi tényezőket, ahelyett, hogy megfordítva járna el.

A modell speciális részsszerkezetei olyan sablonos megoldásokon alapulnak, amilyen például a családfa (14.4 **Sablonok**). Ezeken túlmenően az absztrakciókra (normalizálás, specializáció, aggregálás) vonatkozó alapszabályok maguk is mintákként foghatók fel. Például specializációra akkor kerül sor, ha létezik az a képlet, ami szerint az egyed alternatív feltételes függéseket tartalmaz. A tervezőknek ezen a téren kellene leginkább változtatni a szemléletmódjukon. Ők is használnak sablonos, hagyományos megoldásokat, csak éppen olyanokat, amelyek még az adatbázisok előtti korszakból maradtak rájuk.

### 14.2 Fogalomalkotás

#### 14.2.1 Nevek

A mindennapi életben sűrűn használjuk a „fogalmam sincs” fordulatot. A fogalmakat a modellezésben (is) nevekkkel jelöljük. Ezért nem véletlen, hogy pont a neveknél kezdjük a

tervezési bajok ismertetését. Ugyanis a tervezőnek remélhetőleg az a szándéka, hogy aki a modellt áttekinti, annak legyen fogalma arról, hogy mit is tartalmaz az adatbázis.

**1. probléma:** Az adatmodellt alkotó egyed-, tulajdonság- és kapcsolattípusok nevei nem sérthetik a *valóságghűség* követelményét. A névnek a tényleges tartalomra kell utalnia, különben az adatbázisba helytelen adatok kerülhetnek. Előfordult például, hogy a „piros” értéket adták a Gépkocsi minősége nevű adatnak. Ez talán nem történt volna meg, ha a tartalmat hívebben kifejező Gépkocsi műszaki állapota megnevezést alkalmazták volna.

**2. probléma:** A nevek nem sérthetik az *érthetőség* kritériumát. Gyakori tervezési hiba, hogy a tervezők rövidített neveket és/vagy névkódokat alkalmaznak. Ez persze megtehető a logikai terv szintjén, de a fogalmi modellben nem. Azért nem, mert nem érthetőek és emellett olykor ellentmondanak az egyértelműség szabályának is. Például ha a tervben rövidített nevek is vannak, akkor nem tudható, hogy a magyartalanul írt „Rendszam” név Rendszámot vagy Rendelésszámot takar-e. Még kevésbé célszerűek a névkódok. Például az „A\_szam” éppenúgy vonatkozhat anyagra, mint alkalmazottra. A kódolt név egyrészt nem érthető mások számára (még akkor sem, ha a tábla neve talán ad valami eligazítást), másrészt mindenképpen lehetetlenné teszi a céltudatos elemzést.

**3. probléma:** A nevet magyar adatmodellben *magyarul* kell írni, betartva a helyesírás szabályait. A „Rendszam” nem elfogadható név. A szerző látott olyan adatbázis-tervet is, amiben a Kocsi-suja illetve a Tarolas-heje nevű adat szerepelt... Itt említjük meg azt is, hogy a névnek valóban megnevezésnek kell lennie. A név ne legyen mondat, például ilyen kérdés: „Van-e biztosítása (I/N)?”. A névnek nem illik értékfelsorolást sem tartalmaznia. Beszélő kell, hogy legyen, de mindent úgysem mondhat el a tétel lényegéről. A tartalom teljes kifejtésére nem a név, hanem a leírás szolgál.

**4. probléma:** A neveknek *egyértelműeknek* kell lenniük. Ennek dacára az adatbázisok terveiben hemzsegnek a homonimák és a szinonimák. Ezeket a figyelmetlenség mellett a hamis beállítottság is okozza. Ha két dolog hasonló lényegű, akkor azonos nevet adnak nekik. Például ugyanazt a Rendelésszám nevet használják a vevői és a szállítói rendelések azonosító tulajdonságára (homonima). Mivel úgymond úgyis mindenki tudja, hogy azonos lényegűt takar a Törzsszám és a Személy azonosító (szinonimák), nem fáradoznak azzal, hogy a modellt egyértelművé tegyék. A tervezők nem használnak *névszabványokat* a megnevezések formai írásmódjára. Ebből következik, hogy például az egyik tervező A-B, a másik A\_B módon írja ugyanazt a nevet. Az eltérő írásmódból technikai szinonimák és homonimák származnak. Ezek éppen úgy nehezítik az elemzést, mint a valóság.

**5. probléma:** Az eddigi gondok egyik fő oka lehet a helytelen *minősítés*. A tervezők nem tartják be a következő alapvető névadási szabályt:

— ***Az azonosító szerepű tulajdonságokat nem szabad, a leírókat pedig igen mértéktelenül kell minősíteni.***

Az azonosítókat azért nem minősítjük, mert akkor eltakarjuk a *kapcsolótulajdonságok* által hordozott viszonyokat. Például a vevők számláin a „Vevő rendelésszám” helyett nem szabad használni a „Vevőszámla rendelésszám” túlmínősítést, hanem ott is ragaszkodni kell az eredeti névhez. A leírókat minősítjük, de szerényen. A Mennyiség éppen úgy rossz megnevezés, mint a

Vevő-rendelt-mennyiség-a-cikkből. Vigyázzunk az egyensúlyosságra. Ha alkalmazzuk a Vevő rendelésszám és Szállító rendelésszám párost, akkor ne jelenjen meg az előbbi mellett a sima Rendelésdátum megnevezés, ha az utóbbi mellett a korrekt Szállító rendelésdátum minősített név szerepel.

Egyes minősítési helyzetekben kivételesen lemondhatunk a magyar névképzés egyes szabályairól. Nevezetesen arról, hogy a minősítő jelzőt elvileg a minősített *elé* kell tenni. A Terv összeg és Tény összeg neveket esetleg így is írhatjuk: Összeg terv és Összeg tény. Ez a megoldás egy esetben célszerű. A tulajdonságok listákon jelennek meg. A neveket úgy fogjuk kiadni, hogy az együtt látni akart tulajdonságok egymás mellé kerüljenek. Ha tehát a terv és a tény minősítésű adatokat két külön csoportban akarjuk tudni, akkor az első megoldás a jó. Ha az összegeket, dátumokat, mennyiségeket stb. szeretnénk inkább együtt tartani, akkor - kivételesen - élhetünk a hátulról való minősítés lehetőségével is.

**6. probléma:** Mindent összegezve a legcsúnyább hiba, amit a tervező elkövethet - és ezt igen sűrűn meg is teszi - az, hogy önkényes, a felhasználók teljes körével *nem egyeztetett* neveket használ. Ez azért roppant veszélyes dolog, mert egy adott környezetben a teljesen szokványosnak tűnő név mögött az általunk feltételezettől eltérő tartalom rejtőzhet. Így például a szerzővel is előfordult, hogy a műszaki létesítmény és a műtárgy fogalmakat - egy ideig - pontosan az ellentétes értelemben használta, mint a felhasználó. Az egyeztetés elmaradása pragmatikai hibákat eredményez: a felhasználó nem a kért ismeretet kapja.

#### 14.2.2 Absztrakció

— **Egy adott cégen belül minden fogalmat a teljes szervezet szintjén kell értelmezni.**

**14.1 példa** PARTNER (**Partnerkód**, ...)  
SZERZŐDÉS/PARTNER (*Szerződésszám+Partnerjel+Partnerkód*)  
EGYÉB PARTNER (**Partnerkód**, ...)

Magyarázat: A szerződésekben lévő partnereket úgy osztályozták, hogy ha a partner megtalálható a főkönyvelő partneryilvántartásában, akkor oda kell mutatni, egyébként pedig oda új tételt bevinni nem lehet: azt az egyéb partnerek közé kell tenni. A Partnerjel mutatja ezt a varázslatot. Természetesen a modellben nem lehet külön partner fogalma a főkönyvelőnek és a többi felhasználónak.

**14.2 példa** BERUHÁZÁS (**Beruházás az**, ..., *Beruházás településkód*)  
BERUHÁZÁS TELEPÜLÉS (**Beruházás településkód**, ...)

Magyarázat: Az illető készített egy beruházási úgymond „rendszertervet”. Abban felvett egy település kódtárat, ami azonban csak a vállalat által beruházással ellátott körzetnek a településeit tartalmazta. Eközben természetesen létezik egy általános érvényű kódtár is. Világos, hogy az ilyen elzárkózó és önkényeskedő redundáns megoldás nem megengedett.

**14.3 példa** MODUL (*Tanfolyam az+Modul az*, ...)  
TANTÁRGY (**Tantárgy az**, ..., Modul az)

Magyarázat: A tantárgy a modul része. A két egyed mégsem kapcsolható. Gyakori hiba, hogy a tervezők nem konzekvens *érvényességi körökben* gondolkodnak. Itt az első egyed azt feltételezi, hogy a modulok a tanfolyamtól függő tartalmúak, különben nem is lenne szükség összetett kulcsra. A második egyed arra épült, hogy a modul önállóan is megállja a helyét. Tervezője nem tanfolyami szinten, hanem csak modulszinten gondolkodott. A két modul-fogalom így egymásnak ellentmond.

- ***A modellben kerülni kell az implicit specializációt. Másképpen szólva minden valós jelenséget csak egy egyedben szabad tükrözni.***

**14.4 példa** PARTNER (Partnerkód, ..., Partnernév, Partnercím)  
ISKOLA (Iskolakód, ..., Iskolanév, Iskolacím)

Magyarázat: Cégünknek az iskolákkal speciális szponzori kapcsolata van. Ezért érezte jogosnak a tervező a mutatott megoldást. Az azonban hibás. Azért az, mert az iskola lehet velünk egyéb - normál - partneri kapcsolatban is. Világos, hogy ekkor a konkrét iskola két egyedben fog tükröződni, redundáns - és feltehetőleg egyben inkonzisztens - módon. Ekkor az iskola csak impliciten altípusa a partnernek. Helyette az explicit *specializációt* kell alkalmazni. Ugyanis az explicit egyedaltípus előfordulásai már nem külön egyedek, hanem csak az egyedfőtípus megfelelő előfordulásait kiegészítő tulajdonság-részesorok.

- ***A specializációnak és generalizációnak sohasem a darabszám az alapja.***

Példa: A tervező külön SZEMÉLY és SZERVEZET egyedet hoz létre az általánosított PARTNER nélkül. Döntését arra alapozza, hogy sokkal kisebb a szervezetek száma, mint a személyeké, ezért felesleges a két egyedet összevonni. A tervező elfeledkezik a *feltételes függésről*, ami a minőségekkel kapcsolatos. Mennyiségi alapú döntése feleslegesen kettős kapcsolatokhoz fog vezetni. A személynek is van szerződése, számlája, tevékenységi köre stb., meg a szervezetnek is. Ennek a magatartásnak az ellenkezője is előfordul: a tervező pár üres érték kedvéért nem fogja a PARTNER-t a másik kettőre specializálni. Ez a példa egy másik fontos szabályra is figyelmeztet bennünket:

- ***A modellben igen óvatosan kell bánni a kivételekkel.***

Magyarázat: A kivételek nagy része a jellemzőkkel (metatulajdonságokkal) kapcsolatos. Például a kapcsolaterő általában kötelező, de pár esetben opcionális. Ekkor a viszony nem tehető mesterséges értékek által kötelezővé, azt opcionálisként kell meghatározni. Másik példa: a kapcsolat foka általában 1:N, de néhány esetben 1:1 illetve M:N. Ekkor nincs mit tenni: a legáltalánosabb hálós viszonyt kell alapnak tekinteni.

- ***Ne készítsünk se aggályos, se gondatlan modellrészeket.***

Magyarázat: A *gondatlan* tervező nem törődik a kivételekkel. Nem absztrakt szinten, azaz struktúrával, hanem konkrét szinten, vagyis értékekkel akarja megoldani a gondokat. Például ha egy kapcsolat többnyire kötelező, akkor mondjuk csupa 9 értéket tételez fel a kapcsolótulaj-

donságra, ha egy konkrét kapcsolat mégis opcionális lenne. Ez a megoldás nem valóság - „kamu” - ezért kerülni kell.

Az *aggályos* tervező a „hátha” jegyében készíti a modellrészeket. Hátha megváltozik a kapcsolat foka 1:N-ről M:N-re. Hátha két tulajdonság mai funkcionális függése holnapra függetlenségre változik. Hátha ... Tudomásul kell venni, hogy az élet változik és ezért nem létezik örökké változatlan adatmodell sem. Törekedni kell rugalmasan módosítható adatmodell kialakítására. Azonban ha minden tényező terén a maximális biztonságot akarjuk elérni, akkor a modell méretét a szükséges többszörösére növeljük és a felesleges mesterséges tényezők miatt nagymértékben lerontjuk a majdani kezelési hatékonyságot.

— ***Egy jelenség = egy fogalom = egy modellezési egység = egy név.***

Magyarázat: A tiszta adatmodellezés, vagyis a helyes absztrakció összes egyéb szabálya ebben az egy axiomatikus törvényben fogalmazható meg. Minden gondatlan összevonás és minden aggályos szétदारabolás szükségszerűen torz, rossz modellre vezet.

### 14.2.3 Leírások

— ***A fogalmi szintű leírásokat gazdaságosan kell megadni.***

**14.5 példa** SZEMÉLY (Személy *az*, ..., Anyja neve, Foglalkozáskód)

Definíció: Anyja neve = „Az anyja neve.”

Definíció: Foglalkozáskód = „Az XYZ törvény bla-bla” + hosszú kódlista

Magyarázat: Konkrétan megtörtént esetről van szó. Az első részben a tervező egy blőd leírással élt, teljesen feleslegesen. Talán kényszerítették a definíció kitöltésére. Annak úgy semmi értelme. A leírás a fogalom magyarázatára szolgál: ha nincs mit magyarázni, akkor nem is kell magyarázni. A második részben két hibát is elkövetett. Egyrészt túl bő a megfogalmazás. A leírás nem való törvénycikkelyek citálására: elég csak utalni azokra. Másrészt bár a kód nem fogalmi szintű tényező - hiszen már ábrázolást is involvál - nem hiba, ha már az adatmodellben is utalunk megengedett értékeire. Azonban ha a lista igen terjedelmes, akkor azt ne tegyük a leírásba. A célszerű megoldás az, hogy a leírásban csak utalunk a kódokat tartalmazó külön mellékletre.

— ***A leírások mindig az abszolút tényezőkhez kötendők és így egyszeresek.***

**14.6 példa** SZEMÉLY (Személy *az*, ..., Foglalkozáskód)

Definíció: Foglalkozáskód = „Az XYZ törvény bla-bla” + hosszú kódlista

MUNKAKÖR (Munkakör *az*, ..., Foglalkozáskód)

Definíció: Foglalkozáskód = „Az XYZ törvény bla-bla” + hosszú kódlista

Magyarázat: Sajnos igen sok CASE-eszköz számára ismeretlen az általános tulajdonság fogalom (← *Értéktartomány*) és/vagy a tervezők nem ahhoz kötöttek, hanem az egyedhez kapcsolt relatív tulajdonságoknál (attribútum) adják meg a leírásokat. Jobb esetben a szöveget innen-oda átmásolják (közben mindig hibát ejtenek). Rosszabb esetben teljesen ellentmondásos meghatározásokkal élnek. Ezért fontos a fenti szabály, ami szerint a leírást mindig az abszolút tényezőhöz kötöttek kell megadni, akkor pedig egyszeres lesz.

— ***Leírásban adjuk meg mindazt a fogalmi és formális metaismeretet, amit a névvel és a metajellemzőkkel nem tudunk kielégítő módon tükrözni.***

Magyarázat: A tényezők leírása sokféle jelleget ölthet. A fenti szabály szerint ha a név nem tudja jól kifejezni a tartalmat, akkor szükség van külön *definícióra*. Meg kell adni az előfordulások korlátait. Ez a tulajdonságoknál a megengedett értékek készlete (ideértve a kódokat is), ami a *validálások* alapja. Azonban az egyedek és a kapcsolatok is lehetnek sajátos feltételekhez kötöttek, tehát le kell írni azok *korlátait* is. Ha a metajellemzők azt nem teszik lehetővé, akkor itt kell közölni a tényező *minősítését*, pl. azt, hogy az egyed családfa típusú. Származtatott tulajdonságoknál itt adandó meg az *algoritmus*. Végül ne feledkezzünk el sajátos megoldások esetén *magyarázatot* adni a „miértre” (← *Adatszótár*).

## 14.3 „CSAKiS”

Háttér: A modellezésben különös figyelmet érdemelnek a csoportok (CS), az azonosítók (A), a kódok (K), a szerepnevek (S) és - bár nem fogalmi szintű elemek - az indexek (i). Ezek a dolgok egymással is összefüggenek, méghozzá igen bonyolult módon. Így például vannak csoportos kódolt azonosítók. Célunk az, hogy az alábbiakban csak a viszonylag egyszerű szabályokat adjuk közre.

### 14.3.1 Csoportok

Egy csoport több tagból áll és egy tag eleme lehet több csoportnak is. Tehát a csoportok és a tagok hálót alkotnak. Értelmszerű szabály, hogy a csoportnak legalább két tagja kell, hogy legyen, különben nem csoport, hanem elemi tulajdonság. Vigyázni kell arra, hogy a csoportok ne alkossanak ciklust: ha A-nak tagja B, annak C és annak A, akkor nagy baj van. Számos itt nem részletezendő okból csoportnak nem lehet a tagja is csoport, vagyis a csoportokat mindig elemi tényezők együtteseiként kell megadni. (Több ellenkező irányú kísérlet csődöt mondott a CASE-ek esetében.)

Bonyolult a szerepnevek és a csoportok összefüggése is. A több részsabály közül most csak egyet említünk: egy tulajdonság és annak szerepneve nem szerepelhet együtt azonos csoportban. Ez logikus, hiszen a szerepnév meghatározza az elsődlegességét, tehát minden a csoportot tartalmazó egyed normalizálatlan lenne.

Itt említjük a szerkezeti homonimákat/szinonimákat. Gyakran előfordul, hogy az egyik tervező meghatározza az  $A\{B+C\}$  csoportot, miközben a másik a  $D\{B+C\}$  tényezőt adja meg. Ha két csoport tagjai azonosak, de nevük eltérő, akkor kétféle hiba léphet fel. Vagy a csoportnév szinonima, vagy a tagok valamelyike nem azonosan értelmezett homonima.

Nagyon vigyázni kell az ***implicit csoportokkal***. Ilyen például a gépkocsi Alvázszáma, aminek első pár jele - állítólag - a kocsi típusára utal. Ha mindezek után a KOCSI egyed tartalmaz egy explicit Kocsitípus tételt is, akkor máris várható az inkonzisztencia. Az implicit csoportok azért is veszélyesek, mert nem adnak lehetőséget a kapcsolatok explicit meghatározására. Ha például a Számlaszám implicit Vevőkód+Relatív sorszám együttes, akkor fogalmi szinten a számla nem kapcsolható a vevőhöz.

Az implicit csoport egyik fajtája a **vektor**. Olyan adat, ami előre meghatározott számú részből áll úgy, hogy az összetevők értelme azonos. Például a RENDELÉS egyedben van egy Feltételek nevű adat, ami öt különböző feltételkódot tartalmaz egymás mellett a rendelés minemiségének a függvényében. A vektor nem fogalmi szintű konstrukció, mert mesterségesen von össze több jelenséget egy tényezőbe. A vektort tartalmazó egyed még csak nem is normalizált (← Ismétlődő csoport). Rendkívül kitett a változások hatásainak, mert holnap már hat feltételt fognak igényelni a felhasználók. Emellett procedurális síkra terelődik a kezelés, mert a FELTÉTEL (**Feltételkód**, ..., Feltételnév) egyedből kiindulva nem lehet a struktúra alapján kikeresni az adott feltételnek megfelelő rendeléseket.

### 14.3.2 Azonosítók

Az azonosító és részei sohasem lehetnek ismeretlen vagy üres értékűek. Halálos bűn az ismeretlen értéket valamilyen mesterséges tartalommal pótolni, mert az nem fogalmi szintű - nem valóság - megoldás és mindig komoly bajokhoz vezet. További szabály, hogy a kulcs értéke sohasem változhat. Ezért ha a kulcs kódolt felépítésű (amit lehetőleg kerülni kell), akkor a kód csak nem-változó tagokból állhat.

**14.7 példa** RENDELÉSTÉTEL  
 (Rendelésszám+Cikkszám)  
 (Rendelésszám+Tételsorszám, Cikkszám)  
 (Rendeléstétel az, Rendelésszám, Cikkszám)

Magyarázat: A RENDELÉSTÉTEL-t háromféle módon azonosíthatjuk. Az első esetben a kulcs konkatenáció, minden tagja másik egyednek a kulcsa és nincs benne mesterséges többletelem. Alkalmazásához két index kell. A második esetben a kulcs összetett, de egyik tagja mesterséges többletelem, nem mutat másik egyedre. Alkalmazásához két index kell. Az előbbivel szemben előnye, hogy egy rendelésben többször is kérhetik ugyanazt a cikket (mert a Cikkszám csak leíró tulajdonság). A harmadik megoldás egy teljesen mesterséges kulcsra épül. Három indexet igényel és külön validálandó a két külső kulcs párosának az egyedisége (ha az a korlát él).

Codd úr [7] a harmadik megoldás mellett teszi le a voksát. Szerinte az egyed típusokat **kulcs-helyettesítő** [surrogate] kell azonosítani úgy, hogy azok értékei örökérvényűek. Nem vitás, hogy ez a megoldás véd a legjobban mindennemű változás ellen. Ugyanis az általa felhasznált kulcsoknak hívott „természetes” kulcsok változékonyak illetve nem védnek a „fokok” változásai ellen. (Ma a rendelésekben csak egyszer fordulhat elő egy cikk, holnap már többször is.) A kulcspótlék szinte elkerülhetetlen, ha az összetétel igen nagyméretű lenne. Például a Személyi szám szerencsétlen kiírása óta a személyeket a nemhatósági szervezetek csak kulcspótlékkal tudják azonosítani, mert a név, anyja neve stb. összetétel nem alkalmas kulcsnak. Az adatmodell „alján” lévő többszörös főlérendeltű egyedek kulcsai olyan A+B+C+... jellegű konkatenációk, amik szintén meghaladhatják az elviselhető méretet.

Az abszolút sorszámok helyetti **relatív sorszámok** nagyon sok esetben bajok forrásai. Tegyük fel, hogy a szerződéseket egy Területi kód és egy Relatív sorszám azonosítja. Az előbbi adat arra utal, hogy melyik körzetben kötötték a szerződést. Az utóbbi arra, hogy hányadik ügyletről van szó. Ez a csoportos, területre kódolt, sorszámot tartalmazó kulcs kétszeresen is rossz. Ha a szerződést átadják az X körzetből az Y területre, akkor nemcsak a Területi kód változik, hanem a Relatív sorszám is módosul, mert a szerződés az X egységben az ötödik volt, az Y körzetben viszont már a huszonegyedik lesz.



Területileg osztott adatbázisok esetében a tervezők azért nem vállalják az **abszolút sorszám** kulcsot, mert úgymond annak kiadása és egyediségének az ellenőrzése nehézkes. Ha vállalatunk területenként köt szerződéseket a partnereivel, de nincs on-line adatbevitel egyetlen központi számítógépre, akkor valóban nehéznek látszik a kérdés. Mi garantálja, hogy a Kecskeméten 10 óra 50 perckor rögzített szerződés 1234567 sorszáma után a Kőszegen 10 óra 51 perckor megkötött ügylet a 1234568 sorszámot kapja? Semmi, de nem is itt kell keresni a megoldást. Nyugodtan ki lehet adni helyi kulcstartományokat. Az abszolút sorszámnak nem ismérve a zártság, csak az egyedisége követelmény.

**14.8 példa** SZEMÉLY (Belső azonosító, Személyi szám, Útlevekszám, TAJ-szám, Adó azonosító jel, Tartózkodási engedély száma stb.)

Magyarázat: A tervezők nincsenek tisztában a belső és a külső kulcs lényegével. (Most a külső kulcs nem a relációs foreign-key-t jelenti!) Régen a kulcs egyszerre két feladatot látott el. A *külső* elérésnek olyan eszköze volt, ami egyben a *belső* tárolás alapjául is szolgált. Vagyis az azonosító értékéből képezték, annak feleltették meg a tárolási címet. A mai adatkezelők már nem így dolgoznak és ezért a kétféle feladat szétvált. A belső kulcs is csak közvetve kötődik a tároláshoz, inkább az egyedek közötti összefüggések eszköze. Ugyanakkor egy egyeden tetszőleges számú külső kulcs alkalmazható az elérés céljaira. A kétféle feladatot olykor nem lehet, a legtöbbször pedig nem célszerű összekeverni.

Példánk esetében felvehetők ismeretlen személyek adatai is. Azt tudjuk, hogy az illető férfi, kábé negyvenes korú, feketehajú stb. Viszont egyetlen papírját sem ismerjük. Tehát nem is tudjuk azonosítani azok számával/jelével. Íme egy tipikus példa arra, hogy olykor akkor is szükség van az egyetlen belső kulcsra (Belső azonosító), ha amúgy számos külső kulcs áll rendelkezésünkre.

Az azonosítókat tekintve az egyik leggyakoribb hiba az **alternáló kulcsok** következtlen használata. (NB.: Az előző esetben nem volt szó alternáló kulcsról!) Az adatbázis szintjén már megengedhető, hogy az orvos foglalkozású személy sajátos adatait hol az Orvoscód, hol az általánosabb Személy azonosító alapján keressük elő. Azonban az adatmodellben - az elemezhetőség érdekében - ez a váltogatás nem megengedett. Ha egyszer eldöntöttük, hogy a Személy azonosító az elsődleges kulcs, akkor már nem használható az Orvoscód pl. az ORVOS/RENDELÉS és a további egyedekben kapcsolóként.

### 14.3.3 Kódok

Három szabályt kell követni. Az egyikről már volt szó: az azonosító csakis a legvégső vagy pedig a nagyon egyszerű esetekben lehet kód. Például nincs akadálya annak, hogy a településeket, a pénznemeket stb. kóddal azonosítsák, ha betartják a másik két szabályt.

A második regula úgy szól, hogy ne vonjunk össze egy tulajdonságba eltérő tartalmakat, mert az sérti a fogalmi egységesség elvét. Az ilyen összeépítés komoly veszélyekkel jár, mert hierarchikusra szűkíti le a valójában hálós viszonyt. Például itt van ez a szerelvény, ami az „xxx” azonosítójú termék része. Nosza, adjuk neki az „xxxa” kulcsot. Csakhogy a szerelvény az „yyy” azonosítójú termékbe is beépül. Ezért a mérnökök annak az „yyyb” rajzszámot jelölik ki. Mi a következmény? Az „xxx” termék gyártásához Q, az „yyy” termeléséhez W darab teljesen azonos szerelvény tartalékolására van szükség. Az együttes igény ennél nyilvánvalóan kisebb. Valójában a cégnek csak  $V < Q + W$  alkatrészre lenne szüksége. Mivel a szerelvénynek nincs egyetlen - sorszám jellegű - kulcsa, az azonosítója a végtermékekhez kötődik (xxxa és yyyb), a cikk kétféle jelenséggé jelenik meg az adatbázisban - és felesleges készlete igen drága lesz.

A harmadik szabály a kód szerkezetének a tisztaságával kapcsolatos. Már az is bűn, ha egy kódban több, egymással hálós viszonyban álló fogalmat zsúfolunk össze. Ezt a hibát lehet még fokozni azzal, hogy a kódpozíciók tartalmát is vegyítjük. Ha az első jel X, úgy a második jel az anyagra utal, ha viszont az első jel Y, akkor a második jel szint jelent stb. Bár itt a kód fizikai szintű struktúrájáról van szó, látható, hogy a tervező fejében maguk a fogalmak nem tisztán elrendezettek. A vegyes szerkezetű kód a kapcsolatok halála. Ha pl. a Cikk kódon belül hol az egyik, hol a másik jel(csoport) utal a technológiára, akkor a technológia felől tökéletes képtelenség kikeresni a kapcsolódó cikkeket.

Általában a különböző szoftverek adatállományainak az egybeépítésekor illetve a már meglévő állományok korszerűsítésénél - vagyis az integrálás során - merül fel a *többféle azonosítás* gondja. A hagyományos alapokon tervezett állományok azonosítói többnyire kódoltak. Azonban nemcsak emiatt nem vesszük át ezeket belső kulcsokként, pusztán külsőkként használva őket, hanem azért sem, mert értékkészletük a kelletténél szűkebb. Az integrálás ui. többnyire azzal jár, hogy megnövekszik az egyedek előfordulásainak a száma. Például az A és a B integrált részben lévő partnerek halmazai átfedő viszonyban vannak egymással. Következésképpen egyik régi kulcs sem alkalmas az új egyesített halmaz előfordulásainak a lefedésére.

#### 14.3.4 Indexek

Természetesen minket nem e tényezők fizikai aspektusa érdekel. Két dologra óhajtunk rámutatni. Az egyik az, hogy a korrekt fogalmi struktúra sohasem váltható ki fizikai szintű indexszel. Ha például a KÁR egyedben a Tulajdonoskód ismétlődő adat, akkor nem az a megoldás a tulajdonos kárainak a kikeresésére, hogy „indexet húzunk rá”, hanem az, hogy megszüntetjük az ismétlődést. A másik mondanivalónk az, hogy a kulcsok és a csoportok kiválasztása illetve felépítése természetesen kihat az indexekre is. Például ha egy csoportos vegyes kódot letisztázunk - elemi fogalmakra bontunk szét -, akkor több indexre lesz szükségünk. Ezzel egyszerűbb lesz a program, hatékonyabb az elérés és még az sem biztos, hogy megnő a tárigény, hiszen az egy nagyméretűt váltja ki több kis index.

#### 14.3.5 Szerepnevek

Egy elsődleges tulajdonságnak több szerepneve lehet, de ez megfordítva nem igaz, mert az homonimához vezetne. A szerepnevek többszintű hierarchiát alkotnak: a szerepnévnek is lehet további szerepneve. A szerepnévre épülő struktúrákról és azok korlátairól már sok szót ejtettünk, ezért a tudnivalókat itt már nem ismételjük meg. Itt csak arra hívjuk fel a figyelmet, hogy a szerepnév nem tévesztendő össze a minősítéssel. Így például a Vevő rendelésszám minősítés, nem pedig a Rendelésszám szerepneve. Azért nem az, mert a vevői és szállítói rendelések fogalmilag eltérő lényegek és nem egy tartomány elemei.

Mivel a szerepnevek éppen úgy tulajdonságstruktúrák, mint a csoportok, sok gondot okoz e kétféle struktúra meg nem engedett vegyítése. A legtöbb CASE eszköz nem képes a bajok kiszűrésére. Például csak egy talányt vetünk fel anélkül, hogy elárulnánk magát a megoldást. Az olvasó próbáljon példát kreálni arra az esetre, amikor egy csoportnak a szerepneve maga is csoport. (Vajon milyen probléma fakad ebből?)

Természetesen alapvető szabály, hogy a szerepneveket explicit módon meg kell adni. Vagyis a modellben kell, hogy legyen egy olyan lista, ami rögzíti, hogy egy elsődleges tulajdonságnak mik a szerepnevei. Nagyon sok modell azért nem teljes, mert a tervező elfeledkezik a szerepnevek

mint kapcsolótulajdonságok által kijelölt viszonyokról. Ez annak tudható be, hogy nem látja az összefüggéseket. Ha például nem rögzíti le, hogy a Lektor azonosító is egy Személy azonosító, akkor persze nem határozza meg helyesen a lektornak, mint sajátos szerepű személynek a kapcsolatait sem.

## 14.4 Sablonok

**14.9 példa** SZERVEZET-1 (**Szervezet azonosító**, ...) - a kulcs hierarchikus  
SZERVEZET-2 (**Szervezet azonosító**, ..., *Fölérendelt azonosító*)

Magyarázat: Az adatmodellezés hasonlít a mérnöki munkára. (Az angol „engineering” szó tervezést jelent.) A jó és a rossz mérnököket - többek között - a sablonok használata különbözteti meg. A jó tervező értelmesen él a sablon lehetőségével. A rossz nem használ ilyet, vagy éppen ellenkezőleg: mindenütt a maga előképeit erőlteti.

Példánk két sablonos megoldást mutat. Mind a kettő - rossz! Az első azért az, mert az adatmodell nem tartalmazhat implicit - rejtett - tényezőket, így kapcsolatokat sem. Nem jó megoldás, ha a „1123”-as kódú osztály impliciten utal a „112”-es kódú főosztályra. Azért nem az, mert a kulcs értéke sohasem változhat, márpedig az első átszervezésnél ez fog történni. A második verzió jobb, mert explicitté teszi a visszamutató viszonyt. Ámde nem tökéletes, mert statikus modellt feltételez. Nem tükrözhető benne az, hogy korábban melyek voltak a szervezeti kapcsolatok.

**14.10 példa** SZERVEZET (**Szervezet azonosító**, ...)  
STRUKTÚRA (*Szervezet azonosító-F*+*Szervezet azonosító-A*+**Dátum**)

Magyarázat: Ez a megoldás már kifogástalan. Nem rejti el a kettős dinamikus viszonyt.

**14.11 példa** BÁRMI (**Bármí az**, ..., Érvényesség kezdő dátuma, Érvényesség záró dátuma,  
Állapotkód, Utolsó változás oka, Utolsó változás dátuma)

Magyarázat: A szerző sokszor találkozik olyan adatbázisistervvel, amelyben a tervező az érdemi adatok mellett gondolkodás nélkül felveszi a mutatott tulajdonságokat. Ez a sablon rossz alkalmazásának tipikus példája. A lényeg a „gondolkodás nélkül” kitétlen van. Így például a mutatott adatok megjelennek olyan esemény, terv, idősor stb. jellegű táblákban is, amik a valóságban sohasem (!) változhatnak.

**14.12 példa** SZEMÉLY (**Személy azonosító**, Személy név, ...)  
ÜZLETKÖTŐ (**Üzletkötő kód**, Üzletkötő név, ...)

Magyarázat: Tipikus modellezési hiba a tudatos specializáció/generalizáció elhagyása. Ebből a rejtett szinonimák miatt komoly redundanciák származnak. A példa többszörösen is hibás, mert az üzletkötő és a személy között még kapcsolat sem létezik.

**14.13 példa** SZERZŐDÉS (**Szerződés azonosító**, X, Y, Z, Előző szerződés azonosító)  
ELŐZMÉNY SZERZŐDÉS (**Előző szerződés azonosító**, X, Y, Z)

Magyarázat: A két egyed (csaknem) valamennyi leíró tulajdonsága (X, Y, Z) azonos. Ez logikus, hiszen az előzmény szerződés egykor maga volt a szerződés. A megoldás azt a tipikus hibát mutatja, aminél a visszamutató viszonyt egy logikailag teljesen redundáns külön egyeddel váltják ki. Ez elbonyolítja a kezelést és fogalmilag sem ad tiszta képet. (Ha egy szerződés módosul, akkor az előfordulást ki kell emelni az alapegyedből és át kell tenni az előzmények közé.) A célszerű megoldás az, hogy együtt tartjuk és állapotjellel különítjük el az élő és a kiváltott szerződéseket úgy, hogy az előbbi mutat az utóbbira. Tehát elegendő a példa első egyede is, de azt minden szerződésre kell értelmezni.

#### 14.14 példa SZERZŐDÉS (... 1. fél, 1.fél szerepe, 1.fél szerepének dátuma, 2. fél ...)

Magyarázat: Már említettük, hogy tipikus gond a rejtett *ismétlődő csoport* használata. Ha a szerződő felekről valamit el akarunk mondani, akkor az azt jelenti, hogy a szerződés és a partner viszonyát ismeretekkel kell leírni. Ezt pedig mindig viszonyeggyeddel tesszük. Ha ráadásul az 1. fél, 2. fél stb. nem szerepneve a partner kulcsának, akkor a megoldás azért rossz, mert teljesen hiányzik a kapcsolat. Tipikus hiba, hogy a tervező inkonzekvens a partnerek megjelölésében. Az egyikre kulcsával hivatkozik, ami nem akadályozza meg abban, hogy az egyedben szerepeljen például a „Vezető neve” tulajdonság is.

#### 14.15 példa SZERZŐDÉS (... *Fizetési mód kód*, *Gyakoriság kód* ,..., X-kód) FIZETÉSI MÓD (**Fizetési mód kód**, Fizetési mód megnevezés) GYAKORISÁG (**Gyakoriság kód**, Gyakoriság megnevezés)

Magyarázat: A tervezők teletűzdelik a modellt ún. *kétoszlopos táblákkal* [two-column table]. Ezek a kód-megnevezés párosokból álló valamik nem valós egyedek, hanem tárolási/kezelési megfontolásokból alkalmazott logikai/fizikai szintű táblák. Éppen ezért nem is valósítjuk meg őket külön-külön, hanem összevonjuk őket egyetlen háromoszlopos táblába: **KÓDOK (Kódtípus+Kódérték**, Megnevezés). A Kódtípus mutatja, hogy melyik kódolt tényezőhöz (fizetési mód, gyakoriság) tartozik az érték és a megnevezés.

Kiegészítés: Egyes felhasználók szeretnék látni, hogy az egyedben a gyakoriság kódolt ismeret. Ezért ragaszkodnának a Gyakoriság kód felvételéhez. Meg lehet győzni őket arról, hogy éppen elég az adatnak a Gyakoriság nevet adni és a metajellemzők között lehet utalni arra, hogy logikai szinten majd kódot fognak alkalmazni.

Összegzés: Nem lehet célunk az összes elkövethető hiba ismertetése. Ezért inkább a hibák alapjait foglaljuk össze. Az apróságokon kívül a hibák három kategóriába esnek:

- Szinttévesztés. A tervező bevett szokásai, a kezelő korlátai vagy egyszerűen a fogalmi szint ismeretének hiánya miatt eleve logikai szintű megoldásokat alkalmaz.
- Rossz absztrakció. Hiba, ha a normalizálás, a specializáció és az aggregálás illetve ezen absztrakciós műveletek inverzei vagy elmaradnak, vagy túlhajtottak.
- Rossz struktúra. A tervezők egy része nincs tisztában az altípus, a visszamutatás, általában a speciális szerkezeti elemek/sablonok lényegével és azok helyett a nem eléggé kifejező, az összefüggéseket elrejtő hagyományos megoldásokat választja.

## 15. MODELL ÉS RENDSZER

### 15.1 Az integráció hiánya

„Az adatmodell - egy” kitétel azt jelenti, hogy a szervezetben együtt kell látni az összes alkalmazás általános, lényeges és tartós ismereteit. A tapasztalatok megmutatták, hogy a területi, szervezeti vagy funkcionális alapú rendszerintegrálás sohasem vezetett sikerre. Azért nem, mert az alkalmazások adatokon keresztül érintkeznek egymással és ha azok értelmezése nem azonos két rendszerrészben, akkor nem is születhet információ. Tehát az alkalmazások integrálása csakis a közös fogalmakon - vagyis az adatmodellen - alapulhat.

Hiába beszélünk annyit az integrált rendszerekről, valami miatt a szerzőnek eddig még nem volt szerencséje találkozni ilyesmivel. Azt is pontosan tudja, hogy miért nem. Az ok többszörös. Egyrészt a mai tervezők jártasabbak a vizuális eszközök használatában, mint az adatmodellezés elméletében. Másrészt az eszközök mindig csak rendszerrészek - nem pedig totális rendszerek - tervezésében nyújtanak támogatást. Harmadrészt a felhasználók valamilyen módon mindig képesek a tervezőket megalkuvó megoldásokra kényszeríteni.

*Ennek a fejezetnek az a célja, hogy némi támogatást adjon a jelzett korlátos szemlélet megváltoztatására. E cél érdekében meg fogjuk vizsgálni az információs rendszer általános felépítését az adatmodellező szemüvegén keresztül.*

Az első pontban rámutatunk arra, hogy a teljes modell szerkezetében is követni kell a bemenet-feldolgozás-kimenet logikát, ami fölött a vezérlés, alatt a támogatás húzódik meg (15.2 **Rendszerfolyamat**). Ezt az architektúrát mintegy kibontja a második pontunk, amiben a tevékenységekhez és az erőforrásokhoz kötöten fejtjük ki az általánosság és a fontosság szerepét a modellezésben (15.3 **Tájékozódás a modellben**).

Ha valódi modellt készítünk, akkor a terv szükségszerűen nagyméretű és bonyolult lesz. Áttekinthetővé úgy tehetjük, ha az információs rendszert rendszerrészekre bontjuk és az adatmodell prezentálási szerkezetét ehhez a tagoláshoz igazítjuk (15.4 **Modellkeret**). Az említett bonyolultság által okozott nehézségeket akkor oldhatjuk fel, ha tisztában vagyunk az egyed-típusok jellegével, feladataival, kapcsolódásaival (15.5 **Egyedszerepek**). Olykor előfordul, hogy nem tudunk *elméletileg* tiszta megoldásokat kidolgozni, mert kénytelenek vagyunk tekintetbe venni a vásárolt szoftverek illetve a régi alkalmazások modellezési szempontból szuboptimális állományainak a tartalmát is. Ilyenkor *gyakorlatilag* bizonyos kompromisszumokra kényszerülünk (15.6 **Sajátos megoldások**). Törekednünk kell arra, hogy a szükséges kompromisszum ne megalkuvássá váljon.

Végeredményben az egész fejezet arról szól, hogy a mai rendszerek nem integráltak és a jelenlegi fejlesztések sem vezetnek integrált adatbázisokhoz, hacsak nem tartjuk be a következő pontokban kifejtett modellstrukturálási alapelveket.

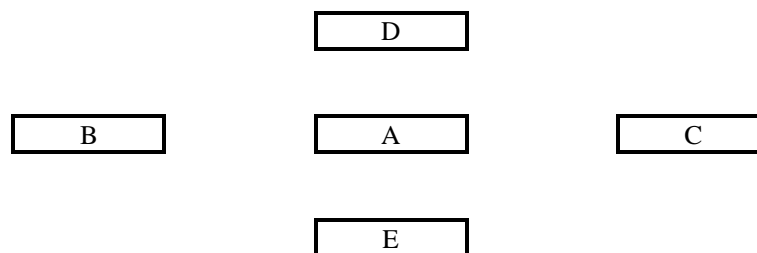
### 15.2 Rendszerfolyamat

Háttér: Az adatfeldolgozásban a programokban megtestesülő tevékenységeket mindig *folyamatként* szokták megragadni. Ez azt jelenti, hogy azokat az input-adatbázis-output irányított

logikának megfelelően fogalmazzuk meg. A külső bemenetből származó adatok az adatbázisba kerülnek illetve onnan az ismereteket külső kimeneten jelenítjük meg. (A külső azt jelenti, hogy az adott ponton emberi közbeavatkozásra kerül sor. Természetesen a gépen belül az adatbázis a bemeneti részfolyamatban belső outputként, a kimenetiben pedig belső inputként szolgál.) A feldolgozás felhasználói tevékenységei fölött a vezérlési műveletek gyakorolják az irányítást (vö. programlogika). A tevékenységeket pedig alulról technikai szolgáltatások (pl. hibakezelés, eligazítás) támogatják.

— ***Az adatbázist egyetlen nagy rendszerfolyamatként kell felfogni.***

Magyarázat: Minden információs rendszernek létezik egy általános - tehát a tárgyától teljesen független - (nem pejoratív értelemben vett) sablonos felépítése. Ennek az átfogó rendszerszerkezetnek a logikáját kell, hogy kövesse az adatbázis makrostruktúrája is, függetlenül attól, hogy egy kórház, egy bank vagy egy termelővállalat rendszeréről van-e szó. Ez a sablonos architektúra az adatfeldolgozási folyamat logikájának felel meg, csak éppen nem egy adott tevékenységre, hanem a feldolgozások összességére vonatkozik. A sablont az alábbi kereszt formájában képzelhetjük el:



15.1 ábra: Az információs rendszer általános felépítésének sablonja

Az ábrán mutatott öt tényezőcsoport mindegyike egyenként is összetett szerkezetű. Most először csak a lényegüket magyarázzuk el; összetevőiket nem részletezzük.

**„Érdemi adatfeldolgozás” (A).** Ez áll a rendszerséma középpontjában. Az „érdemi” jelző leginkább a többi - „nem-érdemi” - tényezővel szembeállítva érthető meg. Ennek dacára megpróbáljuk pozitívan is megmagyarázni. A kórháznak az a feladata, hogy betegeket lásson el. A biztosítónak az, hogy pénzügyi kárainkat csökkentse. A vasút személyeket és árukat szállít. Minden szervezetnek van egy - olykor összetett - alapvető funkciója. A közvetlenül az ezzel a feladatkörrel összefüggő ismereteket nevezzük itt érdemieknak, az azokkal kapcsolatos informatikai feladatok összességét pedig érdemi adatfeldolgozásnak. Az architektúra összes többi tényezője csak arra szolgál, hogy ezt az érdemi részt valamilyen értelemben kiszolgálja. Mivel a központi tényezőt a következő pontban bővebben is kifejtjük, egyelőre nincs több mondanivalónk róla.

**Előkészítés/adminisztráció (B).** Az olvasó jobban megértheti a dolgot, ha eláruljuk, hogy ábránk középső vonala a közismert input-adatbázis-output folyamatnak felel meg. Vagyis a baloldali blokk az inputfunkcióké. Ez is egy igen összetett feladatcsoport, ami felöleli a bizonylatok kontírozásával, adminisztrációjával, tág értelemben az események kezelésével kapcsolatos részfeladatokat. Egyetlen példával mutatjuk be, hogy miért van szükség ennek a blokknak az általánosítására.

A szerző látott olyan adatbázis terveket, amelyek megszegve az alapvető szabályokat az adminisztrációs adatokat is az érdemi egyedekhez kötötten határozták meg. Például az X egyedben szerepelt egy adatsor, ami elárulta, hogy ki volt az adatrögzítő, mikor történt az adatkezelés, ki kontírozta a bizonylatot, mikor helyezték azt a dokumentumtárba, hol található azon belül stb. Majd az Y, Z stb. egyedekben is szerepelt egy-egy hasonló célú - de természetesen az előzővel nem azonos felépítésű - technikai/adminisztrációs adatsor. Ez persze nyilván azt is jelenti, hogy abban a „rendszerben” N-féle programmal és módon kezelik ezeket az elvileg azonos célú és jellegű technikai ismereteket.

Számunkra világos, hogy ha egyes papírokat a bizonylattárba kell helyezni és erről a tényről adatokat kell vezetni, akkor ezt integráltan - vagyis technikai tábláknak egy közös halmazával - kell megoldani. Ha szükség van az események vezetésére, akkor azok kezelésére egyféle logikát kell alkalmazni (← *Eseménymodellezés*). Mindezt pedig nem egyféle pedantéria mondatja velünk. Ha az adminisztrációs ismeretek szanaszét vannak szórva az érdemi egyedekben, akkor ez nem csak kezelési káoszra vezet. Sem azt nem lehet megállapítani, hogy ki mi mindenért felelős; sem azt, hogy mi minden történt egy dokumentummal; sem azt, hogy adott időben milyen események zajlottak; sem azt, hogy a dokumentumtárban mik találhatók és mikor kerültek oda stb.

**Szolgáltatás/adminisztráció (C).** Jobboldalt látható a szolgáltató funkciók sora. Ide elsősorban az on-line kereső és az off-line kiírató funkciók tartoznak. Továbbá minden olyan adminisztráció, ami az output jellegével, felhasználóival, sorsával kapcsolatos. Az előző tétellel kapcsolatos mondanivalót itt nem ismételjük meg. Ámde világos, hogy a lényegét ismerő tervező sohasem fog például kimenetet vezérlő adatokat tenni az érdemi egyedekbe, amint azt a szerző oly gyakran tapasztalja. Az is furcsa, hogy egyes cégeknél szinte naponta készítenek újabb és újabb programokat az outputok előállítására, noha meglehetősen jól általánosítható feladatról van szó. A probléma éppen az, hogy egyesek nem veszik észre a szolgáltatási - tehát általánosítható - jellegét. Itt kell említést tennünk az imázsról is. Mivel nincsenek generikusan vezérelt szolgáltatási funkciók, hol ilyen, hol olyan kinézetű számlát kapunk például egy pénzintézettől.

**Kontrol (D).** A sablon felső részén - nem véletlenül ott - látható rendszerblokk az angol control szó minden jelentését felöleli. Ide tartozik az irányítás, a vezérlés és az ellenőrzés is. Itt nem a programbeli hasonló műveletszerepekre kell gondolni! Magának a teljes adatbázisrendszernek a működési logikájáról van szó. Tehát például az ellenőrzés nem a mikroszintű adatvalidálást, hanem a makroszintű rendszerellenőrzést jelenti.

Ez a felső szintű kontrol nemcsak az érdemi blokk, hanem az összes többi működését is vezérli. Adatmodellezési szempontból különösen érdekes a számunkra. A kontrol funkció csak akkor valósítható meg, ha a rendszer összes többi tényezőjére vonatkozóan ismeri a szerkezeteket, tartalmakat, formákat. Amint azt már tudjuk, ezeket a *metaadatbázisban* tároljuk és kezeljük. A gyógyszerdózis, a biztosítási módozat, a vasúti vonatszám érdemi felhasználói adat. Az ezek sajátosságait rögzítő ismeret viszont olyan metaadat, ami az előbbieket használatát kontrolálja.

**Támogatás (E).** Az érdemi rész alatt található az úgynevezett támogató funkciók sora. Ezekre az jellemző, hogy az összes többi blokkot mintegy alulról kiszolgálják. Az olvasó számára a feladatcsoportok közül leginkább a biztonság-integritás-hozzáférés hármasa ismert. Mivel az „összes többi” kitéltet alkalmaztuk, világos, hogy ezekre a funkciókra közös megoldásokat kellene kidolgozni. A jobb érthetőség kedvéért egy negatív példával fogunk élni. A szerző látott olyan rendszereket - sőt, csak olyanokat látott -, amikben a mondjuk bérszámfejtési alrendszerbe beledrótoztak egyfajta hozzáférési jog ellenőrzést, a könyvelési alrendszerbe egy másikat, az

anyaggazdálkodásiba egy harmadikat stb. Tehát a vonatkozó szervezetnek nincs integrált hozzáférési szisztémája!

Teljesen világos, hogy a támogató funkció részblokkjait a tárgytól - bérszámfejtés, könyvelés stb. - függetlenül kell kialakítani. Ezt nemcsak objektív, hanem szubjektív érvek is indokolják. Egy rendszer hatékonysága nem csak a technikai tényezőkön múlik. Ha a felhasználónak mindig másféle hozzáféréskezeléssel, helprendszerrel, menüvel stb. kell kínlódnia az egyetlen szabványos helyett, akkor nem fogja tudni hatékonyan kezelni az érdemi adatokat. Egyébként is miért kellene egy szervezetben húszféle helprendszert készíteni egyetlen egy helyett? És nevezhető-e integráltnak az olyan rendszer, amiben ötvenféle hibakezelés található? Más a könyvelésben, mint a műszaki nyilvántartásban?

Összegzés: Nemcsak szeretni, hanem tudni is kell integrált rendszert készíteni. Ennek a tudásnak az a titka, hogy ismerjük az információs rendszer általános szerkezeti sablonját, vagyis az adatbázist egyetlen rendszerfolyamatként fogjuk fel. Ha ehhez ragaszkodunk, akkor ezt a struktúrát adatmodellünk is tükrözni fogja. A modellnek lesznek érdemi és technikai fejezetei. Az előkészítő, a szolgáltató, a kontrol és a támogató blokkokat külön egységekként tervezzük meg úgy, hogy azok tartalmát összekötjük, de nem vegyítjük az érdemiével. Végül az is világos, hogy a nem-érdemi blokkok egyenként is igen összetett felépítésűek, vagyis rendszert alkotnak a rendszeren, modellt a modellen belül.

### 15.3 Tájérolódás a modellben

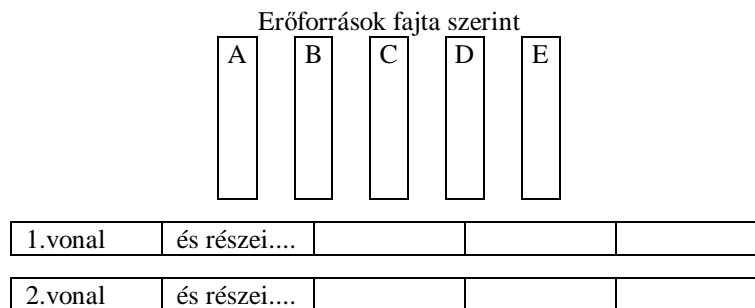
Háttér: Minden szervezet az általa választott és/vagy a számára kijelölt célok elérése illetve a feladatok teljesítése érdekében adott funkciókat lát el és azok végrehajtásához meghatározott erőforrásokat vesz igénybe.

A **funkciók** specifikusak, hiszen például más egy kórháznak a célja és a feladata, mint egy biztosítónak. A közkórház és a klinika eltérően működik. Az egyik biztosítónál van nyugdíjpénztár is, a másikinál nincs. Az egyik életbiztosításra specializálódik, a másik az élet- mellett a vagyonbiztosítást is műveli. Ezeket a legmagasabb szintű funkcióköröket, amik belül tovább is tagolhatók (például a vagyonbiztosításnak lehet több tucat fajtája), a szervezet tevékenységi **fővonalainak** [line] nevezzük. Ezek időben is szakaszolhatók. Például egy olajvállalatnál a vonal az olajkutatásnál kezdődik, a kitermeléssel folytatódik, ezt esetleg követi a finomítás, a szétterítés után pedig a vonal az értékesítéssel záródik. Egy cégnél létezhetnek egymással párhuzamos - azonos jellegű részekből álló - vonalak is. Például a szénhidrogénnel foglalkozó cégnél lehet olajvonallal mellett földgázvonallal is.

Az **erőforrások** nem specifikusak. A kórházban éppen úgy szükség van eszközökre, mint az olajvállalatnál. Mindkettőben feladat a humán, pénzügyi, anyag stb. erőforrással való gazdálkodás. Ez a feladat a szervezeten belül is általános. Az olajvonallalhoz éppen úgy erőforrásokat kell rendelni, mint a földgázvonallalhoz. Ezért a normálisan szervezett cégek a gazdálkodási feladatokat - de azok legfelsőbb szintű kontrolját mindenképpen - központi szervezetekben látják el. E szervezetek vezetői az irányító **törzskar** [staff] tagjai, ezért - bár nem teljesen korrekten, mert a törzskarban mások is részt vesznek - az erőforrásokkal való gazdálkodást itt a törzskarhoz kötjük.

A line és a staff viszonyának a sablonját a következő ábra mutatja:





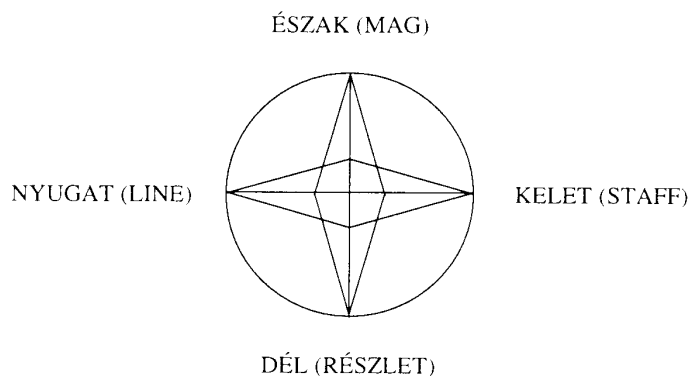
15.2 ábra: A funkciók és az erőforrások viszonya

Adatmodellezési szempontból a „staff”-ok ismereteit mindenképpen integrálni kell. Az egy kérdés, hogy a modern vállalatban az üzleti egységek [business units] az őket érintő ismereteket akár fizikailag különálló adatbázisrészekben is vezethetik. Az pedig már egy másik kérdés, hogy ezek lényegét össze kell hangolni, különben sohasem lehetne vállalati szinten mérlegelni például az anyagfelhasználást.

— ***A megismerés örökérvényű szabályai szerint a jelenségeket általános, specifikus és egyedi kategóriákba soroljuk. Az adatmodellt ezeknek a kategóriáknak megfelelően kell strukturálni.***

Magyarázat: A modellezésben kulcsszó az általános, mivel a modell az általános, lényeges és tartós jegyeket rögzíti. Ezért a fenti hármast - természetesen csak modellezési szempontból - úgy is megfogalmazhatnánk, hogy nagyon, közepesen és kevésbé fontos. Ezt a kitétele nem szabad félreérteni. Ha egy vízügyi vállalatnál előtérbe lép a vízminőség problémája, akkor a vízminőség mérésével kapcsolatos adatok időlegesen a figyelem középpontjába kerülnek, fontosakká válnak. Mi nem ilyen „aktuálpolitikai” szempontból mérlegeljük a fontosságot. A vonatkozó adatok egyedi felhasználásúak, mert a vállalatban belül egy igen kis csoport érdekelt bennük tartósan is. Ők a maguk módján formálhatják meg a vízminőségi idősorok modellrészletét mások beleszólása nélkül. Ezzel szemben például a települések adatai általában nem tűnnek lényegeseknek. Viszont nincsen olyan vállalati részleg, amely ne lenne érdekelt ezekben az ismeretekben. Ezért ezek az adatok általánosak, tehát nagyon fontosak és csak közösen lehet őket kialakítani.

Most pedig térjünk át a strukturálásnak a fenti szabályra épülő elvére. A következő ábra szemlélteti az összefüggéseket:



15.3 ábra: Négy égtáj az adatmodellben

**Mag** (észak). Az adatmodellezésben mag [kernel] egyedtípusoknak hívjuk az általános érdeklődésre számot tartó, tehát igen fontos jelenségek csoportjait. Érdekes módon ezek az egyedek két igen szélsőséges kategóriába sorolhatók. Ezeket például világítjuk meg. A szerző szinte nem ismer olyan információs rendszerrészt, amiben a *partner* ne játszana igen fontos szerepet. Ez a legáltalánosabb - és ezért a leglényegesebb - fogalmak egyike. Olyan érdemi mag egyedben tükröződő, ami minden felhasználót kiszolgál. A másik oldalon vegyük például a *mértékegységet*. A kódolt adatokra érdemes közös kódtárakat kialakítani, hogy egységes nyelvet beszéljünk. Ezek a kódtárak is általános használatúak, ezért modellezési szempontból nagyon fontosak. Végeredményben a magot a tartalmilag is igen lényeges egyedek mellett a formailag is integrálható segédegységek képezik. (NB.: Magában az adatmodellben is felül található az alárendeltekkel tovább részletezendő, ritka kivételtől eltekintve további lényegi fölérendeltekhez nem kapcsolódó egyedek illetve az azokat minősítő/osztályozó kód jellegű egyedek.)

**Staff** (kelet) és **Line** (nyugat). A legáltalánosabb és leglényegesebb jelenségekhez a második szinten kapcsolódnak a specifikusabbak, de még nagyon fontosak. Egy példával világítjuk meg a mondanivalót. A daruskocsi kettős természetű lényeg. Az egyik oldalon olyan műszaki eszköz (gép), amellyel meghatározott feladatokat lehet végrehajtani (line). A másik oldalon olyan vagyontárgy (erőforrás), amely pénzbe kerül és pénzt fial (staff). A daruskocsi mint objektum mind a két oldal számára fontos, ezért az általános, lényeges és tartós ismérvei nyilvánvalóan a magba tartoznak. Ugyanakkor fontos műszaki ismeret a daruskocsi technikai paramétersora (feszítáv, teherbírás stb.) illetve felépítése (alapjármű, daruszerkezet, motor stb.), ami már csak a műszaki line oldalra tartozik. A daruskocsira mint tárgyi eszközre amortizációt kell számolni, ami a gazdasági staff oldal érdekszférája. Tehát a jobb- és baloldal valóban a mag kibővítésére szolgál. (NB.: A modellben az alapegységekhez azokat bővítő alárendelt kiegészítő illetve altípus egyedek tartoznak.)

Megjegyzés: Az adatmodellt sohasem szabad egyszintű sablonként elképzelni. A modell olyan, mint a körön belüli kör. A line maga is égtájakra bontandó. Azért, mert egy adott vonalon belül vannak olyan jelenségek, amikben több részvonal felhasználói érdekeltek, viszont vannak olyanok is, amelyek még specifikusabbak. Például a gépek karbantartása általánosabb ismeretkör, mint a gépek jellegtől függő tipizálása.

**Részlet** (dél). A szemléltetés kedvéért az olvasó gondoljon a térképrendszerre. A főbb tényezőket alaptérképek rögzítik (mag). A térképek egy csoportja arra szolgál, hogy ilyen-olyan aspektust (vízrajz, időjárás, búzatermelés stb.) emeljen ki (line és staff). Ezen kívül léteznek olyan térképek is, amik az eredetieket mintegy kinagyítják illetve újabb részletekkel gazdagítják. Az adatmodellező tájolasát segítő iránytű alján a legkevesbé lényeges, tehát közelítőleg egyedi jelenségek találhatók. Ismét csak nem arról van szó, hogy valakik számára ezek a dolog ne lennének roppantul fontosak. Hanem arról, hogy a részletezés kevésbé általános érdeklődésre tart számot és ugyanakkor nem fejthető ki a generikusabb tényezők ismerete nélkül. Például a vízminőség mérési adatsora csak akkor fogalmazható meg, ha már ismert a vízminőség mérési pont lényege. (NB.: A részletező egyedek a modell legáján helyezkednek el.)

Összegzés: Az adatmodell struktúráját nem területi, funkcionális vagy szervezeti alapon kell kialakítani, mert az nem vezet integrált megoldásra. Az egymásra épültség és a közös használat - végeredményben az általánosság és a fontosság - a mérvadó. Nem tévedünk el, ha az architektúrában a modellezési iránytűt követjük.

## 15.4 Modellkeret

—

### *A modellezés első lépése a modellkeret megteremtése.*

Magyarázat: Az adatmodellezés nem egy részfeladat adatbázistervének a kialakítását jelenti. „Az adatmodell - egy” elv azt mondja ki, hogy a szervezet számára fontos összes általános, lényeges és tartós ismeretet egy modellben kell összefogni akkor is, ha maga a megvalósítás több logikai/fizikai adatbázisban történik. Az elvből az következik, hogy az egyetlen adatmodell meglehetősen összetett és nagyméretű. Ez pedig egyet jelent azzal, hogy nem áttekinthető, hacsak nem alkalmazunk valamilyen ésszerű tagolást, vagyis ha nem hozunk létre egy olyan modellkeretet, amivel az adatmodellt egymással logikusan összefüggő részekre bontjuk. A kettős kérdés tehát az, hogy mi ennek a keretnek a lényege illetve hogyan lehet azt kialakítani?

Segédfogalmak: A szervezetben az erőforrásgazdálkodás (staff) és az alaptevékenység (line) feladatait funkciócsoportokra bonthatjuk le. Ilyen például az értékesítés, a raktárgazdálkodás a számlázás, a beruházás stb. Mivel itt nem részletes és pontos behatárolásról, hanem csak átfogóan szemlélt tényezőkről van szó, ezeket a csoportokat *funkcióköröknek* hívjuk. E körök nem függetlenek egymástól. Köztük kétféle jellegű kapcsolatot fedezhetünk fel. Léteznek közvetlen időbeli/logikai viszonyok. A kitermelést időben követi a szállítás. Vannak közvetettebb kapcsolódások is, amik a közösen használt adatokon át valósulnak meg. Ilyen jellegű például az értékesítés és a raktárgazdálkodás összefüggése. Ha eladunk valamit, akkor azt termelés-sel/beszerzéssel kell pótolni.

Az integrációs keret megteremtésekor még nem foglalkozhatunk a részletekkel. Ekkor még nem az a fontos a számunkra, hogy a cikk jelenséget milyen adatsorral fogjuk leírni. Csak azt kell tudnunk, hogy vannak megrendelések, cikkek, készletek, számlák stb. Ezek az ismeretekkel tükröződő jelenségek ekkor még nem egyed típusokként jelentkeznek. A megrendelés, a cikk stb. ismeretei a végső adatmodellben feltehetően több egyed típusban fognak megtestesülni, de ez egyelőre minket nem érdekel. A keret megalkotásához ún. globális - később részletezendő - egyedekben kell gondolkodnunk. Ezeket *adatköröknek* nevezzük. Az adatoknak lesz a cikkekkel, a rendelésekkel, a számlákkal stb. kapcsolatos köre úgy, hogy a részletekben ezek közvetve vagy közvetlenül összefüggenek egymással.

	A	B	C	D	E	F
1			*	*	*	*
2		*	*	*	*	
3	*	*		*	*	
4	*	*			*	
5					*	

15.4 ábra: Sablonos adathasználati mátrix

Magyarázat: Az adathasználati mátrix a funkció- és adatkörök kapcsolatait tárja fel. Itt és most számokkal jelöltük az előbbieket és betűkkel az utóbbiakat. A mátrix bejegyzései csak annyit mutatnak, hogy a funkció (pl. 3) használja-e (A) vagy sem (C) a megfelelő adatkört. A mátrix megfelelő átalakításával - az itt ki nem fejtendő klaszteranalízissel - kimutatható, hogy mely

körök között létezik szorosabb kapcsolat és melyek között nem. Példánk esetében az alábbi következtetéseket vonhatjuk le:

- Egy-egy rendszerrészt alkotnak a 3,4/A,B és az 1,2/C,D tényezőegységek.
- A két rendszerrész a B/2 és D/3 párosokon át kapcsolódik egymáshoz.
- Az 1,2,3,4,5/E sorozat azt mutatja, hogy létezik egy közös fölöttes mag is.
- Az 1/F páros teljesen egyedi kapcsolódást jelez.

Az ilyen elemzés kiválóan alkalmas a modellkeret megteremtésére. Ugyanis a keret a következő módon épül fel. Az előző pontban tárgyalt égtájaknak megfelelően egy (tovább részletezett) közös fejezetben tartalmazza a **mag** egyedeket (lásd E). Annyi részfejezetben rögzíti a **line** és **staff** egyedeket, ahány rendszerrészt az elemzés során feltártunk. Tehát példánk esetében lesz egy fejezet az A,B és egy másik a C,D egyedekkel. Végül a modell végén jelennek meg funkciókörök szerint tagoltan a részletező egyedek (lásd F). Persze a valóságban az ittenieknél sokkal több adatkör fog alkotni egy-egy modellrészt. Mindez az érdemi adatokra vonatkozik. A kontrol, támogató stb. tényezők külön részekbe tartoznak.

Kiegészítés: A fenti módszer *eltérő mélységekben* alkalmazható. A funkciókörökön belül újabbak alakíthatók ki és ugyanez megtehető az adatkörökkel is. Vagyis a mátrix adott részét mintegy kinagyítva egyre pontosabb modellrészletekhez juthatunk. A végső modelldokumentációban ezek a részletek fogják képviselni a fejezeteken belüli pontokat, esetleg még további tagolás esetén az alpontokat is.

Külön problémát jelenthetnek a *részlegesen közös tényezők*. A kérdés az, hogy vajon azoknak hol a helye a kereten illetve az annak szerkezetét követő dokumentáción belül? Ez nyilván a közös jelleg erősségétől függ. Például egy közüzemi szolgáltató vállalatban a fogyasztási hely adataira a mérnököknek éppenolyan szüksége van, mint a számlázóknak. Ebben az esetben a viták megelőzése érdekében e tényezőt feltoljuk a magba. Egyébként a megfelelő keresztutalás megadása után valamelyik rendszerrészbe helyezzük. Például a számlák ismereteiben nagyon sok funkciókör érdekelt lehet, de nem logikátlan, ha ezt az adatkört a számlák kezelését mintegy összefogó könyvelési rendszerrészben képzeljük el.

Végül meg kell említeni, hogy néha szükség van *technikai* tagolásra is, vagy legalábbis célszerű ilyesmit alkalmazni. Például a folyamat-jelleg indokolhatja azt. Léteznek olyan modellek, amelyekben vannak bemenet- és kimenetoldali egyedek. Ilyesmi a felmérések és a statisztikák esetében fordul elő. Ilyenkor célszerű lehet ezeket egy-egy modellrészbe összefogni akkor is, ha az egyedek tartalmilag nem kapcsolódnak egymáshoz. Ugyanis nem lenne okos húszféle felmérés esetén az egyedeknek húsz külön fejezetet szentelni.

Összefoglalás: A teljes adatmodellt részekre kell bontani. A részek makroelrendezése adja a modell keretét. Ez tudatosan is kialakítható a fentebb bemutatott módszer szerint. Azonban nem annak a merev követése, hanem a józan és természetes tagolás a fontos.

## 15.5 Egyedszerepek

—

*Az adatmodellt az egyedek szerepei szerint érdemes felépíteni.*

**Magyarázat:** A szerző nagyon sokszor találkozik olyan adatbázistervvel - sőt, mással nem is -, amiben a fejezeteken, modellrészekben belül (már ha ilyenek vannak) teljesen ad hoc sorrendben, minden rendezőelv nélkül jelennek meg az egyedek. A tervezők nagy része ezek szerint nem tudja, hogy az egyedeket azok feladata, jellege, kapcsolódási módja - egyszóval: szerepe - szerint is célszerű osztályozni. Az alábbiakban az eltérő szerepeket mutatjuk be a gépek közös példáján keresztül.

**Mag.** A gépek egyszeres, általános és közös ismereteit őrzi. Köré épülnek a további, a többszörös, speciális és nem-közös ismereteket befogadó további egyedek. A gépet annak ellenére magnak tekintjük, hogy egy bizonyos szempontból altípus, mégpedig a generikus vagyonelem főtípusnak az egyik specializációja. Számos ok miatt nem alkalmazhatunk több tucatnyi gép egyedtípust annak dacára sem, hogy a szivattyú távolról sem hasonlít például az emelőgépre. Erre a problémakörre alább még visszatérünk.

**Osztályozó.** Az X és az Y szivattyú két külön fizikai objektum, de teljesítményük, áruk, felépítésük stb. azonos. Ezért megkülönböztethetünk szivattyútípusokat, amiket saját adatokkal írhatunk le (a kocsitípus analógiájára). A magegyedeket több szempont szerint osztályozó egyedekhez köthetjük. Az osztályozó egyed mindig a magegyed 1:N fokú fölérendeltje. Adott esetben osztályozási *hierarchiáról* lehet szó, vagyis a gépnek van típusa, azon belül altípusa, azon belül jellege stb. (Ha valaki nagyon akarja, akkor ide értheti a kódegyedeket is, de mi azokkal a fogalmi szinten nem foglalkozunk.)

**Leíró.** A szivattyúnak van egy maximális kapacitása. Ezt sohasem használják ki. A használati hely, mód, körülmény függvényében a szivattyút adott kapacitásszinteken kell működtetni. Ez a többszörös előírás a magegyedhez kapcsolt olyan alárendeltben jelenik meg, ami azzal N:1 fokú viszonyban áll, de máshová nem kapcsolódik vagy az a viszonya csak osztályozási jellegű. A szivattyú által elfogyasztott áram mint idősoros egyed szintén a maghoz tisztán hierarchikusan kapcsolódó ismeretsor. A lényeg az, hogy a leíró egyed nem mutat ki másik magegyed felé (lásd majd a társító egyedet).

**Kiegészítő.** Ha a gépeket egy egyedbe fogjuk össze, akkor igen hosszú lesz annak tulajdonságsora és sok lesz az üres érték. A szivattyúnak nem ugyanazok a specifikumai, mint az emelőgépnek, jóllehet mindkettőnek van kapacitása, gyártója, gyártási száma stb. A speciális ismereteket érdemes kiegészítő egyedekbe helyezni (← *Feltételes függés*). Az ilyen egyed nem altípus, mert a kiemelés nem alternáló módon történik. A kiegészítő egyed a magegyeddel 1:1 fokú felülről opcionális kapcsolatban áll.

**Társító.** Idegen szóval asszociatív egyed. Feladata az, hogy két magjellegű egyed között viszonyt teremtsen. Például a dinamikus modellben nem a gép tulajdonságaként vezetjük a leltárhelyet, mert akkor nem tudjuk nyomonkövetni a gép leltárhelyek közötti mozgását. Inkább létrehozunk egy gép/leltárhely társító egyedet, ami a két jelenség közötti M:N-es viszonyt írja le ismeretekkel. Tehát az ilyen egyed mindig hálós viszonyt testesít meg. Az itt felmerülő sajátos kérdés az, hogy az asszociatív egyed melyik modellrészben kerüljön megfogalmazásra? A gépeknél vagy a leltárhelyeknél? A válasz egyszerű: az előbbieknél. Ennek két oka is van. Egyrészt a gép az elsődleges(ebb) objektum. Másrészt a leltárhelyek nemcsak a gépekhez, hanem más jelenségekhez is kapcsolódnak (pl. szoftverekhez). Ha náluk vennénk fel a társító egyedeket is, akkor az elbonyolítaná a kérdéses modellrészt.

**Struktúráló.** Itt a (dinamikus) családfákra kell gondolni. A gép tartozékokból és/vagy alkatrészekből épül fel. Mi több, a gépnek a része is lehet gép. Például az emelőgépnek és a szivattyúnak is van motorja, ami külön is leírandó objektum lehet. Másik példával élve a vízvezeték önmagában is leírandó és úgy is tükrözendő, mint a vezetékhálózat része. Egy magegyedhez több M:N fokú struktúráló alárendelt egyed is tartozhat. Például egy autógyárban a gépkocsinak mint terméknek van egy *technikai* darabjegyzéke, ami arról árulkodik, hogy a kocsni milyen főszerelvényekből, szerelvényekből, alkatrészekből áll. Ugyanakkor létezik egy *technológiai* darabjegyzéke is, ami szintén családfa, de nem azonos az előzővel. Azt tükrözi, hogy a gépkocsit milyen sorrendben kell összeszerelni.

**Altípus.** Általános sajátosságai alapján a gépeket egy egyedtípusban tükrözzük. Ezzel a megoldással a gépeket elválasztjuk az ingatlanoktól, amik más absztrakciós síkon a gépekkel együtt egy osztályt alkotnak, hiszen mind a kétféle dolog vagyonelem. Noha nem létezik két teljesen azonos gép - hiszen mindegyikük *egyedi* - az általánosítás, az egyetlen GÉP egyed alkalmazása mégis jogos. Feltétlenül az, hiszen a hétköznapiak során is használjuk a teljesen általános „gép” fogalmat. Ugyanakkor az X gép jobban hasonlít az Y-ra, mint a Z-re. Az X gép elektromotoros fűnyíró, az Y benzinmotoros fűnyíró, a Z pedig egy elektromotoros permetezőgép.

Az általános gép fogalom és az egyedi konkrét gép közötti táv határon belül számos lehetőségünk nyílik arra, hogy a *különös* vonások alapján kialakítsuk a gépek alosztályait. Példánk esetében a megkülönböztetés történhetne a funkció szerint (fűnyíró, permetező), de hasonlóan logikus lehet a meghajtás (áram, benzin) szerinti besorolás is. Eltekintve az *üres értékektől* (← *Feltételes függés*), nincs semmilyen egzakt módszer arra nézve, hogy milyen alapon kell összevonni illetve megbontani (← *Generalizáció és specializáció*) az egyedelőfordulások - esetünkben a gépek - halmazát.

— ***Kerüljük a bonyolult és a többszörös altípus-hierarchiát.***

Magyarázat: Mivel nem ismerhetjük sem a konkrét sajátosságokat, sem a gépek fajtáit, azt természetesen nem tudjuk megmondani, hogy a gépekhez milyen altípus egyedeket kell kapcsolni egy adott szervezetben. Ezért a tervezőknek csak a szabályban foglalt kettős tanács magyarázatával szolgálhatunk.

A specializált egyedek ***altípus-hierarchiát*** alkotnak. A gép egyik alosztálya a szivattyú, annak pedig további alosztálya lehetne a benzinmotoros szivattyú. A szabály első része egy általánosabb elv sajátos vetülete. Általában véve ha hierarchiát alkalmazunk (legyen szó egy szervezet felépítéséről vagy éppen egy könyv szerkezetéről), törekedni kell az egyensúlyosságra. Ez két dolgot jelent. Egyrészt a hierarchia ne legyen túl mély, tehát ne keressük az al-alosztály még további alosztályát. Másrészt ne legyen túl széles, tehát ne tervezzünk annyi altípust, hogy azok szinte már az egyedi gépfajtákat tükrözzék.

Többszörös altípus-hierarchiáról akkor van szó, ha a specializációt több ismérv mentén is végrehajtjuk. Például a gépeket funkció szerint is, energiaforrás szerint is osztályozzuk. Ekkor az elektromos fűnyíróra vonatkozó ismeretek a gép alapegyedben, a fűnyíró gép és az elektromos gép altípusban „szóródnak szét”, ami roppantul elbonyolítja nemcsak az adatmodellt, hanem magát a kezelést is.

Probléma: Számos olyan *gyakorlati* helyzettel találkozhatunk, amiben egyszerűen nincs módunk arra, hogy az *elméleti* elveket tisztán érvényesítsük. A modellezés feladata a valóság hű tükrözése, de nem feladata a rossz valóság jobbítása. Két példával világítjuk meg ezt a rébuszt. A

józanul gondolkodó ember számára a konkrét gép olyan objektum, aminek adott esetben vannak tartozékai is. Azt nem is tudja elképzelni, hogy egy cégnél az X gépnél külön tételes nyilvántartást vezetnek a tartozékokról, viszont az Y gépnél - ami az előbbivel teljesen azonos jellegű és ugyanolyanok a tartozékai is - csak egy laza felsorolásban közlik, hogy van N darab valamiféle kelléke is. Hasonlóképpen meglepő, ha egy cég Q üzemegységében lévő minden műszaki eszköz külön leltári számot kap (amint az illik), miközben a teljesen azonos jellegű Z üzemegységet egyetlen egy leltári számon futtatják, jóllehet található benne ötven külön is azonosítható gép.

Összefoglalás: A jó tervező tudja, hogy miképpen kell a mag-egyedek köré építeni a velük kapcsolatos osztályozó, kiegészítő, altípus stb. részletező egyedeket. Gyakorlatilag nincs mindig módja arra, hogy a valóságot elvileg korrekten tükrözze. Ha a fentiekhez hasonló kaotikus helyzetek uralkodnak egy szervezetben, akkor két eset lehetséges. Ha a probléma nem súlyos és a vezetés hajlamos a gyakorlat megváltoztatására, akkor esetleg alkalmazhatunk szigorúan csak időlegesnek szánt kompromisszumos megoldásokat. Így például kétféleképpen modellezzük a tartozékokat (külön egyedekként is, általános leíró tulajdonságokként is), de nem engedjük meg a kompromisszumos megoldás alkalmazását karbantartási módban. Azaz a régi gépeknél a tartozékok még leírásokként szerepelnek, de új gépnél a tartozékokat már csak külön egyedekként lehet megadni. Ha viszont nem látunk szándékot a valós gyakorlat jobbítására, akkor mindenképpen le kell mondanunk a modellezésről. Azért, mert mi leszünk erkölcsileg is felelősek a káros konzerválásáért. A lelkiismeretes adatmodellező sohasem feledkezhet el az alábbi szabályról:

— ***Káoszt nem lehet modellezni.***

## 15.6 Sajátos megoldások

— ***Nem létezik jó adatmodell józan kompromisszumok nélkül.***

Magyarázat: Az előző pont végén mutattunk példát arra, hogy a tervező olykor a legjobb szándéka ellenére sem képes „tisztá” adatmodellt építeni, mert maga a valóság, a szervezet valós gyakorlata elfogadhatatlanul rossz. Itt önként felmerül az a kérdés, hogy mikor tekinthető a gyakorlat elfogadhatóan rossznak, vagyis mikor van lehetőség arra, hogy a jót a rosszal józan kompromisszumok árán ötvözzük? A válasz megértéséhez azt kell tudni, hogy a „rossz” különböző természetű lehet. Az alábbiakban néhány sajátos tipikus esetet vázolunk fel a kompromisszumos megoldások szemléltetésére.

**Vásárolt szoftver.** A nagy, (ki tudja miért) „alkalmazásoknak” nevezett szoftverekre, a kisebb célspecifikus (pl. könyvelési, bérszámfejtési) programcsomagokra és a multimédia (pl. térgeometria) varázslataira több dolog jellemző. Egyrészt az, hogy vevője egyáltalán nem változtathatja meg a termék mögötti adatállományok szerkezetét és tartalmát, sőt, sok esetben még meg sem ismerheti. Másrészt az, hogy igen ritka kivételektől eltekintve a mögöttes állományhalmaz nem valódi adatbázis, azaz struktúrája nem átgondolt, nem normalizált, nem optimális. Harmadrészt ha netán mégis az lenne (ami szinte kizárt), akkor is az abban használt azonosítók és kódok nem

felelnek meg a mi igényeinknek, a csoport és szerepnév konstrukció pedig ismeretlen ezen szoftverekben ( $\leftarrow$  CSAKIS).

A tervező igen kényelmetlen döntés előtt áll, ha adatmodelljét integrálni kell az ilyen nem adatmodell-alapú szoftverek állományterveivel. Arról nyilván szó sem lehet, hogy a szoftver-szállító változtasson a saját „adatbázisán”. Az pedig nem lenne sem erkölcsös, sem okos dolog, ha a modellező mások kedvéért lerontaná a saját tervét. Ezt már csak azért sem teheti meg, mert sokszor nem egyetlen, hanem több vásárolt termékhez kell kapcsolatot teremteni úgy, hogy azok belső logikája egymással sem harmonizál. Például többféle térgeometriai programcsomagunk van, amelyek a térképobjektumokat másképpen azonosítják - ergo nincs lehetőség az egyikféle azonosítás átvételére.

**15.1 példa** KAPCSOLÓ (Objektumtípus-1, Objektumazonosító-1, Objektumtípus-2, Objektumazonosító-2, Kezdődátum, Záródátum)

Megoldás: A korábbiakban már volt szó a „kamu” egyedeiről ( $\leftarrow$  *Álegyede*). Ehhez kell fordulnunk a jelen esetben is. A szerző *fordítókorongnak* nevezi azt a technikai egyedet, ami az általa jól megszerkesztett modellnek megfelelő adatbázis és a korábbi fejlesztésű illetve idegen szoftverek tényezői között tart kapcsolatot. A mesterséges egyed első két tétele a mi egyedtípusunk egy előfordulását határolja be. Ehhez egy vagy több, a második két tétel által hivatkozott előfordulás kapcsolódhat a szállított szoftvereknek megfelelő állományokban. Több szoftver esetén az Objektumtípus-2 természetesen arra is utal, hogy melyik programcsomagról van szó. Több ok miatt - például szoftververziók - esetleg szükség lehet az érvényesség kezdő- és záródátumának a megadására is.

**Ismeretlen altípus.** Vegyük például a szerződéseket! Egy szervezetben belül létezhet akár ötvenféle szerződés is. Nyilvánvaló, hogy azokat nem fogjuk ugyanennyi egyedben tükrözni. Mindenképpen létre fogunk hozni egyetlen SZERZŐDÉS főtípust és szükség szerint alakítunk ki néhány - semmiképpen nem ötvenféle - altípust. Azonban a törvények és a jogszabályok változása miatt egyes szerződésfajták megszűnnek, mások sajátosságai módosulnak, illetve nyilván születhetnek újféle szerződések is. Ilyenkor a tapasztalatlan tervező kapkodó módon új altípust alkot és átíratja a programokat ami azért baj, mert a tapasztalatok szerint a jogszabályt fél éven belül úgysí módosítják. Mi tehát a megoldás?

**15.2 példa** INVERZ SZERZŐDÉS (*Szerződésszám+Jellemző kód*, Jellemző érték)  
JELLEMZŐ (*Jellemző kód*, Jellemző név)

Megoldás: Az idő modellezése kapcsán már volt szó a tulajdonságok általánosításáról ( $\leftarrow$  *Inverzió*). Az ottani megoldást mindig átvehetjük az előre nem látott altípusok illetve váratlanul fellépő új tulajdonságok modellezésére. Ha új szerződésféle lép fel, akkor nem hozunk létre egyelőre (!) új altípust. A meglévő SZERZŐDÉS egyedünket sem bántjuk. Az új szerződés új tulajdonságait a JELLEMZŐ (meta)egyed előfordulásaiként vesszük fel. Az új értékeket pedig az INVERZ SZERZŐDÉS egyedben tároljuk. Ha majd a helyzet normalizálódik és az igények letisztulnak, akkor egy nyugalmas időszakban az implicit - az inverzióba rejtett - altípust átszerkesztjük és abból explicit valódi altípust hozunk létre.

**Kettős tükrözés.** Többféle helyzetben lehet szükség arra, hogy az adatmodellezés „egy jelenség = egy fogalom = egy modellezési egység ...” elvéből némileg engedve egy valós jelenséget többféle módon modellezzünk. Itt két tipikus helyzetet ismertetünk:



**15.3 példa** SZERZŐDÉS (Szerződésszám, ..., Ügyintéző neve)  
SZERZŐDÉS/PARTNER (Szerződésszám+Partner az+Partner jelleg, ...)

Magyarázat: Hasonló jellegű problémák a régi állományok korszerűsítésénél és/vagy több alkalmazás egyesítésénél léphetnek fel. Vannak *régi* szerződéseink, amikben a modellezési elveket nem ismerve az ügyintézőt még a névvel jelöltük meg. Az *újabb* szerződésekben már nem ezt tesszük. Az ügyintézőt is egy partnernek tekintjük és a külön kapcsolóegyedben adjuk meg, hogy ki/kik a szerződés ügyintézője/ügyintézői. Persze az elegáns megoldás az lenne, ha az új adatbázisból az ügyintéző nevét elhagynánk és azt mindenütt a másik egyedben lévő azonosítóval váltanánk ki. Csakhogy ez a konverzió egyrészt igen jelentős munka, másrészt nem mindig valósítható meg egyértelműen, mert több Kovács Rózsa is szerepel a partnerek között és nem tudni, hogy melyik az ügyintéző. Azt csak nagy energiával lehetne kinyomozni. A modellező ekkor kompromisszumot kell, hogy alkalmazzon. Meghagyja a modellben a nevet is, de nem enged újat bevinni.

**15.4 példa** MŰSZAKI OBJEKTUM | MŰSZAKI LÉTESÍTMÉNY, GÉP  
MŰSZAKI STRUKTÚRA  
LÉTESÍTMÉNY/GÉP VISZONY

Magyarázat: Vannak műszaki objektumok, amelyek fogalma a teljes létesítményektől kezdve a gépeken át egészen a tartozékokig terjed. A létesítmény és a gép az objektum két expliciten modellezett altípusa. Az objektum csak a legáltalánosabb adatokat ölelheti fel, mert a gépház (létesítmény) és a szivattyú (gép) rengeteg sajátosságban eltér egymástól. A modellezőnek tükröznie kell a teljes műszaki struktúrát, tehát azt, hogy egy objektum milyen másikkal épül fel (← *Családfa*). Ez a szerkezet csak a tényeket tudja rögzíteni, az összefüggéseket nem nagyon képes minősíteni. Leírható, hogy a létesítményben milyen gépek vannak, azok milyen tartozékokból állnak, de a létesítmény/gép és a gép/tartozék összefüggés nem rendelkezik generizálható tulajdonságokkal.

A gép a használati hely függvényében más kapacitással és másféle módon működik. A kapacitás és a működési mód nem minden objektum viszonyára jellemző, ezért ez a két tulajdonság nem tehető a MŰSZAKI STRUKTÚRA egyedbe. Nincs más megoldás: a tervező kénytelen egy külön LÉTESÍTMÉNY/GÉP viszonyegyet alkalmazni a kérdéses tulajdonságok befogadására. Ez viszont azt jelenti, hogy két valós jelenség összefüggését az egyszeres tükrözési elvet megsebzve kettősen tükrözi. Amivel bizony karbantartási anomáliákat generál saját magának.

Összefoglalás: Nem áll mindig módunkban szép és tiszta adatmodellt alkotni. Sokszor kompromisszumokra kényszerülünk. Ez önmagában véve nem baj. Baj az, ha túlzottan hamar adjuk be a derekunkat egy-egy problémának anélkül, hogy keresnénk az eredendő megoldásokat. Baj az, ha az alku túllépi a józan kompromisszum határát. (Például a saját magunk által megfelelőnek tartott azonosító helyett átvesszük a mások által elképzelt rossz kulcsot.) Mindaddig, ameddig tisztában vagyunk saját modellünk apróbb hibáival, a nem tökéletes megoldások elfogadhatók. Az viszont megengedhetetlen, hogy nemtörődöm módon tervezve, a hatásokat át nem látva alakítunk ki egy rossz modellrészletet, majd utólag kijelentjük, hogy mi csak kompromisszumra törekedtünk...

## 16. TERVEZÉSI ESETTANULMÁNY

### 16.1 A példa kerete

Elérkeztünk az elméleti tudnivalók végéhez. Nem lezártuk, csak befejeztük azokat. Az adatmodellezés elméletéről még lenne mit elmondanunk, de azt nem engedi a tér és az idő. Főleg azért nem, mert még gyakorlati példákkal kell alátámasztani a mondanivalót.

*Ennek a fejezetnek két célja van. Az első részben egy mintapéldát ismertetünk azzal a szándékkal, hogy felkérjük az olvasót a példában rejlő hibák önálló felfedezésére. A második részben az általunk alkalmazott módszer(ek) lépéseinek a bemutatása közben feltárjuk a mintapélda hibáit és összeállítjuk a hibamentes adatbázistervet.*

A szakkönyvekben általában kicsi, egyszerű és mesterséges példákat szoktak felvázolni egy-egy részprobléma megoldásának a szemléltetésére. Ezeknek az a hátránya, hogy nem érzékeltetik a modellezési feladatok összetettségét és a megoldás sorrendjét. Arra pedig nincs mód, hogy bemutassunk egy olyan nagy, bonyolult és valós esetet, amely az összes lehetséges szerkezetet, megoldási módot és - számunkra ez is fontos - modellezési hibát felvonultatná. Egyrészt ilyen nem igen akad a gyakorlatban. Másrészt bár a szerzőnek van a tarsolyában éppen elég valós adatbázisterv, azokat megbízóira való tekintettel itt nem teheti közzé. Harmadrészt erre a terjedelem sem adna módot.

A fentiek miatt kompromisszumra kényszerültünk. Egy olyan feladványt mutatunk be, ami abban az értelemben mesterséges, hogy a tervnek megfelelő adatbázis sohasem készült el a valóságban. Maga a feladat viszont valós volt. Több tervezőcsoportot bíztak meg azzal, hogy készítsék el a rendezvények szervezésének az adatmodelljét. Ne tessék legyinteni! Eláruljuk, hogy a feladat eléggé bonyolult és egyik tervezőcsoportnak sem sikerült tökéletesen megoldania azt. Csak igen nagy figyelem és türelem árán lehet eljutni a közel optimális modellhez.

A megoldáshoz a következő tanácsokkal szolgálunk. Először olvassák el többször is figyelmesen magát a feladatot. Természetesen nem kell az ábrázolási és tárolási fizikai aspektusokkal törődni: azokra nem is szabad gondolni. Az egyszerűség kedvéért nem kell foglalkozni a nem-strukturális korlátokkal sem. Így lényegében az egyed-, a tulajdonság- és a kapcsolattípusoknak és ezek összefüggéseinek a kialakítására kell csak koncentrálni. Másodszor próbálják meg a hibákat csokrokba gyűjteni és kiküszöbölve azokat felvázolni a jónak tartott saját modellt. Csak ezután nézzék meg az általunk adott megoldást.

Egyáltalán nem biztos, hogy az olvasó ugyanarra az eredményre jut, mint a szerző. Ez nem baj. Viszont egy dologtól óvnunk kell a tervezőket: A modellbe nem szabad a saját feltételezések alapján olyan új tényezőket bevezetni, amiket a példa eredetileg nem is tartalmaz. Nem egy új modell kitalálása, hanem a meglévő terv optimalizálása a feladat.

### 16.2 A kiinduló terv

Ez a példa rendezvényről illetve rendezvényekről szól. A rendezvény jellege közömbös. Elképzelhetnek táncházat, kongresszust, kiállítást, tanfolyamot stb. A lényeg az, hogy van egy rendezvény-szervező cég, a Rendezvény Iroda (a továbbiakban: RI), amit ismeretekkel kell ellátni.

Adatbázisba kell szervezni a rendezvénnyel kapcsolatos adatokat. A tervezők az alább leírt modellt állították össze.

Minden rendezvénynek van egy beceneve és egy teljes neve. Minden rendezvényért egy város az elsődleges felelős.

#### 16.1 RENDEZVÉNY

Becenév	Rendezvény neve	Rend. város	Kezdődátum	Záródátum
<b>DFF</b>	Debreceni Folklor Fesztivál	Debrecen	xx.yy.zz	aa.bb.cc
<b>BVK96</b>	Budapesti Világkiállítás 1996	Budapest	qq.vv.ww	dd.ee.ffa

A rendezvényekben érdekelt összes város listáját tartalmazza az alábbi tábla:

#### 16.2 VÁROS

Városkód	Város
<b>BUD</b>	Budapest
<b>DEB</b>	Debrecen

Az időpontok elrendezése érdekében definiáljuk a rendszerben fontos dátumok listáját.

#### 16.3 DÁTUM (Dátum)

A rendezvényeken emberek működnek közre, nagyon eltérő minőségekben. A Telefon opcionális adat. Ha kitöltik, akkor meg kell adni a város körzeti hívószámát is.

#### 16.4 SZEMÉLY

Egyedi az	Név	Cím	Város	Lakcím	Telefon
<b>AB</b>	XXX	Dr	Budapest	C1	6-...
<b>CD</b>	YYY	Id	Szeged	C2	62-...
<b>EF</b>	ZZZ	Prof	Pécs	C3	-
<b>GH</b>	QQQ	Dr	Szeged	C4	62-...

A személyek sokféle módon működnek közre egy rendezvényen. Először is ott van az RI saját stábja. A Rendező Iroda bizottsági rendszerben működik. Vannak a rendezvények technikai, pénzügyi, személyi stb. feltételeinek a biztosításával foglalkozó bizottságok.

#### 16.5 RENDEZŐ EGYSÉG TÍPUS

Rend Egy Típus Kód	RET név
<b>EL</b>	Elnökség
<b>TB</b>	Technikai Bizottság
<b>PB</b>	Program Bizottság

Az egységtípusokon belül vannak konkrét részfeladatokkal megbízott egységek. A részfeladatok egy része standard, más része lehet ad-hoc is. Így például nem minden eseménynél van szükség új beruházásra. A Létrehozás adat az egység életrehívásának az évét jelöli. Az egységek a fenti típusokba soroltak.

#### 16.6 RENDEZŐ EGYSÉG

Rend egység az	Rend egység típus	Rendező egység megnevezés	Létrehozás
<b>TB1</b>	TB	Belső építész	19xx
<b>TB2</b>	TB	Szállítás	19yy
<b>PB1</b>	PB	Beruházás	19zz
<b>PB2</b>	PB	Bér és díjazás	19qq

A RENDEZŐ EGYSÉG-ek feladatait személyek látják el. Mivel bizottságokról van szó, azoknak van elnöke illetve titkára, több tagja, adminisztrátora stb.

#### 16.7 RI FELADAT

RI Feladat kód	RI Feladat megnevezés
<b>TBT</b>	TB titkár
<b>TBG</b>	TB tag
<b>PBE</b>	PB elnök

A személy által ellátott funkciót a 16.8 tábla mutatja. A táblában akkor történik utalás a városra, ha a szerep betöltése helyhez kötődik. Egyébként az adatnak nincs tartalma.

#### 16.8 SZEMÉLY FELADATA

Személy azonosító	RI feladat	RI szerv egység	Rend város kód
<b>AB</b>	<b>TBT</b>	<b>TB1</b>	BUD
<b>CD</b>	<b>TBG</b>	<b>TB1</b>	-
<b>EF</b>	<b>PBT</b>	<b>PB2</b>	PEC
<b>GH</b>	<b>TBG</b>	<b>TB2</b>	-

A rendezvény pénzügyi fedezetéhez támogatókat keresnek. Támogató lehet maga a rendező egység, a rendezvényt tartó város vagy bármilyen más szervezet.

#### 16.9 RENDEZVÉNY TÁMOGATÓ

Rendezvény becenév	Támogató	RI Szerv Egység	Rend Város	Egyéb
<b>DFF</b>	<b>T1</b>	TB1	-	-
<b>DFF</b>	<b>T2</b>	-	DEB	-
<b>BVK96</b>	<b>T1</b>	PB1	-	-
<b>BVK96</b>	<b>T2</b>	-	-	ABCDE

A RENDEZŐ EGYSÉG az IR kvázi-fix stábja. Amikor egy konkrét rendezvényre kerül sor, akkor kétféle ad-hoc bizottságot állítanak fel. Az egyik a rendezvény tartalmáért, a másik annak körülményeiért felel. Ezek a RENDEZVÉNY EGYSÉG-ek.

#### 16.10 RENDEZVÉNY EGYSÉG

Rendezvény becenév	Rendezvény egys kód	Megnevezés
<b>DFF</b>	<b>PB</b>	Programbizottság
<b>DFF</b>	<b>RB</b>	Rendezőbizottság
<b>BVK96</b>	<b>PB</b>	Programbizottság
<b>BVK96</b>	<b>RB</b>	Rendezőbizottság

A rendezvény aktuális szervezeti egységeiben meghatározott feladatokat kell ellátni.

#### 16.11 RENDEZVÉNY FELADAT

Rendezvény feladat típus kód	Feladat megnevezés
<b>PBE</b>	Programbizottsági elnök
<b>BIR</b>	Program bíráló
<b>RBE</b>	Rendezőbizottsági elnök
<b>ZST</b>	Zsűritag

A 16.12 tábla ismerteti, hogy a személyek milyen feladatokat töltenek be a rendezvény két bizottságának valamelyikében. Vannak olyan speciális feladatok is, amik nem kötődnek a szervezeti egységek egyikéhez sem. Ugyanaz a személy több feladatot is elláthat egy rendezvényen. A Kezdődátum mutatja a tisztség betöltésének a kezdetét.

#### 16.12 SZEMÉLY RENDEZVÉNYI FELADATA

Rendezvény becenév	Feladattípus	Személy az	Szerv Egys	Kezdődátum
<b>DFE</b>	<b>PBE</b>	<b>AB</b>	PB	xx.yy.dd
<b>DFE</b>	<b>PBT</b>	<b>CD</b>	PB	dd.yy.dd
<b>DFE</b>	<b>PBT</b>	<b>EF</b>	PB	xx.zz.dd
<b>DFE</b>	<b>XXX</b>	<b>CD</b>	-	xx.qq.dd

A rendezvények programcsoportokból állnak, amiket műsoroknak hívunk. Minden műsornak saját neve van. Két rendezvényen nem szerepel ugyanaz a műsor. (Ugyanolyan nevű lehet, de ugyanolyan kulcsú nem.) A Kezdés és a Zárás a műsor kezdetének és befejezésének az időpontja. Egy rendezvényen lehetnek párhuzamosan futó műsorok is.

#### 16.13 MŰSOR

Rendezvény becenév	Műsorsz	Dátum	Kezdés	Zárás	Műsornév
<b>DFE</b>	<b>1</b>	<b>D1</b>	<b>K1</b>	<b>Z1</b>	Megnyitó
<b>DFE</b>	<b>2</b>	<b>D1</b>	<b>K2</b>	<b>Z2</b>	Felvonulás
<b>DFE</b>	<b>3</b>	<b>D2</b>	<b>K1</b>	<b>Z3</b>	Táncbáz
<b>DFE</b>	<b>4</b>	<b>D2</b>	<b>K3</b>	<b>Z4</b>	Szalagavató

A műsor programokból áll. Ugyanaz a program nem szerepelhet több rendezvényen ill. egy rendezvény több műsorában. Azonos nevű program létezhet, de azonos kulcsú nem.

#### 16.14 PROGRAM

Programsz	Cím
<b>1</b>	XY előadás
<b>2</b>	ZQ tánccsoport
<b>3</b>	YV szavalat

A programokat a műsorokhoz kell rendelni.

#### 16.15 MŰSOR/PROGRAM

<i>Rendezvény becenév</i>	<i>Műsor szám</i>	<i>Program sz</i>
<b>DFF</b>	<b>1</b>	<b>28</b>
<b>DFF</b>	<b>1</b>	<b>68</b>
<b>DFF</b>	<b>2</b>	<b>22</b>
<b>DFF</b>	<b>2</b>	<b>55</b>

Egy programban több személy vehet részt és megfordítva. Minősítésre nincs szükség.

#### 16.16 KÖZREMŰKÖDŐK

<i>Programsz</i>	<i>Személy az</i>
<b>1</b>	<b>AB</b>
<b>2</b>	<b>CD</b>
<b>3</b>	<b>EF</b>

A rendezők nyomon akarják követni az egyes programokat. Azt, hogy kit - pontosabban milyen programot - hívtak meg és mikor; ki fogadta/utasította el a meghívást és mikor; ki jelentkezett önként stb. A tipikus eseményeket a 16.17 tábla sorolja fel.

#### 16.17 PROGRAM-ESEMÉNYTÍPUS

<b>Program esemény típus kód</b>	<b>Esemény megnevezés</b>
<b>MEG</b>	Meghívás
<b>JEL</b>	Saját jelentkezés
<b>ELB</b>	Elbírálás

Természetesen az eseménytípusokat a konkrét programokra kell vonatkoztatni.

#### 16.18 PROGRAM ESEMÉNYEI

<i>Programsz</i>	<i>Program eseménykód</i>	<i>Dátum</i>	<i>Rendezvény becenév</i>
<b>22</b>	<b>JEL</b>	<b>D1</b>	<b>DFF</b>
<b>78</b>	<b>MEG</b>	<b>D1</b>	<b>DFF</b>
<b>22</b>	<b>ELB</b>	<b>D2</b>	<b>DFF</b>
<b>68</b>	<b>ELB</b>	<b>D3</b>	<b>BVK96</b>

Az események egyike az elbírálás. Ez persze nem mindegyik programra vonatkozik. Ám amelyekre igen, annak az esetében az RI mindent tudni akar a bírálatról.

#### 16.19 BÍRÁLAT

<i>Rendezvény becenév</i>	<i>Programsz</i>	<i>Dátum</i>	<i>Személy az</i>	<i>Szerep</i>	<i>Esemény</i>
<b>DFF</b>	<b>22</b>	<b>D1</b>	<b>KL</b>	<b>BIR</b>	<b>ELB</b>
<b>DFF</b>	<b>22</b>	<b>D1</b>	<b>LJ</b>	<b>BIR</b>	<b>ELB</b>
<b>DFF</b>	<b>22</b>	<b>D2</b>	<b>FH</b>	<b>BIR</b>	<b>ELB</b>
<b>DFF</b>	<b>68</b>	<b>D2</b>	<b>KL</b>	<b>BIR</b>	<b>ELB</b>

A rendezvénnyel kapcsolatos személyeket tájékoztatni kell az eseményekről. A részvétel minőségének megfelelően vagy díjat kell kapni tőlük, vagy éppen ellenkezőleg, fizetni kell a számukra. A személyekre vonatkozó események típusait mutatja a 16.20 tábla.

16.20 SZEMÉLY-ESEMÉNYTÍPUS

Személy esemény típus kód	Személy esemény megnevezés
<b>MEG</b>	Meghívás
<b>VIS</b>	Visszajelzés
<b>SZA</b>	Számlázás

Persze mindezeket az eseményféléket is konkrét személyekhez kell kapcsolni.

16.21 SZEMÉLY ESEMÉNYEI

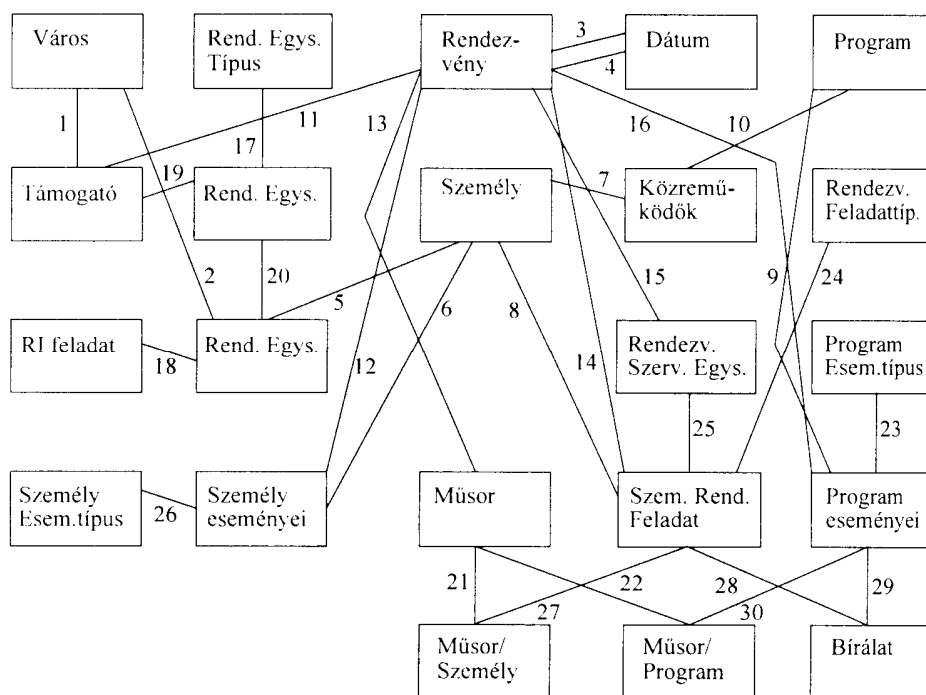
Becenév	Személy eseménykód	Dátum	Egyedi azonosító
<b>DFF</b>	<b>MEG</b>	<b>D1</b>	<b>AB</b>
<b>DFF</b>	<b>VIS</b>	<b>D1</b>	<b>AB</b>
<b>DFF</b>	<b>SZA</b>	<b>D2</b>	<b>AB</b>
<b>BVK96</b>	<b>MEG</b>	<b>D3</b>	<b>CD</b>

Végre elérkeztünk utolsó táblánkhoz. Az RI szeretné ismerni, hogy a rendezvény egyes műsorain kik és milyen minőségben vesznek részt. Vannak, akiknek a szerepe általános és akadnak olyanok is, akiknek a feladata konkrét programhoz kötődik. Ezért a program megadása opcionális. A PRO feladattípus programbeli előadót jelent.

16.22 MŰSOR/SZEMÉLY

Rendezvény becenév	Feladattípus	Személy az	Műsor	Programsz
<b>DFF</b>	<b>PRO</b>	<b>AB</b>	<b>1</b>	1
<b>DFF</b>	<b>YYY</b>	<b>CD</b>	<b>1</b>	-
<b>DFF</b>	<b>ZST</b>	<b>EF</b>	<b>2</b>	22
<b>DFF</b>	<b>XXX</b>	<b>CD</b>	<b>2</b>	-

Az egyedek feltételezett (!) összefüggéseit az alábbi ábra mutatja. Minden kapcsolat fentről-lefelé mutat és 1:N fokú. Ezért nincsenek feltüntetve a nyílak. Az esettanulmány az opcionálításra nem ad eligazítást. A kapcsolattípusok listáját a számoknak megfelelően a 16.23 tábla tartalmazza. (NB.: A rajzot egy az egyben vettük át.)



**16.1 ábra:** Az esettanulmány diagramja

Fel kell hívnunk a figyelmet arra, hogy a példa egy alapvető *ellentmondást* tartalmaz. Amíg erre rá nem jön az olvasó, addig nem fog sikerülnie jó megoldást találni. Segítségként annyit mondunk, hogy figyelje a kulcsokat és a kérdés az, hogy valami egy-e vagy több-e? Saját megoldásunkban mi a „többet” feltételezzük.

### 16.23 KAPCSOLATTÍPUSOK

#### FÖLÉRENDELTEK

- 1 VÁROS
- 2 VÁROS
- 3 DÁTUM
- 4 DÁTUM
- 5 SZEMÉLY
- 6 SZEMÉLY
- 7 SZEMÉLY
- 8 SZEMÉLY
- 9 PROGRAM
- 10 PROGRAM
- 11 RENDEZVÉNY
- 12 RENDEZVÉNY
- 13 RENDEZVÉNY
- 14 RENDEZVÉNY

#### ALÁRENDELTEK

- RENDEZVÉNY TÁMOGATÓ
- SZEMÉLY FELADATA
- RENDEZVÉNY
- RENDEZVÉNY
- SZEMÉLY FELADATA
- SZEMÉLY ESEMÉNYEI
- KÖZREMŰKÖDŐK
- SZEMÉLY RENDEZVÉNYI FELADATA
- KÖZREMŰKÖDŐK
- PROGRAM ESEMÉNY
- RENDEZVÉNY TÁMOGATÓ
- SZEMÉLY ESEMÉNYEI
- MŰSOR
- SZEMÉLY RENDEZVÉNYI FELADATA



15 RENDEZVÉNY	RENDEZVÉNY EGYSÉG
16 RENDEZVÉNY	PROGRAM ESEMÉNYEI
17 RENDEZŐ EGYSÉG TÍPUS	RENDEZŐ EGYSÉG
18 RI FELADAT	SZEMÉLY FELADATA
19 RENDEZŐ EGYSÉG	RENDEZVÉNY TÁMOGATÓ
21 RENDEZŐ EGYSÉG	SZEMÉLY FELADATA
21 MŰSOR	MŰSOR/SZEMÉLY
22 MŰSOR	MŰSOR/PROGRAM
23 PROGRAM ESEMÉNYTÍPUS	PROGRAM ESEMÉNYEI
24 RENDEZVÉNY FELADAT	SZEMÉLY RENDEZVÉNYI FELADATA
25 RENDEZVÉNY EGYSÉG	SZEMÉLY RENDEZVÉNYI FELADATA
26 SZEMÉLY ESEMÉNYTÍPUS	SZEMÉLY/ESEMÉNY
27 SZEMÉLY RENDEZVÉNYI FELADATA	MŰSOR/SZEMÉLY
28 SZEMÉLY RENDEZVÉNYI FELADATA	BÍRÁLAT
29 PROGRAM ESEMÉNY	BÍRÁLAT

Az olvasó most értékelheti az esettanulmány adatbázis-tervét. Tegyen kísérletet a hibák feltárására a saját módszere szerint. Majd próbálja meg felvázolni a helyes modellt.

### 16.3 A megoldás elé

Az adatmodellezés teljes folyamatát egy könyvben nem lehet pontosan utánozni. Azért nem, mert a modellezés a valós helyzet feltárásával - interjúkkal, papírok áttekintésével, kérdőívek kiértékelésével - kezdődik. Mindezeket legfeljebb egy igen terjedelmes szöveges leírással tudtuk volna pótolni. Azonban az ilyen szöveg még kevésbé lenne áttekinthető, mint az amúgy sem egyszerű táblasorozat. Ami nem légből kapott, vagyis nem általunk kitalált mesterséges adatbázis-terv.

1988-ban igen neves szakemberek állították össze a rendezvényszervezési feladatot. Ők úgy vélték, hogy erre a problémára a fenti egyedek és kapcsolatok jelentik a jó megoldást. Olyannyira, hogy azt „teljesen normalizált”-nak titulálták. Tekintettel arra, hogy ez a minősítés igen-igen távol áll a valóságtól, a példa forrását nem jelöljük meg, mert nem a szerzők zavarba hozása a célunk. Mi csak azt akarjuk megmutatni az olvasónak, hogy ha már rendelkezésre áll egy előzetes terv, akkor azt milyen szempontok szerint szükséges felülvizsgálni. Ugyanezt a lépéssorozatot lehet/kell követni akkor is, ha egy meglévő táblahalmazból egy közel optimális adatmodell szerinti adatbázist akarunk építeni.

### 16.4 Ismerkedés a tervvel

Mivel a továbbiakban a saját módszeremet ismertetem, át fogok térni az egyes szám első személyre. Az olvasó megérti, hogy így könnyebb a lényeget elmagyaráznom.

Amikor kézhez kapom egy adatbázis tervét, először *formai* szempontokból vizsgálom azt. Úgy, mint bármilyen dokumentációt, sőt könyvet. Az oldalszámozás hiánya, a nem egységes kép, a

kereshiváthozások elmaradása azonnal gyanút kelt bennem. Ugyanis a külalak nagyon sok mindent elárul. Az adatmodellezés precíziós munka. Nem valószínű, hogy a „slampos” dokumentáció elfogadható adatmodellt takar. Persze esetünkben ezt a szempontot nem lehet mérlegelni, mert nem az eredeti leírást adtam közre. (Mi tagadás, nem volt a dokumentációk gyöngyszeme.)

Második kérdésem az, hogy a leírás tartalomjegyzéke megfelel-e az adatbázistervvel kapcsolatos követelményeknek. Azaz a dokumentáció *struktúráját* mérlegelem. Tekintve, hogy az esettanulmány nem tartalmaz külön tulajdonságlistát és sem a szerepneveket, sem a csoportokat nem emeli ki, azonnal rossz érzések támadnak bennem. Alapos a gyanúm, hogy nem fogalmi modellről, hanem logikai szintű, nem kifinomult tervről van csak szó.

Harmadszor megpróbálom bemérni - most még csak átfogóan - a *tartalom* minőségét. A nem-minősített (pl. Cím, Megnevezés), az elkapkodottan írt (pl. Rend Egy Típus Kód), a hagyományos megoldásokat sejtető (pl. Egyedi az) nevek illetve a *névszabványok* teljes hiánya nem sok jóval kecsegtet. Számos homonimával és szinonimával kell számolnom. Mivel pedig azok elrejtik a teljesség/minimalitás problémáit, további bajok is várhatók.

Negyedszer megvizsgálom, hogy mennyire *bonyolult* az adatmodell. Ezt most magából a diagramból is le lehet mérni. Annak hiányában néhány számra lehet támaszkodni. Ez a terv 22 egyedet és 29 kapcsolatot tartalmaz, tehát méretre kicsi. Nincs olyan egyedtípus, aminek hatnál több tulajdonsága lenne. Viszont igen magas az összetett kulcsok száma. A kulcs/kulcsrészt szereplő tulajdonságoknak a leírókhoz való aránya jól eligazít a modell bonyolultsága tekintetében. Ha egy modellben sok az elemi kulcsú és sok leíró tartalmazó egyed, akkor az egyszerű. Ha sok az összetett kulcs és kevés a leíró tulajdonság, akkor a modell bonyolult. Ez a helyzet a mi esetünkben is.

Ötödször kíváncsi vagyok a modell *kifinomultságára*. Ezt a speciális szerkezetekből (*Sablonok*  $\Leftarrow$ ), az altípusokból, a csoportok és szerepnevek használatából, az alkalmazott sajátos minősítésekből lehet megítélni. Az esettanulmányban nincsen semmi finomság. Ha szabad ilyen kifejezést használni: az adatbázisterv póriasan hétköznapi. Ez pedig a rossz, hagyományos megoldások erőltetését vetíti előre.

## 16.5 Az egyértelműség elemzése

Mivel olyan modellt nem lehet normalizálni, amely nem egyértelmű, legelső feladat az egyértelműség megteremtése. Bár mintapéldánk mérete nem nagy, a szerkezet mégis elég bonyolult. Emiatt feltétlenül számítógépes *adatszótárt* használok az elemzéshez. Sorra veszem a táblaképeket és szótárba viszem az egyed-tulajdonság-viszonyokat. Eközben fokozatosan bővíttem a külön egyed- és tulajdonságlistát. Tehát amikor elkönyvelem, hogy a SZEMÉLY egyedhez kapcsolódik a Név tulajdonság, akkor e viszony mellett rögzítem az egyedlistán a SZEMÉLY, a tulajdonságlistán a Név tételt.

Persze ezt nem pontosan így teszem. Rögzítés közben azonnal *egységes módon* írom át a neveket. Általában kerülöm a *rövidítéseket*. Pl. a „Rend Egy Típus Kód” nevet én így írom: „Rendező egység típuskód”. (Én is szoktam rövidítéssel élni. Például „X az” illetve „Y cs” jelöli néha az X egyed azonosítóját illetve az Y csoportot. Ha viszont rövidíték, akkor azt minden kulcsnál/csoportnál megteszem.) A *túlminősített* neveket egyszerűsíttem. Pl. a „Rend város kód”-ban a jelző nem korlátoz, ezért feleslegesen minősít. Bármelyik város lehet rendező. Ezért itt az egyszerű Városkód nevet használom. Viszont az *alulminősített* neveket kibővíttem. Így lesz a program

Címéből Programcím. A *rosszul beszélő* neveket átírom. Tehát a személy zavaró Cím adatából Titulus lesz.

Ezzel a „keresztelővel” eleve kiszűröm a homonimák és szinonimák egy részét anélkül, hogy külön kutatom kellene utánuk. A kiinduló modellben a Cím homonima volt - már nem az. A „Rend város kód” és a „Városkód” technikai szinonima volt - amit máris felszámoltam. A **homonimákat** az ember az alulminősítésből veheti észre. Ekkor jön az átnevezés, amit általában minden egyedben célszerű végrehajtani. Nem jó az, ha a Címből az egyik egyedben Programcím lesz, a másikban Cím marad. Azonnal szemet szúr jó pár **szinonima** is. Az eltérő írásmódból fakadó technikaiakat (pl. „RI szerv egység” és „RI Szerv Egység”) könnyű megtalálni. Azonban a teljesen eltérőeket (pl. a személy kulcsa az „Egyedi az” és a „Személy az”) nehezebb felfedezni.

A fentiek után a **kétszeres tételek** száma már lecsökken. Általában ritkán követik el azt a durva hibát, hogy kétszeres egyednevet alkalmaznak. Viszont az esettanulmányban két DÁTUM - RENDEZVÉNY nevű kapcsolat található. Persze a kapcsolat kettős, de nevük ennek ellenére nem lehet azonos. A legtöbb baj a tulajdonságtípusokkal van éppen azért, mert a tervek nem tartalmaznak tulajdonságlistát és így a tulajdonságokat a tervező csak egyedekhez kötötten látja. Az egyed- és a kapcsolatnév sohasem lehet többszörös. Viszont a tulajdonságoknál ez a kapcsolati szerep miatt előfordulhat (*Kapcsolótulajdonság*  $\Leftarrow$ ).

Most még a **szerepnevekre** kell kitérnem. Azért, mert azok elhagyása illetve helytelen használata szintén egyértelműségi hiányra vezethet, ami pedig további bajokat jelezhet. Az esettanulmány nem egyértelmű, mert a RENDEZVÉNY Kezdődátum és Záródátum tételéről nem jelenti ki, hogy a Dátum szerepnevei. Minek is? - mondhatná bárki, hiszen ez evidens. Valóban? Ha annyira világos, akkor miért maradt ki a kapcsolatok listájából a DÁTUM - SZEMÉLY RENDEZVÉNYI FELADATA kapcsolat, noha az utóbbi egyedben is szerepel Kezdődátum nevű adat (ami mellel így homonima)?

A szótár feltöltése után két kis segédletet alkalmazok. Először is készítek egy ABC sorrendű listát a tulajdonságokról. Ebből azonnal láthatom (ha addig nem tettem volna), hogy a „Személy az” és a „Személy azonosító” tételek slamposságból fakadó technikai szinonimák. Azért láthatom, mert ezen a listán a két név egymás mellett szerepel, míg a táblaleírásokban mindig távol volt egymástól. Egy kis programmal azt is igen könnyen megállapíthatom, hogy az egyik név része-e a másiknak. Így felfedezhetem, hogy a „Rendezvény becenév” és a „Becenév” szinonimák. (Ez az ún. KWIC-KWOC elemzés. A „Key-Word In Context” és „Key-Word Out of Context” lényegét itt nem írhatom le.)

## 16.6 Azonosság- és azonosítóelemzés

Az egyértelműség elemzésével párhuzamosan - illetve részben már a szótár feltöltése közben is - pontosan meghatározom a szerepneveket és a csoportokat. Mivel ennek a lépésnek nincs külön technikája, sok mondanivalóm nincs róla. Legfeljebb annyi, hogy a feladatot az azonosítóelemzés előtt el kell végezni.

Az **azonosságelemzés** a modell minden tényezőjére vonatkozik. Természetesen itt már nem a durva, szembeötlő kétszerességről van szó. Korábban már beszéltünk a szerkezeti szinonimákról. Ha a tervben előfordul az X {A+B+C} és az Y {A+B+C} csoport, akkor a két tulajdonságstruktúra azonos. Ez a jelenség kapcsolatoknál is felléphet. Az X kapcsolat is az A és a B egyedek köti össze, meg az Y viszony is. Ekkor ha a két egyednek csak egy közös kapcsolótulajdonsága van, a viszonymeghatározás redundáns, mert egy kapcsoló csak egy kapcsolatnak adhat alapot. Az is

megtörténik, hogy a tervező két X (A, C, D) és Y (B, C, D) egyedtípust kreál. A két egyed minden leíró tulajdonságában megegyezik. Ekkor a tervező vagy nem alkalmazta a generalizáció műveletét, vagy - tévedésből - két azonos egyedtípust alkotott. (N.B.: Saját elemző eszközöm *átfedési százalékot* is számít. Kimutatja, hogy két egyednek N százalékban azonosak a tulajdonságai. Ha az N magas, akkor a tervező elgondolkozhat az általánosításon.)

Az *azonosítóelemzés* összetett feladat. Először is vizsgálni kell, hogy nem sérti-e a terv az *azonosítókra vonatkozó korlátokat*. Ha több egyednek ugyanaz a tulajdonság a kulcsa, vagy megfordítva, egy egyednek több kulcsot jelölnek ki nem ismervé az alternáló kulcs intézményét, akkor a hibát ki kell küszöbölni. Másodszor - főleg több kulcsjelölt esetében - elemezni kell az *elsődleges kulcs megfelelőségét*. Tehát azt, hogy a több lehetőség közül valóban a célszerűt választotta-e ki a tervező. Még az is előfordulhat, hogy az azonosító értéke nem egyedi, amennyiben az lehet több tételnél ismeretlen is. Ki kell zárni azt, hogy ilyenkor a kulcs „kamu” értékeket vehessen fel. *Összetett kulcsoknál* vizsgálandó, hogy a részek valóban M:N-es viszonyokban állnak egymással; hogy nincs-e bennük felesleges elem (túldefiníálás); hogy a tagok valóban elegendők-e az azonosításhoz (aluldefiníálás). A *speciális egyedeknél* arra is figyelni kell, hogy a kulcs szerepnev-e (*egyedaltípus*) vagy a fölérendelt kulcsának a szerepnevei alkotják-e (*család*) stb.

Mintapéldánkban a MŰSOR kulcsa (Rendezvény becenév, Műsorsz, Dátum, Kezds, Zárs) katasztrofális. Egy rendezvényen több műsor szerepelhet, de ugyanaz a műsor nem fordulhat elő többször ugyanazon a rendezvényen. Az pedig evidens, hogy egy műsor nem kezdődhet és végződhet különböző időpontokban. Ezért az egyed kulcsa túlhatározott. Az azonosításra elegendő a Rendezvény becenév és a Műsor sorszám. Teljesen felesleges a behatároláshoz a Dátum, a Kezds és a Zárs. Ezek valójában csak leíró tulajdonságok.

Most térünk ki a példa *alapvető ellentmondására*, mert az az azonosítással kapcsolatos. A PROGRAM egyed kapcsán kijelentettük, hogy minden program kulcsa egyedi. Ha ez igaz, akkor a Programszám funkcionálisan meghatározza a Műsor sorszámot, ill. magát a Rendezvény becenév tételt is. Hiszen ugyanaz a program nem szerepelhet több műsorban, sőt rendezvényen sem. Ha ez igaz, akkor rossz a MŰSOR/PROGRAM egyed azonosítója (Rendezvény becenév, Műsor szám, Program sz), hiszen azt sejteti, hogy a program és a műsor funkcionálisan független.

Ehhez képest a Rendezvény becenév és a Programszám tétel a mintapélda számos egyedében mindenféle variációban együtt is szerepel. Ha a Programszám valóban egyedi, akkor meghatározza a Rendezvény becenév tételt, tehát a két adat csak egyetlen egyedben tűnhetne fel együttesen: magában a PROGRAM-ban. A turpisságot „könnyű” felfedezni: a MŰSOR/PROGRAM azonosítója *belső kulcstörő függést* mutat.

Itt tehát egy alapvető döntést kell hozni. A tervező nem tudta elhatározni még azt sem, hogy mi legyen a modell *alapvető tárgya*, hatóköre. Eredetileg egyetlen egy rendezvény tervezéséből indult ki, amely esetben a program egyedisége nem kérdéses. Majd hirtelen átváltott a több rendezvény gondolatára. Ekkor viszont felmerül a kérdés, hogy ugyanaz a program szerepelhet-e több rendezvényen is. Ha nem, akkor a Rendezvény becenév függ a Programszámtól. Ha igen, akkor a két adat független. Mivel mintapéldánkban több egyed inkább az utóbbi helyzetre enged következtetni (és mellel az is a természetesebb), úgy döntöttem, hogy a megoldásnál a több rendezvény hatókört tekintem alapnak.

Az esettanulmány átmenetileg javított verzióját alább adom közre. A kiinduló modellen annyit mindent kellett megváltoztatnom, hogy nem haladhatok tovább az eddigi változat rögzítése nélkül. Magukat az eredeti táblázatokat nem kívánom megismételni, ezért az alábbiakban csak a javított listaképeket mutatom. A csoportokat {}, a szerepneveket [ ] jelek között részletezem. Az összetett kulcsokat az egyedek nevéhez tett „cs” - csoport - betűvel jelölöm.

RENDEZVÉNY (**Rendezvény becenév**, Rendezvénynév, Városkód, Kezdődátum, Záródátum) -  
Dátum [Kezdődátum, Záródátum]  
VÁROS (**Városkód**, Városnév)  
DÁTUM (**Dátum**)  
SZEMÉLY (**Személy azonosító**, Személynév, Titulus, Városnév, Lakcím, Telefon)  
RENDEZŐ EGYSÉG TÍPUS (**Rendező egység típuskód**, Rendező egység típusnév)  
RENDEZŐ EGYSÉG (**Rendező egység azonosító**, Rendező egység típuskód, Rendező egység megnevezés, Létrehozás éve)  
RENDEZŐ EGYSÉG FELADAT (**Rendező egység feladatkód**, Rendező egység feladatnév)  
SZEMÉLY RENDEZŐ FELADAT (**Személy rendező feladat cs**, Városkód)  
Személy rendező feladat cs {Személy azonosító + Rendező egység feladatkód + Rendező egység azonosító}  
RENDEZVÉNY TÁMOGATÓ (**Rendezvény támogató cs**, Rendező egység azonosító, Városkód, Egyéb szervezatkód)  
Rendezvény támogató cs {Rendezvény becenév + Támogató sorszám}  
RENDEZVÉNY EGYSÉG TÍPUS (**Rendezvény egységkód**, Rendezvény egység megnevezés)  
RENDEZVÉNY EGYSÉG (**Rendezvény egység cs**, Rendezvény egység megnevezés)  
Rendezvény egység cs {Rendezvény becenév + Rendezvény egységkód}  
RENDEZVÉNY FELADAT (**Rendezvény feladat típuskód**, Rendezvény feladat megnevezés)  
SZEMÉLY RENDEZVÉNY FELADAT (**Személy rendezvény feladat cs**, Rendezvény egységkód, Dátum)  
Személy rendezvény feladat cs {Rendezvény becenév + Rendezvény feladat típuskód + Személy azonosító}  
MŰSOR (**Műsor cs**, Dátum, Kezdő időpont, Záró időpont, Műsornév)  
Műsor cs {Rendezvény becenév + Műsor sorszám}  
PROGRAM (**Programszám**, Programcím)  
MŰSOR/PROGRAM (**Műsor/program cs**)  
Műsor/program cs {Rendezvény becenév + Műsor sorszám + Programszám}  
KÖZREMŰKÖDŐ (**Közreműködő cs**)  
Közreműködő cs {Programszám + Személy azonosító}  
PROGRAM ESEMÉNYTÍPUS (**Program esemény típuskód**, Program esemény típus megnevezés)  
PROGRAM ESEMÉNY (**Program esemény cs**, Rendezvény becenév)  
Program esemény cs {Programszám + Program esemény típuskód + Dátum}  
BÍRÁLAT (**Bírálat cs**, Rendezvény feladat típuskód, Program esemény típuskód)  
Bírálat cs {Rendezvény becenév + Programszám + Dátum + Személy azonosító}  
SZEMÉLY ESEMÉNYTÍPUS (**Személy esemény típuskód**, Személy esemény típus megnevezés)  
SZEMÉLY ESEMÉNY (**Személy esemény cs**)  
Személy esemény cs {Rendezvény becenév + Személy esemény típuskód + Dátum + Személy azonosító}  
MŰSOR/SZEMÉLY (**Műsor/személy cs**, Programszám)  
Műsor/személy cs {Rendezvény becenév + Rendezvény feladat típuskód + Személy azonosító + Műsor sorszám}

Tudom, hogy a fenti lista összeállítása mennyire nehéz feladat. Nekem is néhány órámba telt. Ehhez több megjegyzésem van. Egyrészt nem kínlódtam volna ennyit, ha az eredeti terv egyértelmű lett volna. Másrészt a nevek hosszúak, viszont magyarázóak. Ezeken lehet még faragni: eddig szándékosan nem tettem. Mert például a REFETIKOD névből az olvasó nem tudhatná, hogy a Rendezvény feladat típuskódra gondolok. Az adatmodellnek beszélőnek kell

lennie, ami sok nyügget jár. Harmadrészt ez a modell már majdnem egyértelmű - és ennyit megért az eddigi kínlódás.

## 16.7 Egyéb elemzések

Nem állhatunk meg egy féligkész elemzésnél. Az adatmodellezéssel úgy van, mint a könyveléssel. Pár forintos eltérés utáni kutatás tárja fel, hogy milliókról van szó. Egyetlen adatmodellezési hiba is számos mást szülhet. Ezért folytatnom kell a nyomozást. Először a szemmel is látható „apró” hibák felé fordulok. A mintasorokkal szemléltetett táblaképek azért hasznosak, mert segítségükkel feltárhatjuk a modell rejtett redundanciáit. Ebben a pontban háromféle titkos fizikai adatátfedésre irányítom a figyelmet.

A 16.6 táblában a „Rend egység típus” tétel értéke mindig megegyezik a „Rend egység az” tulajdonság értékének az első két karakterével. Itt az *implicit csoportból* következő fizikai redundancia tipikus esetével állunk szemben. A „Rend egység az” olyan csoport, ami impliciten tartalmazza a „Rend egység típus”-t. A kézenfekvő megoldás az, hogy az azonosítót explicit csoportként adjuk meg. Ettől persze a modell bonyolultabb lesz, viszont elkerüljük a karbantartási anomáliákat.

A másik probléma a *generikus tulajdonságokkal* kapcsolatos. A 16.19 táblában a Szerep és az Esemény olyan tulajdonságok, amelyeknek az értéke mindig ugyanaz. Akkor pedig nyilvánvalóan feleslegesen terhelik az egyedet, hiszen annak lényege a bíráló, amiben a szerep mindig bíráló, az esemény mindig elbírálás. Az ilyen tételeket ki kell irtani. Ennek során nagy óvatossággal kell eljárni. Meg kell nézni, hogy valóban mindig csak egyféle-e az érték. Ugyanis a minták félrevezetőek is lehetnek. Vessünk csak egy pillantást a 16.12 táblára. Abban a „Szerv Egys” tétel mindig csak „PB” vagy üres értéket vesz fel. Ám ez megtévesztő: a tétel nem generikus tulajdonság, hiszen a tábla előtti szöveg azt sejteti, hogy a kétféle bizottságban betöltött feladatokról van szó. Egyszerűen a tervező ebben az esetben rosszul szemléltette értéksorokkal a táblaképet.

Most jutottunk el az alapvető *normálformák* kérdéséhez. A helytelen normálalak csak akkor fedezhető fel automatával, ha a függő tulajdonság nyílt logikai redundanciát okoz. Ha a Vevőnév a VEVŐ és a RENDELÉS egyedben is szerepel a meghatározó Vevőkód tétellel együtt, akkor a tranzitív függést bármilyen automata könnyen feltárja. Ám ha a Vevőnév csak a RENDELÉS egyedben található a Vevőkóddal együtt, úgy a tranzitivitás automatikus feltárására nincs mód. Ekkor „kézzel” kell normalizálni. Mivel a normálalak hibája mindig fizikai redundanciával jár, feltárásában ismét csak a mintasorok segítenek. A 16.10 táblában feltűnő a Megnevezés redundanciája. Ez a tétel részlegesen függ a kulcstól. A SZERVEZÉSI EGYSÉG nincs 2NF alakban. Ezért abból ki kell emelni ezt a tulajdonságot egy új egyedbe.

Az alábbiakban közreadom a példa újabb időleges megoldási változatát. A listában immár a régebben megszokott dőltbetűs szedettel mutatom a kapcsoló tulajdonságokat. Az egyedek közül a lista végére teszem azokat, amik csak kódokként fognak megjelenni.

RENDEZVÉNY (**Rendezvény becenév**, Rendezvénynév, Városkód, Kezdődátum, Záródátum)

Dátum [Kezdődátum, Záródátum]

VÁROS (**Városkód**, Városnév)

DÁTUM (**Dátum**)

SZEMÉLY (**Személy azonosító**, Személynév, Titulus, Városkód, Lakcím, Telefon)

RENDEZŐ EGYSÉG (**Rendező egység cs**, Rendező egység típuskód, Rendező egység megnevezés, Létrehozás éve)  
 Rendező egység cs {*Rendező egység típuskód* + Rendező egység jel}  
 SZEMÉLY RENDEZŐ FELADAT (**Személy rendező feladat cs**, Városkód)  
 Személy rendező feladat cs {*Személy azonosító* + *Rendező egység feladatkód* + *Rendező egység cs*}  
 RENDEZVÉNY TÁMOGATÓ (**Rendezvény támogató cs**, *Rendező egység cs*, Városkód, Egyéb szervezatkód)  
 Rendezvény támogató cs {*Rendezvény becenév* + Támogató sorszám}  
 RENDEZVÉNY EGYSÉG (**Rendezvény egység cs**, Rendezvény egység megnevezés)  
 Rendezvény egység cs {*Rendezvény becenév* + *Rendezvény egységkód*}  
 SZEMÉLY RENDEZVÉNY FELADAT (**Személy rendezvény feladat cs**, *Rendezvény egységkód*, Dátum)  
 Személy rendezvény feladat cs {*Rendezvény becenév* + *Rendezvény feladat típuskód* + *Személy azonosító*}  
 MŰSOR (**Műsor cs**, Dátum, Kezdő időpont, Záró időpont, Műsornév)  
 Műsor cs {*Rendezvény becenév* + Műsor sorszám}  
 PROGRAM (**Programszám**, Programcím)  
 MŰSOR/PROGRAM (**Műsor/program cs**)  
 Műsor/program cs {*Rendezvény becenév* + Műsor sorszám + *Programszám*}  
 KÖZREMŰKÖDŐ (**Közreműködő cs**)  
 Közreműködő cs {*Programszám* + *Személy azonosító*}  
 PROGRAM ESEMÉNY (**Program esemény cs**, *Rendezvény becenév*)  
 Program esemény cs {*Programszám* + *Program esemény típuskód* + Dátum}  
 BÍRÁLAT (**Bírálat cs**)  
 Bírálat cs {*Rendezvény becenév* + *Programszám* + Dátum + *Személy azonosító*}  
 SZEMÉLY ESEMÉNY (**Személy esemény cs**)  
 Személy esemény cs {*Rendezvény becenév* + *Személy esemény típuskód* + Dátum + *Személy azonosító*}  
 MŰSOR/SZEMÉLY (**Műsor/személy cs**, Programszám)  
 Műsor/személy cs {*Rendezvény becenév* + *Rendezvény feladat típuskód* + *Személy azonosító* + *Műsor sorszám*}  
 RENDEZŐ EGYSÉG TÍPUS (**Rendező egység típuskód**, Rendező egység típusnév)  
 RENDEZŐ EGYSÉG FELADAT (**Rendező egység feladatkód**, Rendező egység feladtnév)  
 RENDEZVÉNY EGYSÉG TÍPUS (**Rendezvény egységkód**, Rendezvény egység megnevezés)  
 RENDEZVÉNY FELADAT (**Rendezvény feladat típuskód**, Rendezvény feladat megnevezés)  
 PROGRAM ESEMÉNYTÍPUS (**Program esemény típuskód**, Program esemény típus megnevezés)  
 SZEMÉLY ESEMÉNYTÍPUS (**Személy esemény típuskód**, Személy esemény típus megnevezés)

## 16.8 Kapcsolatelemezés

Az egyedek belső szerkezetének az elemzése után a *külső szerkezettel* is foglalkoznunk kell. Vizsgáljuk meg a mintapélda által sugallt kapcsolatokat (lásd 16.23 tábla)! Egészen otromba hiba, hogy a Bachman-diagramon 30, a kapcsolatlistán viszont csak 29 viszony van feltüntetve.

Az egyedek számából és azonosítóik összetettségéből pedig az is sejthető, hogy a tervező elfeledkezett néhány kapcsolat megadásáról. A kapcsolatok teljességének és minimalitásának az elemzésében sokat segíthet az, hogy az adatmodellben a viszonyok kettősen meghatározottak: expliciten adottak a *kapcsolatlistán*, de impliciten is kijelöltek a *kapcsolótulajdonságok* által. E két meghatározás egybevágóságát kell vizsgálni. Ha egy kapcsolatot nem alapoz meg kapcsolóadat, akkor az felesleges. Ha létezik kapcsolóadat, de nincs explicit kapcsolatnév, akkor a kapcsolat hiányzik.

Kapcsolatelemzésre csak akkor kerülhet sor, ha a kapcsolótulajdonság egyértelmű. Ha a kapcsoló *homonima*, akkor a viszony csak látszólagos. Ha *szinonima*, akkor pedig rejtett. Szintén nem látható a viszony akkor, ha a tervező rosszul alkalmazza a *szerepneveket*. Végül gondolnunk kell arra is, hogy a modell formálisan jó, de mégis hiányos. Például a SZEMÉLY egyedben a Városnév tulajdonság szerepel. Ezért el kell medítnunk azon, hogy vajon nem szükséges-e a személyeket a városokhoz kapcsolni. Ha igen, akkor ezt az adatot ki kell cserélni a Városkódra. (Egy trükk: Ez a lehetőség könnyen felfedezhető egy nagy modellben is. A Városnév a VÁROS és a SZEMÉLY egyedben nyílt logikai átfedést okoz anélkül, hogy ez normálforma hibára vezetne. Az ilyen átfedés arra utalhat, hogy a hibamentes modell mégsem tökéletes.)

Most térjünk át a gyakorlatra, mintapéldánk valós problémáira. A tervező a DÁTUM egyedet csak kétszer határozta meg fölérendeltként, holott mindenki tisztán láthatja, hogy a Dátum adat több egyedben szerepel. A VÁROS is mostoha sorsra jutott: nem definiálták a SZEMÉLY és a RENDEZVÉNY felé való kapcsolatát. A SZEMÉLY RENDEZVÉNY FELADAT és a PROGRAM ESEMÉNY egyedeknek a BÍRÁLAT felé való kapcsolatait, a PROGRAM ESEMÉNY és a MŰSOR/PROGRAM viszonyát nem indokolják az egyedek tulajdonságai. Például a PROGRAM ESEMÉNY nem tartalmazza a Műsor sorszámát. Viszont e tévesen megadott viszonyok miatt a BÍRÁLAT kicsit a levegőben lóg: annak a RENDEZVÉNY-nyel, SZEMÉLY-lyel és PROGRAM-mal való viszonya hiányzik. Mivel a PROGRAM ESEMÉNY nem kötődik közvetlenül a MŰSOR/PROGRAM egyedhez, az utóbbinak a PROGRAM-mal való viszonya lenne szükséges.

## 16.24 KONSZOLIDÁLT KAPCSOLATTÍPUSOK

FÖLÉRENDELTEK	ALÁRENDELTEK
1 VÁROS	RENDEZVÉNY TÁMOGATÓ
2 VÁROS	SZEMÉLY FELADATA
3 DÁTUM	RENDEZVÉNY
4 DÁTUM	RENDEZVÉNY
5 SZEMÉLY	SZEMÉLY FELADATA
6 SZEMÉLY	SZEMÉLY ESEMÉNYEI
7 SZEMÉLY	KÖZREMŰKÖDŐK
8 SZEMÉLY	SZEMÉLY RENDEZVÉNYI FELADATA
9 PROGRAM	KÖZREMŰKÖDŐK
10 PROGRAM	PROGRAM ESEMÉNY
11 RENDEZVÉNY	RENDEZVÉNY TÁMOGATÓ
12 RENDEZVÉNY	SZEMÉLY ESEMÉNYEI
13 RENDEZVÉNY	MŰSOR
14 RENDEZVÉNY	SZEMÉLY RENDEZVÉNYI FELADATA
15 RENDEZVÉNY	RENDEZVÉNY EGYSÉG
16 RENDEZVÉNY	PROGRAM ESEMÉNYEI
17 RENDEZŐ EGYSÉG TÍPUS	RENDEZŐ EGYSÉG



18 RI FELADAT	SZEMÉLY FELADATA
19 RENDEZŐ EGYSÉG	RENDEZVÉNY TÁMOGATÓ
21 RENDEZŐ EGYSÉG	SZEMÉLY FELADATA
21 MŰSOR	MŰSOR/SZEMÉLY
22 MŰSOR	MŰSOR/PROGRAM
23 PROGRAM ESEMÉNYTÍPUS	PROGRAM ESEMÉNYEI
24 RENDEZVÉNY FELADAT	SZEMÉLY RENDEZVÉNYI FELADATA
25 RENDEZVÉNY EGYSÉG	SZEMÉLY RENDEZVÉNYI FELADATA
26 SZEMÉLY ESEMÉNYTÍPUS	SZEMÉLY/ESEMÉNY
27 SZEMÉLY RENDEZVÉNYI FELADATA	MŰSOR/SZEMÉLY
* SZEMÉLY RENDEZVÉNYI FELADATA	BÍRÁLAT
* PROGRAM ESEMÉNY	BÍRÁLAT
30 DÁTUM	SZEMÉLY RENDEZVÉNY FELADAT
31 DÁTUM	MŰSOR
32 DÁTUM	PROGRAM ESEMÉNY
33 DÁTUM	BÍRÁLAT
34 DÁTUM	SZEMÉLY ESEMÉNY
35 VÁROS	RENDEZVÉNY
36 VÁROS	SZEMÉLY
37 PROGRAM	MŰSOR/PROGRAM
38 PROGRAM	BÍRÁLAT
39 RENDEZVÉNY	BÍRÁLAT
40 SZEMÉLY	BÍRÁLAT

A csillaggal jelölt tételek nem-valóságok, megszűntek. Az azok utániak a tulajdonságok által ténylegesen alátámasztott új - az eredeti tervből hiányzó - kapcsolatokat mutatják. Az új modellt nem rajzolom fel: azt már bárki megteheti a korábbi ábra alapján. Már csak annyit kell elárulnom, hogy miért „időleges” ez a megoldás is.

Icicipi redundanciák még mindig maradtak a modellben. A SZEMÉLY egyed Telefon tételének első pár karaktere a településre utal. Nyilvánvaló fizikai redundanciát okoz. Az egyes feladatok kódjai a rendező illetve rendezvény egységek kódjainak a kiegészítései. Ez is redundancia és a módosításoknál bajokat eredményezhet, de talán elviselhetőeket. Csak arra akarom felhívni a figyelmet, hogy a kódba épített kód problémákat okozhat. Ha valakinek nem tetszik a Rendezőbizottság megjelölés és ahelyett Szervezőbizottság nevet akar - ez bizony gyakran megesik a mindennapi életben -, akkor a korábbi „RB” kódokat át kellene írni minden feladatot jelölő rokon tulajdonságban is „SB”-re. Az egymásra épített beszélőkódok ugyanis nem jól tűrik a változtatásokat...

A normálformák szempontjából pedig modellünk - rossz. Remélhetőleg feltűnt, hogy a MŰSOR/PROGRAM {Rendezvény becenév + Műsor sorszám + Programszám} összetett azonosítója ellentmondásos. Ugyanis a Rendezvény becenév és a Programszám együttese még mindig belső kulcstörő függést okoz. Miután ugyanaz a program egy rendezvény több műsorában nem fordulhat elő, a Rendezvény becenév+Programszám meghatározza a Műsor sorszámat. Vagy élünk ezzel a helyzettel, ami esetünkben nem okoz karbantartási gondot, vagy újra kell gondolnunk a rendezvény, a műsor és a program viszonyát.

## 17. KÉT PÉLDA

Ebben a fejezetben két nagyon különböző jellegű esetet mutatunk be. Ezzel nem csak az a célunk, hogy eltérő tartalmú modellekkel ismertessük meg az olvasót. A két példa didaktikai célokat is szolgál. Vagyis mindegyikben az eddigi elméleti mondanivalót újabb hasznos gyakorlati tudnivalókkal egészítjük ki.

Az első példa (17.1 *Készletezés*) mesterséges. A jó megoldásban felvonultatjuk a modell speciális szerkezeteit az altípustól kezdve a visszamutatáson át a családfáig. Az olvasónak érdemes elgondolkodnia a számított tulajdonságok, a csoportok és a szerepnevek korrekt modellezésén. Emellett meg kell birkóznia kétféle igen ravasz általánosítási problémával is. Eláruljuk, hogy majd a termék/cikk és a raktár/üzlet párosok okoznak nehézségeket.

A második példa (17.2 *Betegségbiztosítás*) kvázi az „állatorvosi ló” szerepét tölti be. Megmutatja, hogy milyen malőrökre vezethet az, ha nem a valós jelenségeket, hanem a papírokat akarja valaki modellezni. Az olvasó három igen lényeges dologra kell, hogy figyeljen a kérdéseinkre adott válaszoknál. Egyrészt az azonosítókra, amiknek elvileg örökérvényűeknek kellene lenniük. Másrészt azokra a hálós viszonyokra, amiket az eredeti megoldásban hierarchikusakra redukáltak. Harmadrészt a dinamikára, mert a kiinduló modell nem vette figyelembe az idő dimenzióját.

### 17.1 Készletezés

Feladat:Eddigi szokásunkkal ellentétben most azonnal a dolgok közepébe vágunk. Az olvasó szíveskedjék áttanulmányozni a mintapéldát és próbálja megállapítani az abban rejlő hibákat. Majd a következő pontban elmondjuk saját elemzési eljárásunkat is. Az utolsó sorban a „|” jelölés azt fejezi ki, hogy a tételek a tényező szerepnevei.

#### 17.1 példa

CIKK	( <b>Cikkszám</b> , Megnevezés, Egységár, Súly, Árucsoport jel, <i>Termékkód</i> , <i>Raktárkód</i> )
NORMÁLIA	( <b>Normáliakód</b> , Név, Egységár, Súly, Árucsoport jel, <i>Raktárkód</i> )
ÁRUCSOPORT	( <b>Árucsoport kód</b> , Megnevezés)
TERMÉK	( <b>Termékkód</b> , <i>Befogadó cikkszám</i> , <i>Beépülő cikkszám</i> , Darab)
RAKTÁR	( <b>Raktárkód</b> , Raktárcím, Raktárnév, <i>Főraktárkód</i> )
FŐRAKTÁR	( <b>Főraktárkód</b> , Raktárcím, Raktárnév)
ÜZLET	( <b>Üzletkód</b> , Boltcím, Boltnev, X...)
KÉSZLET-1	( <b>Cikk kód+Raktárkód+Dátum</b> , Mennyiségi egység, Mennyiség, Érték)
KÉSZLET-2	( <b>Normáliakód+Raktárkód+Dátum</b> , Érték, Mennyiségi egység, Mennyiség)
BEVÉTEL	( <b>Dispozíciószám+Cikkszám</b> , Bevétel dátum, Érték, Egységár, Mennyiség, Raktárkód, Készlet)
KIADÁS	( <b>Dispozíciószám+Cikkszám</b> , Kiadás dátum, Érték, Egységár, Mennyiség, Raktárkód, Készlet)
Cikkszám	Normáliakód, Befogadó cikkszám, Beépülő cikkszám

Magyarázat: A cég számítógépes berendezéseket és alkatrészeket forgalmaz. A cikkeket árucsoportokba sorolják. A csoportokat csoportkóddal azonosítják és megnevezésükkel írják le. A cikkeket cikkszám, megnevezés, egységár és súly jellemzi. A cikkszám első két jele utal az árucsoportra. Az alkatrészekről tudni kell, hogy melyik berendezéshez tartoznak és belőlük hány szükséges egy termékhez. Ezeket az ismeretet vezetni kívánják az ún. normáliákról - csavarok, alátétek, vezetékek stb. - is.

Az árukat célspecifikus raktárakban tárolják, vagyis egy-egy árufélét csak egy helyen tárolnak. Kivételt jelentenek a normáliák, amik minden raktárban megtalálhatók. A raktárt kód azonosítja, név és cím írja le. A lerakatok csillag-rendszerűek, vagyis a területi raktárak egy-egy központi raktárhoz tartoznak. Speciális raktárnak számítanak az üzletetek, ahol a cikkeket forgalmazzák. Az üzletet számos egyéb adat is jellemzi.

A termékekről napi készletnyilvántartást vezetnek. A berendezésekről és alkatrészekről darab, a normáliákról csomag szerint. A bevételeket és kiadásokat diszpozíció- illetve cikkszám azonosítja. Ár, mennyiség és érték írja le. Néhány esetben az eladási és a beszerzési ár kedvezményes.

#### 17.1.1 A mintapélda tartalmi hibái

**Valóságghűség.** A tartalmi problémák elemzését mindig ennél a kritériumnál kezdjük. Ilyenkor azt vizsgáljuk, hogy a tételek megfelelnek-e a tényleges jelenségeknek illetve absztrakciójuk - típusba sorolásuk - helyesen történt-e. Mivel néhány esetben az ár kedvezményes, a tulajdonságok közül a két utolsó egyed Egységár tétele hamisan tükrözi a valóságot. A két egyedbe a Beszerzési ár és az Eladási ár illik. (Ezeket csak akkor töltjük ki, ha az ár eltér az Egységártól.) Ezt a problémát formai úton, normalizálással is felfedezhetjük. A BEVÉTEL és a KIADÁS formailag részleges függést (Cikkszám → Egységár) tartalmaz. Mivel azonban a CIKK egyeddel szemben ebben a két egyed típusban az Egységár nem redundancia, hanem valójában homonima, a tételt nem távolítjuk el az egyedekből, hanem új megnevezéseket alkalmazunk.

A terv téves, amennyiben a NORMÁLIA egyedben szerepel a Raktárkód. A normáliát a leírás szerint több raktár is tárolhatja. Ezért a példa vagy rosszul tükrözi a valóságot, vagy a NORMÁLIA egyed típus nem-normalizált, mert abban a Raktárkód ismétlődő adat. Ezt a problémát is feltárhatjuk formai módon. A KÉSZLET-2 egyedben a Normáliakód és a Raktárkód hálós viszonyt mutat, a NORMÁLIA egyedben viszont - a látszat szerint - hierarchikus. Ez a belső kulcstörés tipikus esete.

A Cikkszám/Normáliakód és a Megnevezés/Név párosoktól eltekintve a CIKK és a NORMÁLIA adatai azonosak. Mi több, a Normáliakód a Cikkszám szerepneve, vagyis az is egy cikk-megjelölés. A tervező nem élt a generalizáció lehetőségével, és emiatt rossz absztrakciót alkalmazott. A két egyed összevonható éppen úgy, mint a két KÉSZLET is.

**Egyértelműség.** A Cikkszám/Cikk kód és az Árucsoport kód/Árucsoport jel párosok szinonimák, amit a kapcsolati hiányon is láthatunk. A KÉSZLET-1 és az ÁRUCSOPORT formailag nem kapcsolható a CIKK-hez, az ÁRUCSOPORT a NORMÁLIÁ-hoz sem. Ha nem fedeztük volna fel a szinonimát, akkor a kapcsolatelemzésnél derült volna ki a turpisság. (Amint látjuk, a kritériumok összefüggének. Jelen esetben az egyértelműség és a teljesség hiánya mutatkozott együtt.)

A tételek között - a vizsgálatból már kiesett Egységáron kívül - számos homonimára bukkanhatunk. Ilyen a Megnevezés és a Mennyiség. Ezek leíró tulajdonságok, ezért közös nevük csak „bocsánatos bűn”, ami megfelelő minősítéssel azonnal korrigálható. Viszont a Diszpozíciószám

is homonima. Ez a tétel kulcsrész, és így a Cikkszámmal párosban sérti az azonosítók egyediségére vonatkozó szabályt, ami szerint két egyednek (BEVÉTEL és KIADÁS) nem lehet ugyanaz az elsődleges kulcsa.

Homonimának mutatkozik az Érték is. Ezzel azért nem foglalkozunk, mert számított adatról van szó, ami sérti a „plusz egy” szabályt. Az érték a mennyiségből és az árból - amely egységár esetén egy eléréssel megkapható a CIKK illetve a NORMÁLIA egyedből - kiszámítható, tehát tárolására nem kerülhet sor sem a készletekben, sem a mozgásokban. Más lenne a helyzet, ha a tervező jelölte volna, hogy az Érték az adott egyedeket jellemzi, de azok tárolására nem kerül sor. Ekkor a homonima-probléma valós, és átnevezésekkel kell pontosítani a modellt.

A legcsúnyább tétel a Készlet. Ez a mozgás utáni készletet mutatja, tehát valójában olyan szinonima, aminek tartalma a két KÉSZLET egyed Mennyiség rovatával egyezik.

**Teljesség.** Elmaradt a tervből a Raktárkód | Főraktár-kód szerepnév megjelölés és ezért formailag nem tudható, hogy a lerakat melyik területi központi raktárhoz tartozik. Persze ezt a hiányosságot a másik kritérium - a minimalitás - vizsgálata során is feltárhattuk volna: kétszeresen szerepel a modellben a Raktárnév és Raktárcím adat. Mivel a főraktár is egy raktár, a FŐRAKTÁR egyedre egyáltalán nincs szükség; a főraktárak adatait is tárolhatjuk a generikusabb RAKTÁR egyedben.

Komolyabb gond, hogy a modell egyszerűen nem tükrözi azt a tényt, hogy a boltokban is van készlet, vagyis az üzlet is bizonyos értelemben raktár. Ha máshonnan nem, onnan mindenképpen észrevehetjük ezt a bajt, hogy az ÜZLET semmilyen más egyedhez nem kapcsolódik. Itt pedig megtorpanunk. Mert az mégsem lenne helyes, valóságghű, ha az üzleteket raktárként modelleznénk. Hiszen az üzletnek számos egyéb adata (X) is van.

A generalizáció és specializáció párosához kell folyamodni. A RAKTÁR, FŐRAKTÁR és ÜZLET egyed mindegyike névvel és címmel ellátott. Egészen príme megoldás lenne, ha alkalmaznánk a generikus EGYSEG egyedet és annak az altípusa, specializáltja lenne a sok sajátos adatot felvonultató ÜZLET. Bizonyos egységekhez kapcsolódna KÉSZLET. Ezzel megoldanánk azt a hiányosságot is, hogy a modell nem tükrözi az üzletekbe mint raktárakba történő bevételeket és kiadásokat.

**Minimalitás.** A rossz logikai redundanciákat normalizálással fedezhetjük fel. A példa szerint a Mennyiségi egység (darab, csomag) közvetlenül a cikkeket jellemzi. Ezért a két KÉSZLET egyed részleges függést mutat, mert a Cikkszám egyedül is meghatározza a Mennyiségi egységet. Ezért a KÉSZLET egyedekből ki kell emelni ezt a tulajdonságot és azt át kell tenni a CIKK egyedbe. Ha eddigre még nem jöttünk volna rá, akkor a normalizálás során azt is észrevennénk, hogy a BEVÉTEL és a KIADÁS egyedben a Készlet függ a Cikkszám+Dátum+Raktárkód együttestől. A két egyed nem volt 3NF alakban, mert nem minden meghatározó az egyed kulcsjelöltje (vö. a BCNF alakkal).

Utolsó lépésként a *rejtett fizikai redundanciákra* vadászunk. A TERMÉK szemmel láthatóan családja egyed típus akar lenni. Általában úgy illik, hogy a családja azonosítója az alapegyed kulcsának a szerepneveiből álljon. Mivel nem ez a helyzet, gyanút fogunk. Az olvasó nem fedezhette fel, hogy a kétszeresen kettős azonosítás gyönyörű példányával áll szemben. Mivel a Befogadó cikkszám+Beépülő cikkszám páros is alkalmas az egyed azonosítására, a Termékkód felesleges. Nagyon gyakran előfordul, hogy a végterméknek van egy cikkszáma is és egy termékazonosítója is (Termékkód). Ez önmagában véve nem probléma. A CIKK egyedben a Termékkód nem alternáló kulcs, mert nem minden cikk végtermék. Viszont a TERMÉK-ben a Termékkód és a Befogadó-cikkszám ugyanazt a dolgot jelöli, sértve az egyszeres azonosítás elvét.

Egy további gond: az élesszeműek nyilván észrevették, hogy az Árucsoport jel a CIKK egyedben *implicit redundanciát* okoz, hiszen ez az ismeret a Cikkszám első két jelével egyezik meg. Ezért ezt az adatot el kell távolítani, miközben a Cikkszám implicit csoportot explicitté kell tenni.

A 17.2 példa mutatja a megoldást. A { } jel a csoportokat, a | jel a szerepneveket, a  $\subset$  az altípusokat jelöli. A számított adatok helyét megadjuk, de azokat nem tároljuk majd. A megoldás egyelőre „hevenyészett”, amire magyarázatot a következő pontban adunk.

### 17.2 példa

CIKK	( <b>Cikkszám</b> , Cikk megnevezés, Egységár, Mennyiségi egység, Termékkód, Súly, <i>Egységkód</i> )
Cikkszám	{Árucsoport kód, Cikk sorszám}
ÁRUCSOPORT	( <b>Árucsoport kód</b> , Árucsoport megnevezés)
TERMÉKCSALÁDFA	( <b>Családfa azonosító</b> , Beépülő darab)
Családfa-azonosító	{Befogadó cikkszám+Beépülő cikkszám}
Cikkszám	Befogadó cikkszám, Beépülő cikkszám
EGYSÉG	( <b>Egységkód</b> , Egységnév, Egységcím, <i>Főraktárkód</i> )
Egységkód	Főraktárkód, Üzletkód
ÜZLET	( <b>Üzletkód</b> , X...) $\subset$ EGYSÉG
KÉSZLET	( <b>Készletazonosító</b> , Készlet-mennyiség, Készletérték)
Készletazonosító	{Cikkszám+Egységkód+Dátum}
Készletérték	= Készlet-mennyiség*CIKK(Egységár)
BEVÉTEL	( <b>Bevételazonosító</b> , <i>Készletazonosító</i> , Bevételi ár, Ártípus, Bevétel mennyiség, Bevételérték)
Bevételazonosító	{Bevétel diszpozíciószám+Cikkszám}
Bevételérték	= Ha Ártípus=„standard”, CIKK(Egységár)*Bevétel mennyiség, egyébként Bevételi-ár*Bevétel mennyiség
KIADÁS	( <b>Kiadásazonosító</b> , <i>Készletazonosító</i> , Kiadási ár, Ártípus, Kiadás mennyiség, <b>Kiadásérték</b> )
Kiadásazonosító	{Kiadás diszpozíciószám+Cikkszám}
Kiadásérték	= Ha Ártípus=„standard”, CIKK(Egységár)*Kiadás mennyiség, egyébként Kiadási-ár*Kiadás mennyiség

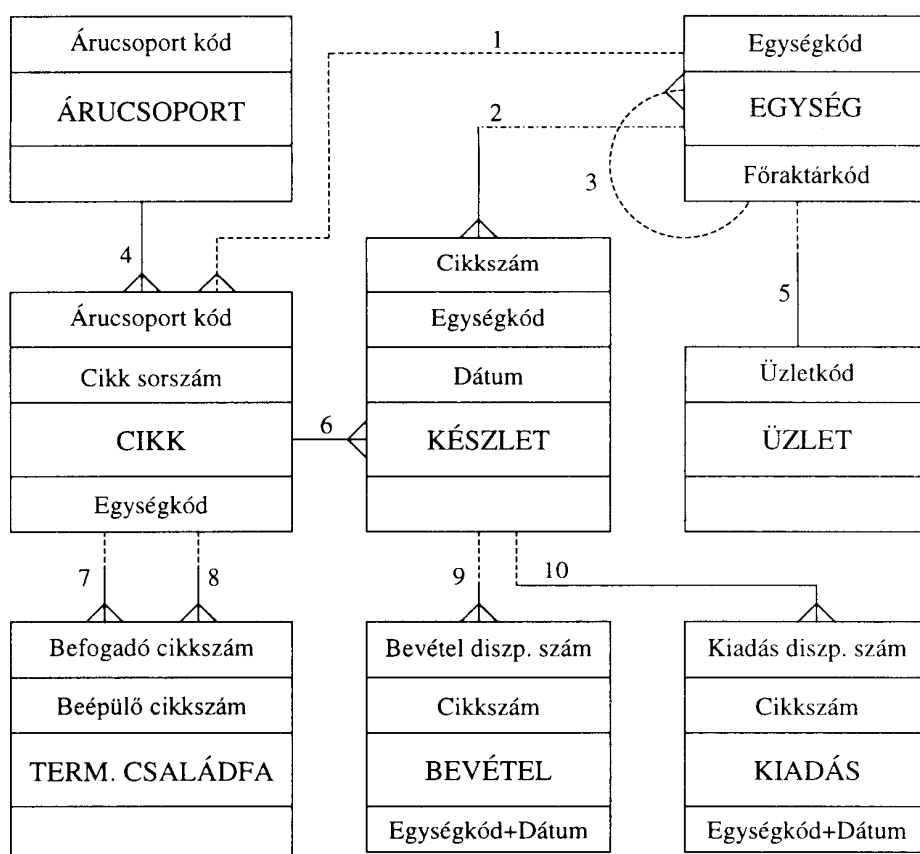
#### 17.1.2 Az adatmodell dokumentálása

Most pedig elmondjuk, hogy a fenti megoldás miért „hevenyészett”. Az adatmodell dokumentálásának igen szigorú szabályai vannak és a 17.2 példa nem tesz eleget ezeknek a követelményeknek. Nézzük csak meg, hogy miképpen épül fel egy jól-meghatározott adatmodell-dokumentáció!

**Első fejezet.** Ez megadja az *adattábazis nevét* (pl. ÁRUFORGALOM), majd röviden leírja annak célját illetve környezetét. Mondjuk így: „Az ÁRUFORGALOM adattábazis a cég összes részlege által forgalmazott termékeknek, az azokkal kapcsolatos készleteknek és készletmozgásoknak, a termékek felépítésének és tárolási helyeinek az ismereteit rögzíti. Célja az, hogy...”

A bevezető fejezetnek feltétlenül tartalmaznia kell a **Bachman-diagramot**. A mi példánkat a 17.1 ábra mutatja. A diagramhoz rövid magyarázatot illik fűzni főleg akkor, ha a modellben vannak kétségeket támasztó tényezők. Nagyon kevesen értik meg például azt, hogy a szervezeti egységek összevonhatók egyetlen egyedtípusba és a raktárakat, főraktárakat, üzleteket már azért

is egy EGYSEG-ben jó tükrözni, mert különben több KÉSZLET egyedtípusra lenne szükség. A CIKK és az EGYSEG közvetlen kapcsolata is kételyeket okozhat, mivel van közvetett kapcsolatuk is a KÉSZLET-en át. Ezért el kell mondani, hogy a berendezéseket specifikus raktárakban tárolják és arra utal a kapcsolat, miközben a normáliák több raktárban is fellelhetők.



17.1 ábra: Az ÁRUFORGALOM adatbázis-részletének a diagramja

A bevezető végén utalni kell arra a Függelékre, amely a leírásokban, listákban és diagramokon alkalmazott konvencionális jelöléseket ismerteti. Az pedig természetes, hogy a jól megszerkesztett tartalomjegyzéken kívül a bevezetőben is illik felsorolni a többi fejezet várható tartalmát. Itt kell elmondani például azt, hogy a modellt miért és miért úgy tagoltuk részekre (*Modellkeret*  $\Leftarrow$ ). Persze ilyen kis modellnél a tagolásnak nincs értelme. Ha teljes valós modellt adnánk közre, akkor abban három rendszerrészt különítenénk el: a cikkekre, az egységekre és a készletekre/mozgásokra vonatkozóakat.

**Rendszerrészek.** Egy-egy fejezetet szentelünk a modellrészeknek. Ezekben a célok és a főbb összefüggések meghatározása után a lényegi egyedek köré csoportosítva megadjuk a fogalmi szintű „rekordképeket”. A szerző az egyedeket a következő módon dokumentálja:

**Név.** CIKK. (A továbbiakra nézve lásd *Extenzió és intenzió* ⇐.)

**Extenzió.** A CIKK az általunk forgalmazott termékeket jelenti. Ezek számítógépes berendezések, alkatrészek és segédanyagok (normáliák)... A CIKK magegyed. Minimum X, átlagosan Y, maximum Z tétel nyilvántartására van szükség.

**Intenzió.** Ez a „rekordkép”. Lásd az alábbi példát. A csoportot betördelten mutatjuk az első rovatban. A szokásos félkövér/dőlt szedet mutatja az azonosító/kapcsoló szerepet. A harmadik tétel az opcionálitásra (K=Kötelező, O=Opcionális), az egyediségre (E=Egyedi, N=Nem-egyedi) és a jellegre (K=Kódolt, N=Nem-kódolt) utal. Csak akkor tekintendők a tételek kódoltaknak, ha nem külön egyedek kulcsai (pl. Egységkód), hanem majd egyetlen közös táblában kerülnek megvalósításra (pl. Mennyiségi egység).

CIKK

Tulajdonságnév	Megjegyzés	OEK
<b>Cikkszám</b>	Virtuális cs	KEN
<b>Árucsoport kód</b>		KNN
<b>Cikk sorszám</b>		KNN
Cikk megnevezés		KNN
Egységár		KNN
Mennyiségi egység kód		KNK
Termékkód	1)	OEN
Súly		ONN
<b>Egységkód</b>	2)	ONN

**Magyarázat.** Ha szükséges, a tábla alatt adjuk meg a kiegészítő magyarázatokat. Így például elmondjuk, hogy mikor tekintendő terméknek a cikk (1) és csak akkor kell az Egységkódot kitölteni, ha a tétel nem normália (2). Ha az adat kódolt és kevés kódértéke van, akkor azokat itt is meg lehet adni. Ha sok értéke van, akkor az értékek a **kódlistákat** tartalmazó külön fejezetben lesznek felsorolva. A kódok magyarázatban való felsorolása tehát kivételes eset. Azért az, mert ha egy tulajdonság többféle összefüggésben is szerepel, akkor a rá vonatkozó korlátokat nem itt, a tartalmazó táblánál („ennyszeresen”), hanem a tulajdonságlistában (lásd alább), vagyis egyszerűen célszerű megfogalmazni.

**Megjegyzés:** Logikai és fizikai szinten a táblákat kiegészítjük a rövid adatnevekkel és a szükséges ábrázolási/indexelési paraméterekkel.

**Listák.** Némileg redundáns módon külön is megadhatjuk az **egyed típusok** ABC-rendű listáját. Ez éppen úgy függelékebe kerül, mint a **tulajdonságtípusok** listája. Ez nevet, rövid leírást, általános ábrázolást és validálást, abszolút szerepet tartalmaz. Külön listákat illik szentelni a **szerep-neveknek** és a **csoportoknak**, mint sajátos tulajdonságstruktúráknak.

A **kapcsolattípusok** listájában a nevet, a fölé- és alárendeltet, a kapcsolótulajdonság nevét és a kapcsolatjellemzőket (fok, opcionálitás, jelleg) adjuk meg. Ha szükséges, rövid leírással is szolgálunk. (Mivel az alábbi táblában a név tartalmazza a fölé- és alárendeltet, itt kivételesen eltértünk a fenti konvenciótól. A nevek előtti számok a 17.1 ábrán található vonalakra utalnak.)

4	ÁRUCSOPORT - CIKK	Árucsoport kód	N	K-K birtoklási
6	CIKK - KÉSZLET	Cikkszám	N	O-K birtoklási
7	CIKK - TERMÉKCSALÁDFA-1	Befogadó cikkszám	N	O-K családfa
8	CIKK - TERMÉKCSALÁDFA-2	Beépülő cikkszám	N	O-K családfa

1	EGYSÉG - CIKK	Egységazonosító	N	O-O birtoklási
3	EGYSÉG - EGYSÉG	Főraktárkód	N	O-O visszamutató
2	EGYSÉG - KÉSZLET	Egységkód	N	O-K birtoklási
5	EGYSÉG - ÜZLET	Üzletkód	1	O-K altípus
9	KÉSZLET - BEVÉTEL	Készletazonosító	N	O-K birtoklási
10	KÉSZLET - KIADÁS	Készletazonosító	N	O-K birtoklási

17.2 ábra: Az ÁRUFORGALOM adatbázis kapcsolatlistája

A modell dokumentációjának nagyon pontosnak kell lennie. Nemcsak azt kell tudni vissza-keresni belőle, hogy a Családfa azonosítónak része a Beépülő cikkszám (az ilyen irányú felépülést hívjuk *szerkezetnek*), hanem azt is, hogy a Beépülő cikkszám a Családfa azonosító csoport tagja (az ilyen irányú beépülést nevezzük *keresztreferenciának*). Ebből következik, hogy a modell-dokumentáció szükségszerűen redundáns. A tényezők közötti összefüggéseket oda-vissza - kétszeresen - tartalmazza. Ez pedig szükséges baj. Azért az, mert vigyázni kell arra, hogy a neveket mindenütt konzekvensen írjuk le. Világos, hogy a dokumentáció egyértelműsége és teljessége csak automata segédlettel biztosítható. Az már más kérdés, hogy jó - nem papírhalmazt gyártó - automatát nehéz találni.

Az automatákkal kapcsolatosan egy másféle kettős gond is fellép. Az egyik oldalon a mai CASE és/vagy vizuális segédeszközök nem fogalmi dokumentálást adnak, hiszen azt is megengedik, hogy a logikai adatbázisokat illetve az adatmodell egyes részeit egymástól külön, vagyis nem-integráltan határozzák meg a segítségükkel. A másik oldalon nincsenek tekintettel a modell célszerű tagolására (*Rendszerezési folyamat*  $\Leftrightarrow$ ). Számukra „egyedeknek” mutatkoznak a bemeneti, kimeneti, átmeneti, biztonsági stb. táblák is. Ezen a problémán csak az eszköz módszeres használatával lehet segíteni. A tervező felelőssége az, hogy a dokumentációban külön fejezetet szenteljen az érdemi és az egyéb modellrészeknek. Erre nézve lásd a 15. MODELL ÉS RENDSZER című fejezetet.

## 17.2 Egészségbiztosítás

Pár évvel ezelőtt még az ún. „kék kártya” szolgált a társadalombiztosítási jogosultság igazolására. Azóta a régi rendszert kiváltotta a „TAJ-szám”-on alapuló, ami informatikai szempontból a korábbinál sokkal jobb. Mivel a kék kártya immár a múlté, talán nem túl nagy illetlenség, ha annak esetét felhasználjuk néhány rossz modellezési megoldásnak a szemléltetésére. Ezzel másnak bajt nem okozunk, viszont az olvasót gazdagíthatjuk.

Az alábbiakban adott résztémák köré gyűjtött példák formájában bemutatunk egy-egy elgondolkodtató modellrészletet. Ahhoz kapcsolatosan felteszünk egy vagy több kérdést. Az olvasó töprengjen el a válaszon. A helyes megoldást az utolsó alpontban adtuk meg.

### 17.2.1 Papírazonosító

A régi rendszerben magát a kártyát tekintették központi jelenségnek. Ezt a papírdarabot a Kártyaszám azonosította. A technikai jelektől eltekintve a kártyán nem volt más érdemi adat, amikor azt kinyomtatták. A kártyákat kötegekben eljuttatták a szervezetekhez illetve az egysemélyes vállalkozások vezetőinek a kártyájukért el kellett menni az adott helyre.



**17.3 példa** KÁRTYAKÖTEG-1 (*Kártyaszám -tól+Kártyaszám -ig, Partner az*)  
KÁRTYAKÖTEG-2 (*Partner az, Kártyaszám -tól, Kártyaszám -ig*)

1. kérdés: Nem sérti-e az egyedi sorszámszerű azonosítás elvét a -tól -ig tartomány?
2. kérdés: A két egyed közül melyik tükrözi helyesen a valóságot?

A szervezet kiosztotta dolgozói között a kártyákat. Ekkor kerültek a papírra a szervezet megjelölése mellett a személy egyes adatai. Az egyén ezek után elment a háziorvoshoz, aki letépvé a megfelelő szelvényt azt beküldte a TB-hez. Így egészült ki a kártya az orvos adataival. A számítógépes nyilvántartásban vezettek egy üres kártyaállományt; egy olyat; amin már rajta volt a szervezet jele; egy olyat, amire rákerült a személy is; végül pedig egy olyat, amin már az orvos hivatkozása is szerepelt.

3. kérdés: Mi a véleménye erről az adatbázisról?

Ha a személy elvesztette a kártyáját, akkor újat kapott. Ha egyszer háziorvost váltott, akkor a második szelvényt is letépték a kártyáról. Mivel csak két szelvény volt, ha valaki másodszor is orvost váltott, akkor szintén új kártyaszámú kártyát kellett igényelnie.

4. kérdés: Miért volt hibás maga a kártyaváltás elképzelés?
5. kérdés: Miért volt hibás a két orvosszelvény koncepciója?
6. kérdés: Miért nem szabad a kártyához kötni a foglalkoztató szervezetet?

Az önálló keresettel nem rendelkezők (gyerekek, öregek, betegek) más személy jogán részesültek biztosításban. Esetükben a kártyán feltüntették ez utóbbi személy adatait is. Az idevágó egyed a következő módon nézhetett ki:

**17.4 példa** KÁRTYA (*Kártyaszám, Partner az, Gyám az*)

7. kérdés: Melyik a szerepnevekre vonatkozó szabályt sérti ez a megoldás?
8. kérdés: Az egyed visszamutatást akarna kifejezni, de azt nem teszi. Egyébként is a viszonyfok miatt mindenképpen helytelen a szerkezet. Miért? Vázolja fel a jó struktúrát!
9. kérdés: Az eddigiektől eltekintve az időfaktor szempontjából miért rossz az egyed?

Mindent összevéve a fentiek jól szemléltetik, hogy komoly gondok származnak abból, ha tévesen ragadjuk meg a központi lényegét. A TB a problémát mára kiküszöbölte. Azonban más szervezetek még mindig ragaszkodnak a papírazonosítókhoz, vagyis az olyan tervrészletekhez, amelyekben az ismeretet hordozó papír és az a valaki vagy valami, amire az ismeret vonatkozik szétválék egymástól.

### 17.2.2 Háziorvos

A háziorvosokat az Orvoscód azonosítja. Ez nem más, mint az orvos pecsétjén lévő szám, ami országosan egyedi. Az orvos finanszírozásában szerepet játszik az is, hogy milyen területekre specializálódott az illető. Ezért vezetik nála a szakvizsgákat is.

**17.5 példa** BIZTOSÍTOTT (**Partner az**, Partner név, ...)  
HÁZIORVOS (**Orvoskód**, Orvos név, {Szakvizsgák})

- 10. kérdés: Miért rossz a második egyed kulcsa?
- 11. kérdés: Milyen absztrakció maradt el ebből a példából?
- 12. kérdés: Miért rossz a két egyed együttese? Adja meg a helyes megoldást.
- 13. kérdés: Hányadik normálformában van a HÁZIORVOS? Mi a teendő és miért?

A háziorvosi körzetnek nincs külön egyede. Az indoklás szerint a körzet nem más, mint maga a háziorvos. Vagyis a körzet egyenlő azzal, hogy adott helyen - nem egy épületben, hanem egy szobában! - egy orvos valamilyen napi időszakokban rendel. Ha ugyanabban a szobában délelőtt is, délután is rendel valaki, akkor az már két körzet.

- 14. kérdés: Miért teljesen hibás ez a megközelítés? Minek a tükrözésére nem alkalmas?

Az orvosoknak havonta jelentést kell készíteniük az ellátott betegekről. Ezen a papíron rengeteg adat van. Lényege az, hogy pontosan megjelöli az időt és az ellátás módját. Bár statisztikának, kimenetnek látszik, valójában korrektil tükrözi három jelenség viszonyát.

- 15. kérdés: Melyik ez a három jelenség? Az ellátás módja már a negyedik!

16. kérdés: A fentiekben rejtetten szó volt egy olyan viszonyról, ami az ellátásnak a közvetlen fölérrendeltje. Az ellátás általában (!) nem pusztán egy személyhez valamint egy orvoshoz kapcsolódik. Hanem mihez?

17. kérdés: Mit gondol, miért alkalmaztuk az előbbieken „általában” kitételt? Milyen módon kell megkülönböztetni az általánosat a kivételestől? Kivételes esetben hová mutat az ellátás ismeretsora?

18. kérdés: Megengedhető-e az, hogy kivételes esetben létrehozzunk egy olyan „kamu” SZEMÉLY/ORVOS egyedelőfordulást, ami a sürgősségi ellátottat köti az orvoshoz? Ha igen, akkor mi az extenzionális következmény és milyen többletadatot kell alkalmazni?

19. kérdés: Milyen sajátos hivatkozási nehézségeket okozhatnak a hontalanok illetve a feledékenyek? Valóban megfelelő azonosítást nyújt a TAJ-szám?

20. kérdés: Az előbbi felvetéshez kötöten lát-e olyan helyzetet, amit egyszerűen nem lehet semmilyen modellezési technikával megoldani?

Az említett beszámolón kell utalni arra, hogy a páciens nem maga a háziorvos látta el, hanem az őt helyettesítő doktor. A Kezelés időpontja a napot is tartalmazza.

**17.6 példa** ELLÁTÁS (...+Kezelés időpontja, ..., Helyettesítő orvoskód)

- 21. kérdés: Hányadik normálformában van a fenti egyed?
- 22. kérdés: Miért rossz a példa szerinti koncepció és mi a jó szerkezet?
- 23. kérdés: A helyes megoldás kényelmetlen. Miért az? Miért kell mégis alkalmazni? Gondoljon az adatok és az eljárások összefüggésének a szabályára!

**17.2.3 Finanszírozás**

A háziorvosi körzeteket régebben csak a vonatkozó önkormányzat finanszírozta. Ezért az alábbi egyed megfelelőnek látszott. Arra ugyanis nem lehetett számítani, hogy majd egyéb

finanszírozók (pl. alapítványok) is a képbe lépnek. Ma már az önkormányzat más szervezetekkel együtt közösen vállalja a körzetek költségeit.

#### **17.7 példa** KÖRZET (**Körzet az**, ..., Önkormányzat kód)

24. kérdés: A példa mégis tartalmaz a specializációval összefüggő apró hibát. Mi az?

25. kérdés: Most, hogy több finanszírozó is fellép, miképpen néz ki a helyes struktúra? Kell-e változtatni a KÖRZET egyeden? Vigyázat: a kérdés becsapós!

#### **17.2.4 Válaszok**

1. válasz: Az egyedi sorszámszerű azonosításnak nem követelménye a zártság. Ezért a megoldás elfogadható.

2. válasz: Az első egyed a megfelelő. Az új belépések, kártyaelvesztések stb. miatt egy szervezetnek többször is ki kellett adni kártyaköteget.

3. válasz: Mindaddig, ameddig a kártyán nincsen rajta az összes szükséges ismeret, az átmeneti állapotúnak tekintendő. A totális megismétlés (*Állapotmodellezés*  $\Leftrightarrow$ ) az összes létező megoldás közül a lehető legrosszabb. Átmeneteket egyébként sem modellezünk. Az adott esetben elegendő lett volna egyetlen állomány vezetése megfelelő statuskóddal.

4. válasz: A Kártyaszám egy papírt azonosított, nem pedig a biztosított személyt. Ez a megoldás nem valósághű (nem a papír, hanem a személy lesz beteg) és egyben megszegi az azonosítás alapelvét. Az elméleti elv szerint a jelenség - a biztosított - azonosítója nem változhat. Megszegése gyakorlatilag azzal jár, hogy feleslegesen kell vezetni egy összetett nyilvántartást arról, hogy melyik kártyaszám melyik másikat követte. A mai TAJ-számot már okosan használják: az kiküszöböli ezt a malört.

5. válasz: A megoldás nem valósághű, mert az N-et 2-re korlátozza. Ha két jelenség viszonyát ismerettel kell leírni, akkor a korlátos megoldás (*Többszörös kapcsolat*  $\Leftrightarrow$ ) nem vezet célra. Mindenképpen kapcsolóegyetet kell bevezetni az orvos és a biztosított M:N fokú viszonyának a tükrözésére, hiszen azt le kell írni pl. a -tól/-ig ismerettel.

6. válasz: Lásd az előző két megoldást. Ha valaki húszszor váltott munkahelyet, akkor húsz Kártyaszámot kapott „azonosítóként”.

7. válasz: Egy egyedben csak akkor szerepelhet egy elsődleges tulajdonság (Partner az) és annak szerepneve (Gyám az), ha az egyik maga az azonosító. Ez esetünkben nincs így.

8. válasz: Ma én vagyok a lányom gyámja. Majd netán meghalok vagy keresőképtelen leszek és a nejem válik gyámmá. Továbbá több lányom is van. Ezért a helyes szerkezet a következő (a második egyed első két kulcsrésze a Partner az szerepneve):

#### **17.8 példa** PARTNER (**Partner az**, ...)

ELTARTÁS (*Eltartó az*+*Eltartott az*+*Eltartás kezdő dátuma*, ...)

9. válasz: Ha a gyerek befejezi a tanulmányait és önálló kereső lesz, akkor kártyát kell cserélnie, mert nem áll fenn már az eltartás ténye. Az előző szerkezettel ez a probléma is elkerülhető.

10. válasz: Az orvos elveszíti a pecsétjét és újat kap. Vagyis megváltozik az azonosítója, ami nem engedhető meg.

11. válasz: A példa implicit specializációt tartalmaz, amit explicitté kellene tenni, mert maga az orvos is biztosított!

12. válasz: A két egyed nem kapcsolható. A HÁZIORVOS-nak rossz a kulcsa és abban komoly redundancia mutatkozik (név).

**17.9 példa** BIZTOSÍTOTT (**Partner az**, Partner név, ...) **HÁZIORVOS** (**Partner az orvos**, Orvoscód, {Szakvizsgák})

13. válasz: A HÁZIORVOS nem-normalizált (ONF). Abban fellép az egyensúlytalanság gondja, ami csak úgy oldható fel, ha a Szakvizsgákat külön leíró egyedbe vágjuk le.

14. válasz: Az orvos elköltözik, nyugdíjba megy, meghal stb. Helyét más veszi át. Tehát a megoldás még akkor sem valóságos, ha netán utódja megkapja a pecsétjét. Azért nem az, mert a hely és a személy nem azonos lényeg. A körzet és a háziorvos két külön egyed, ami között az ELLÁTJA kapcsolóeggyeddel kell kifejezni a dinamikus összefüggést. Ennek tükrözésére nem alkalmas az egyetlen HÁZIORVOS egyed, mert az statikus.

15. válasz: A három jelenség az orvos, a beteg és az idő. Ezek kulcsaiból fog állni az ELLÁTÁS egyed összetett azonosítója.

16. válasz: A SZEMÉLY/ORVOS dinamikus viszonyeggyedhez. Általában (!) az orvos azokat a személyeket látja el, akik saját betegei.

17. válasz: Az orvos köteles ellátni sürgősségi esetben azt a személyt is, aki nem a saját páciense. Az ellátásban külön adattal kell utalni a saját/nem-saját jellegre. Ugyanis az utóbbi esetben az ellátás valóban nem a SZEMÉLY/ORVOS viszonyhoz kapcsolódik, hanem külön a személyhez és külön az orvoshoz.

18. válasz: Általában az álmegoldásokat kerülni kell. A SZEMÉLY/ORVOS viszony anyagi következményekkel is jár. Az orvos olyan páciens után is kap juttatást, aki soha nem volt nála, de hozzá van bejelentve. A sürgősségi ellátott után ez nem jár. Tehát nem célszerű a „kamu” előfordulás alkalmazása. Ha mégis megteesszük, akkor átértelmezzük a SZEMÉLY/ORVOS viszony fogalmi kiterjedését, azaz extenzióját. A mondott okok miatt ezt csak akkor tehetjük meg, ha itt vesszük fel a saját/nem-saját páciens jelet. Viszont ekkor az az ELLÁTÁS egyedben feleslegessé válik.

19. válasz: Ha nincs kártyája a hontalannak vagy más, nem saját páciens otthon felejtí azt és sürgősségi ellátásban részesül, akkor nem lehet őt azonosítani. Ezért a TAJ-szám sem tökéletesen alkalmas belső azonosítónak. Azt külsőként célszerű használni, miközben belsőként inkább egy sorszámot célszerű alkalmazni.

20. válasz: A sorszám helyett a „kamu” TAJ-szám felvétele sem megoldás. Egyszerűen nem lehet megoldani azt, hogy az X orvosnál ellátott hontalant és az Y orvosnál kezeltet az egész rendszerben önmagával azonos személyként határoljuk be.

21. válasz: Az ELLÁTÁS csak INF alakú. A helyettesítő orvos egy nap több beteget is ellát. Tehát az időszaktól mint kulcsrésztől nem függ a Helyettesítő orvoscód! Ezt nyilván érzékelhetjük a többszörösségből is.

22. válasz: Az ELLÁTÁS két valós tény tükrözését keveri össze: az ellátás tényét és azt a tényt, hogy valakit valaki helyettesített. Az utóbbit egy külön családfa eggyeddel kell tükrözni. Az egyszerűen fogja mutatni, hogy az egyik orvost mettől-meddig váltotta ki a másik doktor.

23. válasz: Az előbbi helyes megoldással összetettebbé válik az egyikféle adatkezelés. Az, amelyikben ki kell gyűjteni, hogy az ELLÁTÁS előfordulások közül melyik alapul helyettesítésen. Azonban a modellezés szabályai szerint a valóságos tükrözés érdekében a modell nem torzítható el csak azért, hogy az eljárás egyszerűbb legyen. Azért nem, mert úgy másikkéle kezeléseket bonyolítunk el. Például azt, amiben ki akarjuk gyűjteni, hogy egy orvos mely másikat és mettől-meddig helyettesített.

24. válasz: Minden rendszerben általános partnerekben célszerű gondolkodni. Nem jó, ha egyes sajátos szervezeteket (önkormányzat, bankok, iskolák stb.) a mai gyakorlatnak megfelelően explicit egyed típusokként tükrözünk. A sokoldalú viszonyok miatt az a jó szerkezet, amiben

ezeket a partner explicit altípusaiként adjuk meg. Tehát a példa azt az apró hibát tartalmazza, hogy az Önkormányzat kódot nem tekinti a Partner azonosító szerepnevének és azon keresztül az önkormányzatot a partner altípusának.

25. válasz: A helyes felelet kettős. Eltekintve attól, hogy az előző válasznak megfelelő módon az Önkormányzat kódot amúgy is le kell cserélni pl. Finanszírozó azonosítóra, ami a Partner azonosító szerepneve, az egyeden nem kell változtatni. Ugyanis amikor az önkormányzat és egy alapítvány együtt gondoskodik a közzetről, akkor ezt együttesen és nem külön-külön teszik. Vagyis létezik egy sajátos partner, ami a kettőből áll. A válasz második fele úgy hangzik, hogy létre kell hozni egy partner családfát, ami a finanszírozó társasághoz köti az önkormányzatot és az alapítványt. Ehhez hasonló megoldással már találkozott az olvasó a házastársak szerződése kapcsán. Lásd a 12.13 példát.

## 18. TULAJDONSÁGMODELLEZÉS

A mai adatmodellezésre két dolog jellemző. Az egyik az, hogy a tervezők hajlamosak „nagy koncepciókban” gondolkodni és az általuk „nüanszyninak” tartott részletekkel nem foglalkoznak. Olyasmi ez, mint amikor valaki épít egy kívülről gyönyörűnek látszó házat, belülről is talán elfogadható beosztással, de a kellékek - az ajtók, ablakok, csapok stb. - megtervezése és kivitelezése gondatlan. A másik az, hogy a tervező csak a saját tervével törődik, amibe viszont másnak nem enged belátást sem, nemhogy beleszólást. Olyasmi ez, mint amikor egy társasházon belül az egyik porta így, a másik úgy néz ki, a kerttel és a lépcsőházzal pedig senki sem törődik.

„Az ördög a részletekben rejtőzik.” Ezért nem véletlen, hogy ebben a fejezetben ennyire aprólékos - olykor fárasztó és unalmas - példák sorozataival kínozzuk az olvasót. Végre tudomásul kellene venni, hogy a tulajdonságok korrekt meghatározása rettenetesen fontos dolog. Rossz téglákból nem lehet jó épületet építeni. Mindezt jól szemlélteti az alábbi három egyedpáros. Döntse el az olvasó, hogy melyik megoldás jó.

SZEMÉLY-1 (Személy szám, **Cím**)

SZERZŐDÉS-1 (Szerződésszám, Személyi szám, **Cím**)

SZEMÉLY-2 (**Személy azonosító**, Cím)

SZERZŐDÉS-2 (Szerződésszám, **Személyi szám**, Cím)

SZEMÉLY (Személy azonosító, Nem)

SZERZŐDÉS (Szerződésszám, Cím, ?)

Most jön a meglepetés. Az első páros matematikailag inkorrekt, azaz nem-normalizált. Azért látszik annak, mert a Cím tranzitív függést mutat. Ám valójában normalizált, mert persze az olvasó nem tudhatja, hogy a Cím *homonima*, tehát a második egyedben nem a személy laccímét, hanem a szerződés teljesítési címét jelenti.

A második páros matematikai értelemben véve korrekt, azaz normalizált. Pontosabban szólva annak tűnik. Nem látszik rajta, hogy a Személy azonosító és a Személyi szám egymás *szinonimái*. Következésképpen valójában a második egyedpáros nem normalizált.

Végül a harmadik páros matematikai értelemben látszólag korrekt, azaz normalizált. Nem látható benne a Cím rejtett tranzitivitása. Valójában a tervrészlet *hiányos*: a két egyed nem kapcsolható egymáshoz. Ha a viszonyt helyreállítanánk azzal, hogy a Személy azonosítót betesszük a SZERZŐDÉS egyedbe, akkor azonnal kiderülne a turpisság.

A fenti példák alapján az olvasó azt mondhatná, hogy az eljárásunk nem fair, mivel a példák olyan tényezőket és összefüggéseket tartalmaztak, amiket ő nem is ismerhetett. Nos pontosan erről szól ez a fejezet. Az adatbázis-tervezők olyan „modelleket” készítenek, amiket azok olvasói képtelenek megérteni. Amelyek látszólag (matematikai értelemben) jól tervezettek, de valóságosan (szemantikai értelemben) egyáltalán nem azok.

### 18.1 Egy átfogó példa

**Célkitűzés.** A tervezőnek sokszor nem az a feladata, hogy nulláról indulva alkosson egy új rendszert, hanem - ami az előbbinél sokkal nehezebb - a meglévő zűrzavarban kellene rendet teremtenie. Ezért ennek a pontnak a célja hármas. Először is szemléltetni akarjuk, hogy a szabad

útjára engedett fejlesztés milyen tragikomikus eredményekre vezethet. Másodszor az egyetlen példa számos oldalról történő vizsgálatával rá akarjuk ébreszteni az olvasót arra, hogy az aprónak vélt hibák mögött milyen súlyos problémák húzódnak meg a mélyben. Harmadszor a példán át gyakorlatilag is szemléltetni kívánjuk a legfontosabb modellezési axiómákat, alapelveket és kritériumokat.

**A példa.** Egy összetett feladatokat ellátó intézményben elkészítették a nagyszámú alkalmazási terület adatmodelljeit. A több tucatnyi „adatmodellben” a személyek adatai között természetesen szerepelt az anyai felmenőre való utalás is, méghozzá mintegy 70 (!) különböző adatról. Ezek közül csak párat sorolunk fel:

ANYJANEVE, ANYNEV, Anyjaneve, Anyja Neve, Eanyanev, Mnélk Anyja neve, CSANYJA, Megbízott Anyja, Megbízott Anyja neve, ANEV, M109 Anyja Neve, Munkanélküli Anyja neve, Ügyintéző-anyja neve, Segélyezett anyja neve, Banyanev, Aynja neve stb. (A neveket változtatás nélkül vettük át.)

Ez a „gazdagság” a tapasztalt modellezőben gyanút ébreszt és számos gondolatot vet fel. Az alábbi részfeladatok mindegyike egy-egy modellezési problémát szemléltet. Némelyek általánosabbakat, mások speciálisabbakat, de nem kevésbé lényegeseket.

**A megoldáshoz.** Mielőtt a részfeladatok egyenkénti megoldásának nekilátna, olvassa el figyelmesen még egyszer a fenti leírást! Állapítsa meg, hogy hányféle jellegű problémát vél felfedezni. Tehát egyelőre ne a konkrét hibákra, ne azok *menyiségére* gondoljon, hanem a szemléleti hiányosságokat próbálja meg feltárni, vagyis a bajok *minőségére* próbáljon koncentrálni! Tegyen kísérletet arra, hogy a modellezési hibákat minősítse! Csak ezután nézze át és oldja meg az alábbi feladatokat!

18.1.1 A leírás első mondatában ez a kitétel szerepelt: „...elkészítették a nagyszámú alkalmazási terület adatmodelljeit.” Ez a mondat egy igen súlyos szemléleti hibát rejt. Gondolkodjon el a modellezés lényegén és tárja fel a hiba mibenlétét!

18.1.2 Ön szerint milyen gondokat okoz a több modellben való gondolkodás? Soroljon fel néhányat, majd próbálja meg egyetlen szóban összegezni azok közös gyökerét! A több ugyanis ebben az esetben kevesebb, mint az egy. Minek a hiányára utal a több „modell”?

18.1.3 Mit gondol: a ‘Munkanélküli Anyja neve’ és a ‘Mnélk Anyja neve’ nevek azonos ismereti lényegre takarnak-e vagy sem? Melyik választás a helyes az alábbiak közül:

- a) igen
- b) nem
- c) nem lehet tudni.

18.1.4 Tegyük fel, hogy az előző feladatban a két tulajdonságnév azonos lényegre takar! Ez persze azt jelenti, hogy a kérdéses adat a vizsgált adatbázisban többször is előfordul. Ezzel milyen két modellezési elvet sért meg a vonatkozó „adatmodellek” együttese?

18.1.5 A modellekben vannak tipikus *leíró* tulajdonságok, amelyek modellezésén szinte nem is kell gondolkodni. Ön szerint hány darab egyedtípusban fordulhat elő az Anyja neve tulajdonságtípus? Az egyik választási lehetőség megtevesztő. Melyik az és miért az?

- a) Szükség szerint néhányban, de nem sokban.
- b) Tetszőleges számúban, a szükségletek szerint.
- c) Csakis egyetlen egyben.
- d) Az adatbázist illetően egyben, de a kimeneteket tekintve többben is.

18.1.6 A hidegebb „anyja” szó helyett sokszor szívesebben alkalmaznánk a melegebb „édes-anyja” kifejezést. Helyesen tennénk-e ezt a kérdéses adatmodellben is?

- a) Igen.
- b) Az adatmodell nem az érzelmek tükrözésére való.
- c) Általában nem.
- d) Nem.

18.1.7 A vonatkozó példában az Anyja neve tulajdonsághoz meghatározást is adtak, ami így hangzott: ‘Az anyja neve.’. Ön szerint ez és a hasonló definíció

- a) kevés
- b) sok
- c) sok is, meg kevés is.

18.1.8 Hogyan nevezzük azt a jelenséget, amikor egy lényegében azonos dologra eltérő megnevezésekkel hivatkozunk? Melyik modellezési elvet sérti ez a jelenség? Milyen esetben nem szenved mégsem sérelmet a vonatkozó modellezési elv?

18.1.9 Megengedett-e, hogy az adatmodellben egy lényegében véve azonos dolgot eltérő megnevezésekkel lássunk el? Ennek megfelelően helyes-e a példa adatmodellje? Mindkét kérdésre az alábbi mondatok egyikével válaszoljon úgy, hogy folytatja azt!

- a) Nem, mert ...
- b) Igen, ha ...

18.1.10 Nagyon sok esetben a tervezők úgymond véletlenül generálnak eltérő neveket az azonos tényezőkre. Ilyen esetekben technikai szinonimákról beszélünk. Mi lehet ennek a hibának az oka és a példa nevei közül melyik tartozik ebbe a kategóriába?

18.1.11 Fejtse ki, hogy miért fontos a modellben szereplő tényezők nevének a formája is! A feleletben utalni kell egy alapvető hibára.

18.1.12 Ön szerint egy összetett modellben a tulajdonságok listája alapján feltárhatók-e a rejtett - valós és technikai - szinonimák? Igen vagy nem?

18.1.13 Ön szerint van-e a fenti nevek között *formailag* minden szempontból helyes is?

- a) Nincs.
- b) Van, mégpedig a következő(k) ...



18.1.14 Miért helytelenek az ‘Anyjaneve’ és ‘Anyja Neve’ megjelölések? Egy általános, nyelvi természetű elvre kell utalnia a válaszbán.

18.1.15 Miért helytelenek a ‘CS ANYJA’ és a ‘Megbízott Anyja’ - illetve általában a hasonló, a konkrét ismeretre nem utaló - megjelölések? Ön szerint mit takar ez a két név?

18.1.16 Az írásmódtól függetlenül miért teljesen rossz az ‘ANYNEV’ megjelölés? Milyen két elvet sértenek a hasonló megnevezések?

18.1.17 Milyen az időfaktorral kapcsolatos problémát okoz az, hogy együtt szerepel a modellben a ‘Megbízott Anyja neve’ és a ‘Munkanélküli Anyja neve’ nevű tulajdonság?

18.1.18 Az ‘Eanyanev’, ‘Banyanev’, ‘ANEV’ stb. megjelölés egy általánosan elterjedt rossz tervezési gyakorlatot szemléltet. Elfeledkezve a formai és az egyértelműségi hibákról mondja el, hogy milyen módon történik ma az adatnevek minősítése illetve miért rossz is, ostoba is ez a megoldás! Gondoljon az adatbázis szintjeire és a modellezésben alkalmazott ismereti forrásokra!

18.1.19 Az ‘M109 Anyja Neve’ név egy újabb problémát vet fel. Az M109 „titokban” egy kézi nyilvántartást takar. Miért rossz ez a megoldás elvileg is és formailag is? Milyen követelményt sért a megnevezés és milyen alapvető lépés elmaradását sejteti?

18.1.20 Milyen a tulajdonságokon már túlmutató problémát jelez az, hogy a modellben található az ‘Ügyintéző-anyja neve’ és ‘Segélyezett anyja neve’ tulajdonságnév páros?

## 18.2 A tulajdonság lényege

**Célkitűzés.** Az egyik legsúlyosabb modellezési szemléleti hiba az, amikor a *fogalmi* szintű tulajdonságokat összetévesztik a *logikai és fizikai* szintű adatokkal. Tudomásul kell venni, hogy nem minden modellezett fogalomból lesz majd valós adat és megfordítva, nem minden adat modellezendő. Ennek az alpontnak az a célja, hogy rámutasson: mely adatok/ismeretek nem tárgyai a modellezésnek.

**A megoldáshoz.** A megoldásnál gondoljon arra, hogy csakis azokat a tételeket kell modellezni, amik *tulajdonságtípusok*. Ez azt jelenti, hogy

- fogalmi szintűek
- vannak előfordulásai (értékhalmazaik), azaz többféle tartalmat vehetnek fel
- adatszerű megjelenésűek (pl. a fotó nem tulajdonságtípus)
- tartósak és lényegesek.

18.2.1 Egy adatmodellben az alábbi tényezőket adták meg tulajdonságokként. Válogassa ki közülük azokat, amelyek alkalmazása elfogadhatatlan! Indokolja meg, hogy miért az!

pecsét, nyomtatványszám, vezető aláírása, cikkszám, Anyagrendelés

18.2.2 Az előző példa egy általános elvi hibát is mutat. Mérlegelje az első mondatának minden egyes szavát és az alapján mondja meg, hogy a Saját vállalatunk címe nevű ismeret elvileg miért nem modellezendő!

18.2.3 Milyen problémát lát az 'ügyintéző aláírása' nevű valaminek az adatmodellbe való foglalásában a már említetten kívül? Ez a főleg milyen hiánnyal párosulhat? Gondoljon a manuális és a gépi ismeretkezelés eltéréseire!

18.2.4 Ugyanebben a modellben tulajdonságként szerepelt az Ügyintéző telefonszáma is az intézett ügyhöz kapcsolva. Ez egy optimalitási és egy karbantartási problémát vet fel. Egyáltalán: kell-e modellezni ezt az adatot?

18.2.5 Egy kérvény modellrészletében a kérelmező személyi adatai között szerepelt az Életkor nevű tétel is. Mikor szabad és mikor nem az életkort modellezni?

18.2.6 Egy egyedben a következő két problémás adatnév található: 'BEGINGROUP' és 'ENDGROUP'. Mi velük a baj?

18.2.7 A tervező a tulajdonságok közé felvette a CDV nevű tételt is. Mint tudjuk, a CDV (Check Digit Verifier) a nem sorszámszerű numerikus azonosítók ellenőrzésének az eszköze. Mi a kettős gond ennek a modellben való tükrözésével?

18.2.8 Egy modellben találkoztunk a Gépi azonosító nevű tétellel. Ez így hibás. Egyrészt árulja el, hogy mi a hiba! Másrészt válaszoljon arra, hogy a modellben szerepelhet-e a külső, a természetes emberi azonosító pótléka, a belső technikai azonosító?

18.2.9 Egy tervben az alábbi adatnevet és definíciót adták meg. Mi velük a baj?

Név: 'Határozat szövegezése'  
Definíció: 'Előre meghatározott formális szöveg.'

18.2.10 Egy modellrészletben a következő adatnév és megjegyzés szerepelt. Milyen kommentárt fűzne hozzá?

Név: 'Alkalmazási kérelem szövege'  
Definíció: 'Csak papíron szerepel!'

18.2.11 Egy modellben az alábbi „tulajdonságtípusok” találhatók. Hány hibát lát ebben a megoldásban? Vajon mi tévesztette meg a modellezőt e részlet kialakításakor?

- hozzájárulás 0-5%
- hozzájárulás 5-15%
- hozzájárulás 15% felett.

18.2.12 A HATÁROZAT egyedben tulajdonságként adták meg a Mellékletek száma adatot is. A melléklet a határozathoz kapcsolt iratot jelenti. Milyen speciális és általános hibát vél felfedezni ebben a modellrészletben?

18.2.13 Egy egyedben szerepelt az Oldalszám nevű tulajdonság. Ez legalább háromféle problémát vet fel. Próbálja meg kitalálni őket!

18.2.14 Egy bérügyi nyilvántartás modelljében fordult elő a '4.2%-os' nevű adat. Ezzel a munkaadói járulék egyik ismeretét jelölték. A kérdés többszörös:

- a) Szabad-e az adatbázisban tárolni ezt a számított adatot?
- b) Szükséges-e modellezni az ilyen ismereteket?
- c) Megfelelő-e az adat megnevezése?

18.2.15 Egy nyilvántartásban szerepelt az 'Altér' nevű adat. Csak az adatkörnyezetből ismertük fel, hogy az átlagkeresetről van szó. Az első kérdés: Szabad-e ezt az adatot modellezni? A második kérdés: Milyen hibákat követett el a tervező?

18.2.16 A kirendeltség által kiállított papírokon található a következő adat: Kirendeltség címe. Milyen kettős hibát követett el a tervező, amikor ezt modellezte?

18.2.17 Mit gondol a 'Jelenlévő képviselő neve' adatról? Három hibát is rejt a kis részlet.

18.2.18 Egy egyedben ez az adat található: 'Kiírandó'. A tartalma nyilván igen vagy nem. Mi vele a kettős probléma?

18.2.19 Mi a véleménye a 'Tételsor állapota' nevű adatról?

18.2.20 Mit gondol a tanfolyamhoz kötött 'Feltételezett előzetes végzettségek' adatról?

18.2.21 Az egyik egyedben szerepel a 'Jelszó' - értsd: password - adat. Ez a megoldás kétszeresen is rossz. Miért az?

18.2.22 Az adatmodellben ez a három tényező is szerepel a jegyzőkönyvhöz kapcsoltn: 'AF7 Kiállítás éve', 'AF7 Kiállítás hava', 'AF7 kiállítás napja'. A tervrészlet három hibát tartalmaz. Mik azok?

18.2.23 Az egyed egyik adatalemének a neve: 'Eszköz'. Definíciója: 'Értéke mindig N'. Elfeledve a név pontatlanságát, miért hibás ez a részlet?

18.2.24 Adattétel neve: 'Szerződés állapota'. Meghatározása: 'Megmutatja, hogy él-e még a szerződés. Az adatbázisban csak élő szerződéseket tárolunk'. Mi a véleménye erről a részletről? Felesleges-e az adat?

18.2.25 Egy egyedhez kötöten szerepelt a következő adatsor: 'bevitte', 'törölte' stb. A buta neveken kívül milyen gondokat lát e részletben?

18.2.26 Szerepelhet-e a modellben az 'Állomány típusa' adat, ha az a fájlra vonatkozik?

18.2.27 Egy modellben a következő adatnevekbe ütköztünk: 'Jogos1 (karbantarthat)', 'Jogos2 (lekérdezhet)'. A tervező egyszerre több hibát is elkövetett. Mik azok?

18.2.28 Az adatelem neve: 'Verzió'. Ráadásul leíró adatról van szó. A terv két hibát is tartalmaz. Tárja fel őket. Gondoljon az időfaktorra!

18.2.29 Egy egyedben a következő nevek fordultak elő: 'X Megléte', 'Y Megléte', ..., 'Z Megléte'. Mi a véleménye e részletről akkor, ha a meglét valamilyen adatbázistényezőre vonatkozik illetve akkor, ha csak például csatolt papírokról van szó?

18.2.30 Szerepelhet-e a modellben a 'Kapják' adat, ami a jegyzőkönyvek alján fordul elő?

18.2.31 Az egyedben a következő három tulajdonság szerepel: 'X kezdete', 'X vége', 'X időtartama'. Elfogadható-e ez a tervrészlet? Gondolkodjon!

18.2.32 Egy feladatot leíró egyedben szerepel a kezdő- és a befejező dátumon kívül a 'Göngyölt napok' adat is. Milyen hiányosságot rejt ez a terv? Honnan lehet megtudni az időtartamok méretét?

18.2.33 Szerepelhet-e a modellben a 'Keltezés helye' tulajdonság, ami arra utal, hogy pl. a szerződést mint papírt hol állították ki? Például Kelt: Győr. Mi a célszerű megoldás?

18.2.34 Az adat neve: 'Rendben'. Meghatározása: 'Sikeres volt-e a visszatöltés?' Azt csak feltételezhetjük, hogy az érték igen/nem lehet. Milyen egyéb bajok vannak az adattal azon kívül, hogy a név csacsí és az sem tudható, hogy minek a visszatöltéséről van szó?

18.2.35 A SZERZŐDÉS-ben található az 'Ügyvezető neve' és az 'Igazgató neve' adat. Az előbbi a partner, az utóbbi a mi főnökünk neve. Mind a kettő logikailag hibás. Miért az?

18.2.36 Az adat megnevezése: 'VALTO'. Meghatározás: 'Az egy sorozatban könyvelt tételek összekapcsolásának jelzője.' A név rossz, a definíció nem túl sokatmondó. Csak egy dolog bizonyos. Mi az?

18.2.37 A tervben szerepel a 'Jegyzőkönyv tárgya' nevű tulajdonság. Világos, hogy ez papíradat, tehát általában nem modellezendő. Milyen - igen ritka - esetben és milyen feltételek mellett mégis az?

## 18.3 Tulajdonságnevek

**Célkitűzés.** Már tapasztalhattuk, hogy a tulajdonságnevek megválasztása nem az egyéni tervezői ízléseknek megfelelően kell, hogy történjen. Azért nem, mert a név egy fogalmat takar és azt minden fejlesztési résztvevőnek fel kell tudnia ismerni. Ezért világos, hogy a név eleget kell, hogy tegyen bizonyos alapvető követelményeknek. Az alpont példái megmutatják, hogy a névválasztás mikor jó és mikor rossz.

**A megoldáshoz.** A követelmények közül kiemelendő az érthetőség, a valósághűség és az egyértelműség (a homonimák kizárása és a szinonimák kontrollálása). Külön is figyelni kell a

minősítésre. Egy név lehet a szükségesnél csacsogóbb, vagy éppen túl szűkszavú. Mindkét hibát el kell kerülni. Végül ismételten utalni kell arra, hogy magyar modellben magyar neveket kell használni a helyesírási szabályainknak megfelelő módon.

18.3.1 A tantárgyak sorszám-jellegű kulcsát a 'Tkód' névvel jelölték. Milyen hármast jelent ez a megoldás?

18.3.2 Mi a gond a következő adatnévvel: 'Kiállítás dátumának kelte'?

18.3.3 Milyen problémára vezethet a 'Kezdetének dátuma' név? Ön milyen nevet adna ugyanennek a lénységnek?

18.3.4 Mi a baj a következő névvel: 'Munka kezdete'? Annyi sejtethető, hogy időről van szó. Miért hiányos a név?

18.3.5 Mi a gond ezzel a névvel: 'Munka kezdési időpontja'. Az itt nem közölt definíció azt sejteti, hogy valamelyik napról van szó.

18.3.6 Mire gondol a következő tulajdonságnév olvastán: 'Indoklás szövegezése'?

18.3.7 Egy adatmodellben ez a két tulajdonságnév szerepelt: 'Feladatkezdés' és 'Feladvég'. Milyen formai hibát követett el a tervező?

18.3.8 Egy oktatási modellrészben adatnévként szerepelt a 'Tanfolyam'. Mi vele a baj?

18.3.9 A következő név mindenképpen hibás. Ha innen nézzük egyféle, ha onnan, akkor másféle problémát jelez. Mi a gond az 'Érintett személyek' névvel?

18.3.10 Egy tervben a következő két adatnév szerepelt: 'empír' és 'empirkód'. A lényegről többet nem tudhatunk. Mégis milyen két hibát követett el a tervező?

18.3.11 Egy tervezetben a következő adatokat jelölték ki: FEOR, FEOR-szám, feorkod, FEOR93. A formai következtetlenségről most feledkezzen el! A kérdések az alábbiak:

- a) Hiba-e, ha a modellben együtt szerepelnek a FEOR és a FEOR93 nevű adattételek?
- b) Milyen nem várt bajok származhatnak a kettő együttes alkalmazásából?

18.3.12 Mi a véleménye a szellemesnek látszó 'HUF' adatnévről, ami a pénznemre utal?

18.3.13 Mi a véleménye a 'Határozatsorszám' adatnévről, ha definíciójának a kezdete ez: 'A határozatszám első két karaktere az évre utal.'?

18.3.14 Milyen megjegyzéseket fűzne a 'flag' nevezetű adathoz?

18.3.15 A tárgyi eszköz tulajdonsága a 'Leírás'. Definíciója: 'Vagy abszolút összeg, vagy kötelező százalék.' Három hibát is találni ebben a részletben. Mik azok?

18.3.16 Az adat neve: 'Új megállapodás'. Kódolása: 1 = új megállapodás, 2 = módosítás. Milyen kettős hibát lát ebben a részletben?

18.3.17 Az adatelem neve: 'FEOR kód'. Definíciója: 'A FEOR-szám első négy jegye.' Mi a hiba ebben a tervrészletben?

18.3.18 Az egyik adat neve: 'Van-e biztosítása?'. A másiké: 'Van-e szerződése (I = igen, N = nem)?'. Mi a hiba e két példában?

18.3.19 Mi a probléma a 'Tanfolyami létszám' megnevezéssel és adattal?

18.3.20 Az adatelem neve 'óra'. Definíciója: 'Óra, de, ha pam-pam, akkor perc, ha pim-pim, akkor nap.'. Mi a véleménye erről az adatról?

18.3.21 Egy egyedben együtt szerepel ez a két adat: 'eszköz' - 'eszközkód'. A tervrészlet kétszeresen hibás. Miért az?

18.3.22 A TERV egyed kulcsrészeként lép fel az 'év azonosító' és a 'negyedév azonosító' nevű két adat. Mi erről a véleménye?

18.3.23 Egy adatmodellben együtt szerepelt a 'munkakör', a 'foglalkozás' és a 'FEOR' nevű adat hol ilyen, hol olyan minősítéssel. Egy másikban a 'képzettség' és a 'végzettség' kifejezés fordult elő többször is. Milyen általános elvi problémákat vetnek fel az ilyen és hasonló adatnevek? Tudja ön, hogy mi a képzettség és a végzettség lényege?

18.3.24 Helyes-e a modellekben az efféle név használata: 'A személy neve'?

18.3.25 Mi a véleménye erről a névről: 'Eatime'?

18.3.26 Az egyedben egyetlen munkakörre való utalás van. A neve ez: 'LASTFEOR'. Mi a hozzáfűzni valója?

18.3.27 Mi a véleménye a következő nevekről: 'a végrehajtás időpontjának kezdete' és 'az érvényesség hatályosítása'?

18.3.28 Az adat neve 'SHJELZŐ'. Meghatározása: 'S = személy, H = hely'. Megfelelő-e az adatnév?

18.3.29 Egy egyed tulajdonságai között bújít meg a következő nevű adat: 'TOL/IG'. Mi a véleménye róla?

18.3.30 Mi a közös hiba a 'Feorkód', 'x-áfa%', 'pm-rendeletszám' nevekben?

18.3.31 Az adatelemek megnevezése: 'Alapsor(n)', ahol n=1-3. Három darab olyan alapsorról van szó, amelyek egy bizonyos papíron ilyen nevekkkel jelennek meg. Milyen hibát követ el a tervező, ha ezeket a neveket az adatbázistervbe is átveszi?

18.3.32 Az előzővel rokonnak látszik a megrendeléseken használt ‘Tételsorszám’ adat. Hibás-e az ilyen adat modellezése ezen a néven?

18.3.33 Milyen közös hibát fedez fel a következő adatnevekben: ‘alkalmazott médiák’, ‘szorgalmi út léte’, ‘alternatívák száma’?

18.3.34 A tervező a tantárgy részének - a modulnak - az azonosítóját így nevezte: ‘Modulsorszám’. Ezzel kétféle hibát követhetett el. Mik azok?

18.3.35 A modellben a következő adatnevek tűntek fel: ‘Támogatás ezévi’, ‘Múltévi támogatás’. Milyen hibát követett el a tervező?

18.3.36 A munkakör egyedben szerepel az ‘M87 Pályakezdő megfelel e?’ adat. Az ‘M87’ egy papírra utal. A formai hibáktól eltekintve mi a baj ezzel a névvel?

18.3.37 Az adat megnevezése: ‘PUFLAG’. Definíció: ‘Pénzügyi rendszer felé jelző flag. (1 esetén pénzügy látja.) Sorolja fel, hogy milyen elvi és formai bajokat rejt ez a részlet!

18.3.38 Olykor előfordul, hogy szótárkezelőnk hosszkorlátai miatt nem tudjuk megadni a kellően beszélő nevet a maga teljes egészében. Ilyenkor nincs akadálya a rövidítésnek ill. a mnemonikus kódolt névrészletek használatának. Például a tervező a ‘foglalkozás’ szót ‘fogl.’-ra rövidítheti. Azonban milyen kétféle hibát követhet el, ha nem vigyáz?

18.3.39 Mi a véleménye a következő két adatról, amikhez nincs definíció: ‘havi létszám’ és ‘leírási kulcs’?

18.3.40 A tervező a következő adatot szerepeltette a modellben, meghatározás nélkül: ‘Hiteltérítés százalék’. Milyen kétféle hiányosságról lehet szó?

18.3.41 A ‘Vállalkozás típusa’ adatnál nem szerepel meghatározás, hiszen az adott környezetben mindenki ismeri annak lényegét. Hibát követett-e el a tervező vagy sem, amikor elhagyta a definíciót?

18.3.42 Gyakran előfordul, hogy a papíron egy adatnak nincs neve, hanem csak az értékei vannak felsorolva. Például a televízióadást leíró bizonylaton a ‘fekete/fehér - színes’ értékpár szerepel. A tervező habozás nélkül saját nevet ad az adatnak, például így: Adás típusa. Milyen hibát követ(het) el ezzel a megoldással?

## 18.4 Ami a nevek mögött van

**Célkitűzés.** A tulajdonságnév a tulajdonságtípus fogalmi lényegére utaló megnevezés. Természetéből következik, hogy a rövid megjelölés olykor nem képes kifejezni a pontos tartalmat. Márpedig két tulajdonságtípus akkor és csak akkor azonos, ha nemcsak a neve, de a lényege és az értékkészlete is azonos. Ebben az alpontban a meghatározások, az értékkészlet-kijelölések, továbbá a sajátos tényezőosztályozások fontosságát emeljük ki. Bemutatjuk, hogy ezek elmaradása vagy nem kielégítő megadása milyen bajokra vezethet.

**A megoldáshoz.** A mai CASE-eszközök nem jeleskednek a tulajdonságok egyértelmű kijelölésének a támogatásában. Néhány feladatnál erre a tényre is ki kell térnünk. A példánál gondoljon arra is, hogy nemcsak a tényezőleírások - nevek, kategorizálások, meghatározások - hiánya, hanem azok terjengőssége illetve ellentmondásos volta is tervezési problémákra vezethet.

18.4.1 Egy tervben az alábbi két adatleírás szerepelt. Milyen hibákra lehet következtetni belőlük? (NB.: Az alábbi kategória a tulajdonság általános értéktartományának - az ún. doméjnnak - az ilyen-olyan osztályozására szolgál.)

‘Név: Vezető telefonszáma. Kategória: kód.’

‘Név: Felelős telefonszáma. Kategória: szám.’

18.4.2 Tipikus tervezési hiba, amikor a Neme nevű adat a következő definícióval párosul: értéke férfi, nő vagy gyerek. Mi a hiba lényege?

18.4.3 A Státusz adattétel neve nem eléggé beszélő. Definíciója a következő: ‘1 = x ideje regisztrált munkanélküli, 2 = pályakezdő munkanélküli’. Miért pontatlan a tervrészlet?

18.4.4 Milyen sajátos, az értékkel kapcsolatos gondot vet fel az alábbi adat és leírása?

‘Név: Átképzett. Meghatározás: Ha értéke 1, akkor átképzett.’

18.4.5 Milyen általánosan elkövetett hibát takar a következő részlet?

‘Név: Levizsgázott. Meghatározás: I = igen, üres = nem’.

18.4.6 Érez-e valami gondot a következő tétel definíciójával kapcsolatosan?

‘Név: Haskorlétsz. Meghatározás: Hasonló korábbi létszám.’

18.4.7 Egy csöppnyi félrevezetés található a következő részletben. Mi az?

‘Név: Végzettség kódja. Meghatározás: Természetes azonosító.’

18.4.8 A könyv egyed egyik adata az alábbi. Miért többszörösen rossz a tervrészlet?

‘Név: Oldal. Meghatározás: Oldalszám illetve ív.’

18.4.9 Az egyik CASE-eszköz lehetővé teszi az adat kategóriájának és mértékegységének a meghatározását. Ez alapján véve helyes és még az is csak bocsánatos bűn, hogy a magyaroknak is csak CHAR, NUMBER stb. nevű választások állnak a rendelkezésükre. Viszont a kategóriák között van NUMBER, DECIMAL, FLOAT stb. lehetőség. Ez miért baj a fogalmi modellezés szintjén?

18.4.10 Az előbbi példát folytatva a következő tervrészlettel találkoztunk. Miért hibás?

‘Név: Rendelésszám. Kategória: szám. Mértékegység: szám.’



18.4.11 Nálunk sok helyen pontatlanul használják a Megyekód adatot. Mi e pontatlanság lényege? Gondoljon az adat szokásos értékeire!

18.4.12 A DOLGOZÓ egyik adata a Szolgálati hely. Meghatározás: 'Vagy település, vagy szervezeti egység'. Mire következtethet ebből a definícióból?

18.4.13 Közismert tény, hogy iratokat nem tudunk kezelni. Egy modellben a következő adatok fordultak elő: 'Iktatószám', 'Iktató főszám' és 'Iktató alszám'. Részlet a közös meghatározásból: 'Ha a főszám így, akkor az alszám 5 jegyű és..., viszont ha a főszám úgy, akkor az alszám további alszámra bomlik és 3+1 jegyű.' Mik itt a bajok?

18.4.14 Egy modellben szerepelt az 'Alkalmazott iskolai végzettsége' nevű adat. A definíció szerint ez a 'foglalkoztatott legmagasabb iskolai végzettségi szintjét' jelenti. Értéke 0-9. Részlet a kódolási sémából: 1-7 = általános, 8 = elvégezte a nyolcadikat stb. Négy probléma is feltárható ebben a részletben. Ön hányat fedez fel?

18.4.15 Az adat neve 'Szakgyakjel'. Meghatározása: 'Szakmai gyakorlat-e a külső hely?'. Hibátlan-e ez a tervrészlet?

18.4.16 Az ÜGYFÉL egyedben a következő két adat szerepel: 'Cím1, Cím2'. Definíciók: 'Cím1 = irányítószám', 'Cím2 = utcacím'. Milyen hibákat követett el a tervező?

18.4.17 A tervező a GAZDÁLKODÓ SZERVEZET egyedben a következő két adatnevet használta: 'gazdnev1', 'gazdnev2'. A definíciók szerint az első a teljes megnevezés, a második a rövidített illetve fantázianév. Hibás-e a terve?

18.4.18 A JOGCÍM egyedben szerepel a 'Jogcím megnevezés' és a 'Rövid név' adatpáros. Az utóbbi név alulminősített, mert sok mindennek lehet rövid neve. Ezért a 'Jogcím rövid megnevezése' nevet kellett volna használni, amivel elkerülhető lenne a (potenciális) homonima. A rövid névre a táblákon való kiíráshoz van szükség. Ha feltételezzük, hogy a tervező a pontos nevet használta, akkor hibátlan a terv?

18.4.19 Az előző példában helyesen járt-e el a tervező akkor, ha a rövid név mindig a teljes megnevezés első X darab karaktere?

18.4.20 A MEGHIRDETETT ÁLLÁS egyedben van a 'követelmények' tétel. Definíciója: 'végzettségi, nyelvi, gépjárművezetési és más speciális feltételek'. Milyen hibákat követett el a tervező?

18.4.21 Az adatmodellből a következő nevek ütköztek ki: 'Összeg1, Összeg2, Összeg3, Összeg4'. A tervező sorra megadta a meghatározásokat. 'Összeg1 = Nyitó, Összeg2 = Tartozik, Összeg3 = Követel, Összeg4 = Záró'. Helyesen járt el?

18.4.22 A 'határozat típusa' nevű adat több egyedet is leír. A definíció az alábbi módon néz ki. Milyen alapvető hibát követett el a tervező?

'Ha az X határozatról van szó, akkor az E1 érték azt jelenti, hogy...'

'Ha az Y határozatról van szó, akkor az E1 érték azt jelenti, hogy...'

18.4.23 Az adat neve a postai küldemény egyedben 'Küldő'. Definíció: 'A küldeményt feladó személy vagy szervezet'. Tartalmaz-e komoly hibát ez a tervrészlet? Milyen kisebb veszélyt okozhat mindenképpen?

## 19. EGYEDMODELLEZÉS

### 19.1 Csoportok és rejtett egyedek

**Célkitűzés.** Az egyed- és a tulajdonságtípus relatív modellezési tényezők. Egy valós jelenséget a helyzettől függően egyedként és/vagy tulajdonságként tükrözhetünk a modellben. Így például a szín lehet ismeretekkel leírandó dolog, azaz egyed, miközben nyilván olyan tulajdonság is, amit más egyedhez (pl. gépkocsi) kapcsolunk. A tervezők olykor tévesen választják ki a modellbeli reprezentánst. Azaz egyedként tükröznek olyan jelenséget, amit helyesen tulajdonságként kellene felfogni. A fordított megoldás pedig még ennél is gyakoribb: tulajdonságként tükrözik az önállóan megfogandó lényeket. Ez a példasorozat támpontot kíván nyújtani a helyes reprezentáns kiválasztásához.

**A megoldáshoz.** A megoldásnál gondoljon az *egyed* meghatározására, ami szerint az egyed az *önálló lényeggel* rendelkező és *saját ismeretekkel* leírandó jelenség tükörképe. Ez azt jelenti, hogy azonosítható tényezőről van szó. Ezt azért kell hangsúlyozni, mert vannak olyan ismeretsorok, amelyek bizonyos fokú önállósággal rendelkeznek, de nem írandók le saját ismeretekkel és ezért nincs is külön azonosítójuk. Ezek a *csoportok*. A modellezés egyik feladata az, hogy eldöntsük: egy ismeretsor csoportként vagy egyedként kerül-e megfogalmazásra. A megoldásnál mindig a többszörösségre kell figyelni. A többszörös hasonló név, a többes számú megnevezés és a többféle tartalom (az adat nem egy értéket vesz fel) mind-mind esetleges külön egyedtípus előjelzője.

19.1.1 A személyek leírásában a következő két adat szerepel: Cím1, Cím2. Ez a megoldás több kisebb és egy súlyosabb problémát vet fel. Az utóbbi a számokhoz kapcsolódik. Ön milyen hibákat vél felfedezni?

19.1.2 Az előző példával kapcsolatos aggályok nem alaptalanok. Egy egyed adatai között a következők találhatók. Mi a véleménye a dolgokról? Ön szerint miképpen keletkezhet efféle sületlenség?

‘Név: Kitöltésért felelős neve1’

‘Név: Kitöltésért felelős neve2. Meghatározás: A kitöltő hivatali státusza.’

19.1.3 Egy egyed tulajdonságai között ez a kettő is szerepelt: Jogcím1 és Jogcím3. Mi a véleménye erről? Milyen súlyú a probléma?

19.1.4 Egy számlán Összeg1, Összeg2, Összeg3 és Összeg4 adat található. Mindenki megesküszik, hogy több nem lesz és mindig mind a négyet ki kell tölteni. Első kérdés: Nem kellene-e mégis inkább külön egyedbe levágni az összegeket mint ismétlődéseket? Második kérdés: Megfelelőek-e a nevek?

19.1.5 Az OKTATÁSI PROGRAM egyedet a Beszámítható előtanulmányok adat írja le. Milyen problémára utal a névbeli többes szám?

19.1.6 Az előzőhöz hasonló problémát vet fel a Betölthető alapmunkakörök nevű adat. (Ezt szándékosan írtuk úgy, ahogy a tervben szerepelt.) Definíció nem áll rendelkezésünkre. Milyen kettős bajt lát a kérdéses adat mögött?

19.1.7 A RENDEL egyed egyik tulajdonsága a következő: 'SZLA'. Nem tekintve a nevek konvencióinak a hiányát és elárulva, hogy a 'SZLA' számlát jelent, milyen problémát vél e tervrészletben felfedezni?

19.1.8 A JOGOSULTSÁG egyed egyik változatában a Jogdat és a Jogdat1, másik verziójában csak a Jogdat1 adat szerepel. Mindkét megoldás rossz. Miért az?

19.1.9 A TANTÁRGY egyed egyik tulajdonságának a neve Használt eszköz. Mi itt a baj?

19.1.10 Kitérő. Az előző egyedben más eszközre utaló tulajdonsággal nem találkoztunk. Ebből arra következtettünk, hogy nincs is másféle eszköz. Mi tehát a probléma?

19.1.11 Standard adat a személyi nyilvántartásokban a Legmagasabb iskolai végzettség, ami a konkrét iskolára - tehát nem az iskola szintjére - utal. Ez pedig hiba. Miért az?

19.1.12 Szokványos adat a személyi nyilvántartásokban az Előző munkahely. Az előző példával összevetve vajon ezt a megoldást is hibásnak kell-e tekinteni? Mikor nem?

19.1.13 Az adat neve: 'Szerzmoddat'. Meghatározása: 'Ha volt módosítás, akkor az első módosítás dátuma.' Mi a kettős gond ezzel az adattal?

19.1.14 Egy összetett modell egyik részében a Születési dátum nevű adat, másik részében a Születés éve, Születés hava, Születés napja hármas szerepelt. Milyen alapvető elvet sértett meg a tervező?

19.1.15 Egy egyed típus leírásában találtuk a 'Kod1', 'Kod2', 'Kod3' nem éppen kifejező, magyartalan megjelöléssel ellátott adatokat. A tervező mindenképpen hibázott. Melyik két elv egyikét sértheti e megoldással?

19.1.16 Egy összegző egyedben szerepelt az Sz1, Sz2, ..., Sz12 névsorozat. Az Sz1 januárt, az Sz12 decembert jelentett - amire úgy kellett rájönni, mert meghatározást nem adtak. Ismétlődő adatról van-e szó ebben az esetben? Milyen hibát követett el a tervező?

19.1.17 A modul egyedben található a Vizsga részei nevű tulajdonság. Milyen hibára utal ez a megnevezés?

19.1.18 Egy könyvelési környezetben szerepelt az Alszámlaszám adat. Ez mindig a Számlaszám mellett fordult elő, de ez megfordítva nem volt igaz: a Számlaszám nem mindig az Alszámlaszámmal együtt jelent meg. Milyen esetleges tervezési hibát lát ebben a részletben? Mikor nincs szó hibáról?

19.1.19 Lehetnek-e a modellben ilyen tulajdonságok: 'Egyéb feltételek', 'Megjegyzések', 'Kikötések', 'Mellékletek' stb.?

## 19.2 Az egyed lényege

**Célkitűzés.** Az eddigiekben többször hangsúlyoztunk két kitélt. Egyrészt azt, hogy a modellezés az általános, lényeges és tartós jelenségek tükrözésére szolgál. Másrészt azt, hogy az adatmodell nem azonos az adatbázistervvel. Ez a pont olyan példákat mutat be, amelyek valamilyen módon sértik az alapelveket. Néhány furcsa esettel is találkozni fog. Ne lepődjön meg: a vonatkozó részleteket valós adatmodell-dokumentációból vettük át!

**A megoldáshoz.** A józan észre hagyatkozva gondolkodjon el azon, hogy ön mit és mit nem tekintene egyedtípusnak.

19.2.1 Az adatmodellben szerepel a KÉRELEM nevű egyed. Kommentár: Csak a kézi nyilvántartásban szerepel. Mi az ön következtetése?

19.2.2 A tervező elképzelt egy CÍMLISTA egyedet, amit az ÜGYFÉL/CÍMLISTA kapcsoló egyeddel az ÜGYFÉL-hez kötött. A címlista a különböző ügyleteket tartalmazza (ezért persze a neve rossz), amiknek kapcsán a kapcsoló egyedbe be lehet írni az érintett ügyfeleket. Ön mit gondol erről a modellrészletről?

19.2.3 A modellben van egy HAVIJÁRULÉKOK egyed. Ennek adatait a bérszámfejtés során válogatják össze statisztikai/elemzési célokból. Mi vele az egyetlen probléma?

19.2.4 A KÁRTYAELL definíciójából tudjuk, hogy az a KÁRTYA adatait tartalmazza még ellenőrizetlen állapotban. Éppen arra szolgál, hogy rajta végrehajthassák a köteget ellenőrzést. A megoldás ellen nincs semmi kifogás. Csak éppen... Folytassa a gondolatot!

19.2.5 A KONYVSEG a meghatározás szerint segédtábla, ami a könyvelési táblák neveit és paramétereit tartalmazza. Miért említjük ezt a példát?

19.2.6 Egyednév: TERVKALK. Meghatározás: A tervek kalkulációjának a fejkordja. Ez a részlet csak egy szépséghibát tartalmaz. Mi az? Gondoljon a tervezési szintekre!

19.2.7 A NYOMTAT egyed a definíció szerint a kiíratás módját vezérli. Adatai többek között: Adatnev, Poztol, Pozig. Mit gondol e részletről?

19.2.8 Egyednév: KONZULTÁCIÓ. Definíció: Az egyed megszűnt. Mi a véleménye?

19.2.9 Az ÁGAZAT egyednek nincs egyetlen tulajdonsága sem, jóllehet más egyedek meghatározásaiban sokszor hivatkoznak a vele való viszonyra. Ön szerint egyednek tekinthető-e a vonatkozó tényező?

19.2.10 Az egyed neve BÉRKARTON. Tulajdonságai között fel vannak sorolva mindazok az adattételek, amik valóban megtalálhatók minden rendes bérkartonon. Azonban a tervnek ez a részlete mégsem fogadható el adatmodellrészként. Miért nem?

19.2.11 A PARAMETEREK egyeddel már nem is kell többet törődni. Miért nem?

19.2.12 A FEJLEC egyed a definíció szerint a saját cégünk nevét, címét stb. tartalmazza. A tervező hibát követett el, amikor felvette a modellbe. Miben áll a hiba elvi lényege?

19.2.13 Az ORSZÁG egyed egyetlen tulajdonsága az Ország. A kérdés kettős. Egyrészt *általában* szabad-e modellezni olyan egyedet, aminek csak egy tulajdonsága van? Másrészt *speciálisan* az adott tervrészlet jó-e?

19.2.14 Az OKTATÓ egyed leírása: Nem használatos. Helyette a DOLGOZÓ egyedet alkalmazták. Mi a véleménye e részletről?

19.2.15 A SZOBA tulajdonságai: épület, ajtószám, telefonszám stb. Ezen kívül szerepel benne az ügyintéző neve is. A tervrészlet a cég roppant nagy gazdagságát sejteti amellet, hogy van vele egy kis gond. Miből sejthető, hogy gazdag a cég és mi a probléma?

19.2.16 Az egyednév: X\_TANF1. Ebből már tudjuk, hogy felesleges és/vagy sok elemzési gondot okozó dologról lehet csak szó. Vajon miért lehetünk ebben biztosak? (A TANF tanfolyamot jelent.)

19.2.17 Az egyed: ÉPÜLET-MELLÉKLET. Leírása: az épület műszaki dokumentációja. Itt valami nem helyénvaló. Mi az?

19.2.18 Az egyed neve: ADATLAP. Ebből a névből már tudjuk, hogy nem lesz könnyű dolgunk az elemzéssel. Vajon miért nem?

19.2.19 Az egyed neve: JELENLÉTI ÍV. Modellezhető-e ez a dolog egyedként? Ha igen, akkor miért? Ha nem, akkor miért nem és mi modellezhető?

19.2.20 Az elemzési munka során a JEGYZŐKÖNYV név olvastán felsőhajthatunk. Miért?

19.2.21 A modellben szerepel a CIKK1 és a CIKK2 egyed. Tulajdonságaikból tudható, hogy nem altípusokról van szó, hanem a cikkek két nézetéről, amelyeknek adatai átfednek egymással. A modellrészlet rossz. Miért az?

### 19.3 Rossz egyedmegadások

**Célkitűzés.** Az egyedek kijelölése akkor is rossz, ha a tényező nem valósághű, nem egyértelmű és az aggregálási/specializálási absztrakciókat nem alkalmazták jól. Az alábbi példasorozat a téves egyedmegadásokra mutat eseteket.

**A megoldáshoz.** Gondolkodjon el azon, hogy ön hol vonná össze vagy bontaná szét az ismertetett egyedeket. Vizsgálja meg az általánosítás/specializáció lehetőségeit. Végül vesse fel az egyszerűség/többszörösség kérdését is. A tervezők igen sokszor egyszerűsnek feltételezik a többszöröset. Lásd például majd a devizával kapcsolatos feladatot.

19.3.1 Az adatmodellben tucatnyi ...-PÁLYÁZAT nevű egyed szerepel. A '...' jelzőtől függetlenül ezeknek az egyedeknek a tulajdonságai 80-90 %-os mértékben megfelelnek egymásnak. Ugyanez a megállapítás vonatkozik a viszonyaikra is, hiszen a pályázatokhoz mindig kapcsolódik pályázó, bíráló és bíráló, a pályázatot kezelő szervezeti egység stb. Ráadásul egyes pályázatok egymást kizárják, mások egymást feltételezik. Mindezek ismeretében mit lehet elmondani a modelltől? Milyen hibák forrása lehet ez a terv?

19.3.2 A biztosítónál úgy döntöttek, hogy módozatoként külön rendszereket készítenek. Az ilyen-olyan élet-, gépkocsi- és egyéb biztosítások ezen rendszereinek mindegyikében van ügyfél, szerződés stb. egyed. Mi a véleménye erről a megoldásról?

19.3.3 A modellben szerepel SZERVEZET, GAZDÁLKODÓ SZERVEZET, CÉG, ADAT-SZOLGÁLTATÓ SZERVEZET, FOGLALKOZTATÓ, ALKALMAZÓ stb. egyed. Már e nevek-ből is lehet tudni, hogy nem jó a modell. Egészen bizonyos, hogy két egymással összefüggő hibával lesz teletűzdelve. Mik ezek a hibák?

19.3.4 Az EGYEN1 és EGYEN2 egyed adatai nem fednek át egymással. Azonban ez a részlet hibás annak dacára, hogy a közös kulcsra át a két egyed egymáshoz kapcsolható. Milyen elvi hibáról van szó?

19.3.5 Egy adott szervezet országos hálózata megyei központokból, azokon belül helyi kirendeltségekből, legvégül pedig irodákból épül fel. A tervező ezért KÖZPONT, KIRENDELTSÉG és IRODA egyedeket vett fel. Jó-e a megoldása? Ha igen, akkor miért? Ha nem, akkor milyen problémákat von maga után?

19.3.6 Az egyed: VÁRAKOZÓ ÜGYFÉL. Most lényegtelen az, hogy mire is várakozik. A kérdés az, hogy létezhet-e ilyen egyed egyáltalán? Ha igen, mikor? Egyébként milyen problémákat okoz az alkalmazása?

19.3.7 A támogatásért folyamodó SZERVEZET akkor kapja meg a hitelt, ha az előző évi eredményei megfelelőek. Ezért a tervező az egyedbe felvett egy Előző évi eredmények adatsort, amit szépen-rendben ki is fejtett. Célszerű megoldást választott?

19.3.8 A tervező a SZERVEZET tulajdonságaként vette fel a Számlaszáma adatot. Ezzel két hibát követett el. (NB.: A bankszámláról van szó.) Mik azok és mi a helyes megoldás?

19.3.9 A modellben a SZÁMLA és a DEVIZA egyedek közül az utóbbi a Devizanem, Dátum és Átváltás ismereteket rögzíti. A számlában utalás történik a Devizanemre és a számlának természetesen van dátuma. A két egyed így kapcsolható, a modell mégis hibás, mert nem valóságghű. Mi a hiba lényege?

19.3.10 A biztosítási rendszerben a tervező külön KÓRHÁZ egyedet vett fel. Az INTÉZMÉNY-hez képest ennek csak egy-két sajátos adata van. A többi adat közös, például a kórháznak is vannak számlái. Milyen hibát vétett a tervező? Ha az általános SZERVEZET egyedre is gondolunk, mi az egyetlen helyes megoldás?

19.3.11 Az adatmodellben GAZDÁLKODÓ SZERVEZET, PÉNZINTÉZET, ISKOLA, SZÁLLÍTÓ stb. egyedeket találjuk. Így azonnal tudhatjuk, hogy a modell rossz. Elegendő egyetlen tulajdonságot keresni. Vajon melyik lehet az árulkodó tétel?

19.3.12 Egy oktatási modellben szerepelt a MODUL/TERVEZETT MODUL egyedpáros. Mindkettőt a Modulszám azonosította, sorszám jellegű adatként. A részlet többszörösen rossz. Mik a hibái?

19.3.13 A modellben KONFERENCIA és SZEKCIÓ egyedtípus szerepel. Az utóbbi kulcsa a Konferencianév+Szekciósorszám együttese. A leírás szerint a Konferencianév mindig ugyanaz. Milyen elemi hibákat rejt ez az „apró” kitétel?

19.3.14 Milyen hibát sugall az MSZ SZEMÉLY egyedmegnevezés?

19.3.15 Az egyednév: NYELVTUDÁS. Meghatározás: Csak a HALLGATÓ és TANÁR egyedekhez kapcsolódik. Ez a részlet egy speciális hibát és egy általános félreértést mutat. Mik azok?

19.3.16 A SZEMÉLY egyed egyik tulajdonsága az Önadózás jele. Ez egy igen komoly modellezési hibát takar. Január, december ... Mi a hiba lényege? Hasonló gondot mutat például a Besorolás tulajdonság is.

19.3.17 A SZEMÉLY egyed tulajdonságsorában szerepelnek a szabadságra, a betegségre, az egyéb távollétre vonatkozó adatok. Ezek egyszeresek, nem ismétlődőek. A modell mégis hibás. Miért az?

19.3.18 A modellben Bank, BANK, CEGBANK stb. nevű egyedek találhatók. Nemcsak a tulajdonságnevek, hanem az egyednevek írásánál is hiba, ha nem alakítunk ki írásmód konvenciókat és/vagy technikai szinonimákat tartalmaz a modell. Sőt, ebben az esetben a baj még nagyobb. Miért az?

19.3.19 A modellben szerepel egy BÉRJOGCÍM és egy KIFIZETÉSI JOGCÍM egyed. Ön szerint hibás-e ez a terv, ha az utóbbi egyed a szerződésekkel kapcsolatos és nem a bérekre vonatkozik?

19.3.20 A bérügyi modellrészben fontos jelenség a kifizetőhely. Ezért a tervező létre is hoz egy ilyen nevű entitást. Ezzel szerintünk hibát követett el. Úgy véljük, hogy a kifizetőhely nem egyed, hanem tulajdonság. És ön?

19.3.21 Az adatmodellben van SZEMÉLY, ALKALMAZOTT és DOLGOZÓ egyedtípus. Alkalmazott az, aki belsőként vagy külsőként sajátos munkát végez. Milyen problémát von maga után ez a tervrészlet?

19.3.22 Az egyed neve: JOGCÍMKÓD. Van hozzá megjegyzése? A tervező nyilván egy Jogcímkód/Jogcímnév párost tartalmazó tipizáló egyedre gondolt.

19.3.23 A modellben a TERV, TERVTÍPUS és TERVTÍPUSVÁLTOZAT egyedhármas található. A TERV meghatározása: A tervtípus vagy a tervtípusváltozat konkrét megvalósulása. Ez a modellrészlet egy komoly átgondolási hiányt rejt, amit a „vagy” szócska sejtet. Miért?



19.3.24 Tipikus tervezői hiba, hogy a térben, időben vagy fizikailag többszörös jelenségeket és azok mintáját nem jól modellezzik. Például fizikailag többszörös jelenség a könyv (több példány), aminek mintája a mű (a több példány közös lényegét leíró valami). Ebben a példában a tervező jól látta, hogy a tanfolyam egy olyan közös lényegű minta, ami időben többszörös. Például a szervezői tanfolyamot x alkalommal tartják meg. Ezért létrehozott egy TANFOLYAM és egy AKTTANFOLYAM nevű egyedet. Azonban mindkettőnek a Tanfolyamszám kulcsot adta, ami nyilván tévedés. Miért az és milyen elemzési problémát generál?

19.3.25 Az előző példa folytatása. Az egyednév: HALLGATÓ/TANFOLYAM. Definíció: A tanfolyam és a hallgató viszonya. Kérdés: Miért rossz a név is és a meghatározás is?

19.3.26 Az egyed neve: VIZSGA. Meghatározása: Kapcsolat a hallgató és a tárgy között. Az adatok között szerepel a vizsga helye is. Helyes-e ez a modellrészlet?

19.3.27 Bizonyos egyedekből hiányoltuk a Régiókód adatot. A tervező erre kijelentette, hogy ez a tétel felesleges, mert amúgy is régió szintű rendszert készítenek. Milyen elvet sért a modellje?

19.3.28 Az egyik alrendszerben megtalálható a HALLGATÓ/TANFOLYAM, a két egyed kapcsolatát tükröző egyed, a másikban pedig a HALLGATÓI RÉSZVÉTEL egyed. Ez utóbbi definíciója: A hallgató és a tanfolyam viszonya. Mi a véleménye a modellről?

19.3.29 Az egyedtípus neve: SZERZODES RENDELES. Meghatározása: Vagy szerződés, vagy rendelés. Mit gondol e tervrészletről?

19.3.30 Az egyed neve: BIZONYLAT TETEL. Itt akár meg is állhatunk, mert a modell biztosan rossz. Miért az?

19.3.31 Van két egyedünk: NYELV és PÓTLÉK. Mindkettő két tulajdonságot tartalmaz: Kód és Megnevezés. Milyen hibát követett el a tervező? Gondoljon a tervezési szintekre!

19.3.32 Az előző példában már maga a PÓTLÉK megnevezés is némileg inkorrekt. Vajon miért az? Gondoljon az általános és a konkrét különbségére.

19.3.33 Egyednév: NYELVTUDÁS. Meghatározás: A hallgató nyelvismeretét írja le az adott nyelvből. Tulajdonságok: Kód és Megnevezés. Többféle hibát is elkövetett a tervező. Mik azok?

19.3.34 Egyednév: SZAKKEPZ. Meghatározás: Ez kapcsoló a SZEMÉLY és a SZAKMA között. Nincs attribútuma. Mi a véleménye erről az elemről?

## 19.4 Azonosítók

**Célkitűzés.** Az azonosító nem más, mint az egyed reprezentánsa. Ezért aki rosszul választja ki a kulcsokat, az nyilván helytelenül tervezi meg az egyedet is. Tervezőink nem jeleskednek az azonosítók kialakításában. Nem gondolnak az azokkal kapcsolatos egyértelműségi, egyszerűségi, teljességi, minimalitási és kapcsolhatósági kritériumokra. A pontban bemutatott példasor az ebből fakadó problémákat szemlélteti.

**A megoldáshoz.** Gondoljon arra, hogy az azonosítónak típus és előfordulás síkon is kölcsönös és egyértelmű viszonyban kell állnia az egyeddel. Ne feledje el azt sem, hogy az azonosító kapcsolási célokat is szolgál. Ezért kiválasztásakor nem csak az azonosított, hanem a kapcsolt egyedekre is gondolni kell.

19.4.1 A KÉSZLET egyedhez kapcsolt MOZGÁS egyednek a tervező a Bizonylatszámot választotta azonosítóul ezzel a névvel. Helyesen járt el?

19.4.2 Milyen többletgondot okoz az elemzésben, ha az azonosító szerepű tulajdonságok homonimák? Gondoljon arra, hogy az azonosítók egy másik szerepet is betöltenek! Tehát miért hiba az, ha például több egyedben szerepel azonosítóként a Törzsszám nevű adat?

19.4.3 Az előző példát folytatva: miért nem célszerű az olyan minősítetlen név használata a kulcs szerepű tulajdonságokra, mint amilyen pl. a Belső azonosító?

19.4.4 Milyen többletgondot okoz az elemzésben, ha az azonosító szerepű tulajdonságra más egyedekben szinonim névvel hivatkoznak? Például az azonosított egyedben Belső azonosító, a többiekben Szerződésszám a megnevezés? Erre a kérdésre akkor tud igazán helyesen válaszolni, ha ismeri az alternáló kulcs (kulcsváltozat) lényegét és tudja azt is, hogy egyes esetekben ez a kulcs nem valódi változat.

19.4.5 Az Iktatószám hol ilyen, hol olyan felépítésű és éppen ezért valójában nem is egyetlen tulajdonság. Mivel nem fogunk annyi adatfélélt alkotni, ahányféle iktatószám létezik, erről a valamiről el kell döntenünk, hogy fogalmi szinten strukturálatlan, csak eljárás által strukturálható adat. Ez önmagában nem baj. A gond az, hogy az iktatószám az alárendelt egyedek tucatjaiban is előfordul, még hozzá az egyikben csak az X, a másikban csak az Y változatban. Viszont a modellben nem lehet utalni arra, hogy ilyen felépítés esetén az iktatószám ide, olyan felépítés esetében pedig oda kapcsol. Hogyan lehet ezt a kapcsolati problémát okosan feloldani?

19.4.6 A tantárgy modulokból áll. A modulnak a tervező a következő összetett azonosítót jelölte ki: Tantárgyszám+Modulsorszám. A feltételezések szerint egy modul több tárgyban is szerepelhet. Milyen többszörös hibát követett el a tervező?

19.4.7 Az ESZKÖZ egyedét számos tulajdonság jellemzi. A tervező ezeknek csak a nevét adta meg. Semmilyen megjegyzést nem találni arra nézve, hogy mi a tulajdonságoknak a ... A miye? A tulajdonságoknak arról a lényeges metajellemzőjéről van szó, ami mutatja a tételnek a modellben betöltött feladatát. Mi ez a metajellemző?

19.4.8 Az egyed neve: NAPMEGOSZLÁS. Definíció: Hogyan oszlik meg a rendelkezésre álló időkeret az egységek között. Attribútum: NAP - és semmi más. Mi a hiba?

19.4.9 Az előző példa kapcsán a tervező arra hivatkozott, hogy ő azonosítási módként a „kapcsolattal való azonosítás” megoldását választotta. Tehát a fölérendeltek azonosítják az alárendeltet. Ezzel a kijelentéssel nem javított, hanem rontott terve minőségén. Miért?

19.4.10 Az egyed neve: AFAERV. Definíció: Megmutatja, hogy egy adott időszakban az adott termékjegyzékhez melyik ÁFA-kulcs tartozik. Tulajdonságok: Érvényesség kezdete, Érvényesség vége. Mi ennek a tervrészletnek a szépséghibája? Az előző feladathoz képest ez a modellrész még pontatlanabb. Miért az?

19.4.11 A HALLGATÓ egyed azonosítója: Iktatószám+Ügyfél neve. Hány hibát tartalmaz ez a modellrészlet?

19.4.12 Ugyanebben a modellben (!) a BEADVANY azonosítója: Ügyfél azonosítója+Iktatószám. Az előbbi hibákat a tervező még kettővel tetézte meg. Mi a két újabb hiba?

19.4.13 A KIKÜLDETÉS azonosítója: Kiküldetés megnevezése. Mi a kettős - az általános és a speciális - hiba akkor, ha a kiküldöttre is gondol?

19.4.14 A SZERZŐDÉS azonosítója: Iktatószám. A fentebbi példákat is nézve milyen újabb probléma vetődik fel? Milyen elvet sértett meg a tervező?

19.4.15 A MODUL azonosítója: Iktatószám+Tanfolyamszám+Modulsorszám+Megtartás sorszáma. Ez az azonosító túldefiniált. Miért az? Gondoljon az egyedelőfordulásokra!

19.4.16 Ez a példa az előző folytatása. A TANFOLYAM kulcsa ekkor már Partnerszám +Tanfolyamszám. Milyen súlyos problémát lát akkor, ha az előző példára is figyel? A MODUL a tanfolyam része. Kapcsolható-e a két egyed típus, ha ilyen az azonosítójuk?

19.4.17 Ön szerint mi a HELYSÉG egyed célszerű azonosítója? Számításba jöhet az irányítószám, a települések KSH-törzsszáma és esetleg egyéb adat is. Gondoljon a helységek lehetséges körére és ennek megfelelő módon adjon választ!

19.4.18 A Kötvényszám részeként határozták meg a Körzetszám adatot. Ez hol egy személyt, hol egy bejárati utat, hol egy valódi körzetet jelent. A körzet adott esetben lehet több utca is, máskor viszont csak egy. Milyen többszörös problémát lát ebben a tervben? A Kötvényszám tulajdonságot azonosítónak is használják.

## 19.5 Teljesség és minimalitás

**Célkitűzés.** Az egyedek és tulajdonságok viszonyokat alkotnak. Egy egyedhez több tulajdonság kapcsolódik - és megfordítva. Az előző esetben beszélünk az *egyed belső szerkezetéről*. Ennek optimalizálását többnyire matematikai alapon - normalizálással - szokták elvégezni. Azonban a normalizálás bizonyos teljességi és minimalitási hibák feltárására nem alkalmas. Ezen túlmenően a normalizálást igazából csak az érti meg, aki amúgy is érzi a címben jelzett két követelmény fontosságát és felfogja azok mibenlétét. Ehhez nyújt támogatást ez az alpont.

**A megoldáshoz.** A fentieknek megfelelően az olvasó ne pusztán normalizálási alapokon gondolkodjék, mert néhány feladat nem oldható meg matematikai eszközzel. Inkább vegye elő a józan eszt.

19.5.1 Egy egyed típusban együtt láthatók a Személyi szám, Születési dátum és Személy neve adatok. Folytassa az alábbi mondatokat:

- a) Ez a megoldás rossz, ha ...
- b) Ez a megoldás nyilván redundáns, de mégis jó, ha ...

19.5.2 Az előbbi példa b) megoldásának az esetében a tervezőre egy további feladat hárul. Egy eljárást kell ír(at)nia. Mi annak a lényege?

19.5.3 A munkaviszony egyedben együtt szerepel a Munkakörkód és a Feorkód adat. A definíció szerint a munkakörkód egyik összetevője a FEOR 1-4. jegye. Hibás-e a megoldás és tehet-e valamit a tervező?

19.5.4 A tantárgy egyedben együtt szerepel az Elméleti óraszám, Gyakorlati óraszám és Óraszám hármas. Az utóbbi az előbbi kettő összege. Ön szerint redundáns-e az egyed? A tervező azzal érvel, hogy a tantárgy tervezési fázisában még nem ismeretes az elméleti és a gyakorlati órák megoszlása, de az összóraszám már adott.

19.5.5 A pénzügyekkel kapcsolatos egyedekben sokszor szerepel a nyitás, mozgás\*, zárás adatsor, ahol a „\*” a többféle mozgást jelöli. A zárás nyilván redundancia, mert a többi adatból kiszámítható. Ezt az átfedést elviseljük, azonban az időfaktor miatt egy másik gonddal is kell számolni. Milyen redundanciát okoz a január havi zárás vezetése és miről kell gondoskodnia a tervezőnek?

19.5.6 Az egyedben három dátum található: Támogatás tervezett kezdete, Támogatás tervezett vége, Támogatás kezdete. Formailag nincs is ezekkel semmi gond. Egy tervezési hibát mégis felfedezhetünk akkor, ha az egyedben nincs negyedik dátum. Mi az?

19.5.7 Az ellenőrzés egyedben található az Ellenőrzéskor foglalk. létszám nevű adat. Ez azt mutatja, hogy a vállalkozás hány főt alkalmazott az ellenőrzés időpontjában. A vállalkozásnál vezetik az alkalmazottak létszámát. A kérdés az, hogy normalizált-e a modell és jó-e az adatnév?

19.5.8 A diszponálás több tételben történik. A diszpotétel írja le, hogy az összes rendelt mennyiségből egy-egy alkalommal mennyit írtak ki szállításra. A diszpotételben vezeti a tervező az Eddig összesen kiadott adatot is. Helyesen jár el?

19.5.9 A szerződés egyedben található a Központ címe nevű adat. Azt nem tudni, hogy kinek a központjáról van szó, a partneréről, vagy a mi fölöttes szervezeti egységünkéről? A megoldás mindkét esetben rossz. Miért az?

19.5.10 Vannak bizonyos adatnevek, amelyek láttán a gyakorlott elemző szinte azonnal megérzi a bajt. Alább mutatunk néhány ilyen nevet. Milyen baj léphet fel ezek esetében?

típus, jelleg, sorszám, dátum, mennyiség, összeg, leírás, név stb.

19.5.11 Az előző példában az aluminösítés okozta a bajt. A túlminösítés szintén elemi hibákra hívhatja fel a figyelmet. Miben áll e hibák lényege? Pár példa segíti a megoldást:

- a munkát végző dolgozó neve
- a családi pótlékot kérő személyi száma
- a pályázó lakcíme.

19.5.12 Az egyedben szerepel az A-díj, B-díj, C-díj és Díj-összes adat. Hibás-e ez a tervrészlet?

19.5.13 A CASE-ek által nyújtott keresztreferencia kiválóan alkalmas a normalizálatlan egyedtípusok feltárására. Miről árulkodik az alábbi tulajdonságlista-részlet:

‘Név: Képzési díj. Előfordulása egyedekben: Tanfolyam, Hallgató’.

19.5.14 A tervező azzal érvel az előző példa kapcsán, hogy egyes hallgatók nem fizetik a teljes díjat, mert kedvezményt kapnak. Tehát nem feleslegesen vezetik az adatot mindkét egyedben. Miért kétszeresen rossz az érvelése?

19.5.15 Az adatelem megnevezése: Sorszám. Előfordulása egyedben: - itt egy igen hosszú lista következik. Ennek alapján ön mire gyanakodna? Nem képzelhető el, hogy a modell mégis hibátlan? Lásd a következő feladatot is!

19.5.16 Miért tökéletesen rossz - hiányos - a mai CASE-ek által szolgáltatott tulajdonság kereszt-referencia? Gondoljon az előző példára! Minek alapján dönthető el az, hogy egy tulajdonság több egyedben is jogosan szerepelhet-e vagy sem?

19.5.17 Az adatelem megnevezése: Partner neve. Előfordulása egyedben: Rendelés, Számla. Milyen kettős hibát takar ez a részlet?

19.5.18 Mikor szerepelhet a SZEMÉLY-ben a Gyermekeinek száma adat? Gondoljon arra, hogy egyes személyek lehetnek a mi dolgozóink, mások viszont külső partnerek! Ennek megfelelően kettős választ várunk.

19.5.19 A tervező a CIKK egyedtípusba felveszi az ÁFA-% nevű adatot. Ezzel kettős szerkezeti hibát is elkövet. Terve normalizálatlan. Miért az és mi a helyes megoldás? Melyik egyedet jellemzi a kérdéses adat?

19.5.20 A modellben az állományok leírásánál szerepel a Tablespace adat. Itt nem az a baj, hogy ilyen technikai adatnak nincs helye a fogalmi modellben. A fizikai adatbázisban is helytelen ez a megoldás. Miért az?

19.5.21 Az ÁLLÁSKERESÉS egyedben a Szakmakód adat mellett szerepel egy jelző is, aminek Hiányszakma a neve. Jó ez a megoldás? Van-e jobb, az időfaktorra is gondolva?

19.5.22 A HATÁROZAT egyedben szerepel az Utolsó határozat dátuma adat. Milyen hibát követett el a tervező?

19.5.23 A MUNKAKÖR egyedben szerepel a Pályakezdő megfelel-e? adat, ami az igen és nem értékeket veheti fel. Jó helyre szánta a tervező ezt az adatot?

19.5.24 Az egyedben Szamert(n) adatsorozat található. A név csacsí, de tudható a definícióból, hogy számított értékekről van szó. Le kell-e vágni külön egyedbe ezeket az ismétlődést sugalló adatokat, ha beléphet a Szamert(n+1) adat is? Mi a teendő?

19.5.25 A PÁLYÁZÓ egyedben szerepel a Középfokú végzettség nevű adat, ami csak egy jelző. Azt mutatja, hogy van-e ilyen végzettsége a pályázónak. Ön szerint megfelelő-e ez a tervrészlet?

19.5.26 A JELENTKEZÉS egyedben szerepelt a Középfokú végzettség nevű adat, ami a jelentkező adatsorának az egyik eleme. A tervező hibát követett el. Az első kérdés az, hogy normalizált-e a vonatkozó egyed. A második az, hogy megfelelő-e a megoldás. Lásd az előző feladatot is!

19.5.27 Egy értékesítési adatmodellben a Mennyiség adatban ilyen értékeket találtunk: '115 kanna', '1313 üveg', '3 hl' stb. Milyen kettős hibát követett el a tervező?

19.5.28 Létezhet-e a modellben ilyen nevű adat: Megbízott pénzüintézet neve? Egyelőre ne törődjön azzal, hogy lehet-e a modellben külön PÉNZINTÉZET nevű egyed!

## 20. KAPCSOLATMODELLEZÉS

A tulajdonságok és az egyedek modellezésével összefüggésben számos probléma merül fel. Azonban messze nem annyi, mint amennyi a kapcsolatmodellezés terén. Ennek az általánosan tapasztalható jelenségnek két oka van. Egyrészt a mai adatkezelő rendszerek semmit sem tudnak kezdeni a kapcsolattípusokkal. Azokat nem a nevükön át expliciten, hanem csak a kapcsolótulajdonságokon keresztül impliciten kezelik. Így a logikai szintű adatbázistervben a kapcsolat már nem is szerepel mint fogalmi szintű tényező. Másrészt éppen emiatt a tervezők nem fordítanak kellő gondot a kapcsolatok kialakítására.

Ebben a hanyagságban nagy szerepük van a CASE-eknek is, amik emberinek aligha nevezhető módon fogalmaztatják meg a modellezővel a kapcsolatokat. Lássunk csak pár példát. Az alábbiak részletek egy úgymond adatmodellből:

„Minden egyes SZERVEZETI EGYSEG *Munkahelye* (esetleg)  
egy vagy több ALKALMAZOTT (-t /-nak /-ból /-ra).  
Minden egyes SZEMELY *Szereplő Mint* (esetleg)  
egyetlen egy ALKALMAZOTT (-t /-nak /-ból /-ra).  
Minden egyes FEORKOD *Azonosítója* (esetleg)  
egy vagy több KERESSETT SZAKMA (-t /-nak /-ból /-ra).  
Minden egyes ALKALMAZOTT „*entitáshoz tartozik*” (mindenképpen)  
egyetlen egy EGYEB ALKALMAZOTT (-t /-nak /-ból /-ra).  
Minden egyes SZ KOD *Azonosítója* (esetleg)  
egy vagy több KERESSETT SZAKMA (-t /-nak /-ból /-ra).”

A hihetetlenül bugyuta megfogalmazásokkal ne is törődjünk, van éppen elég komolyabb gond is. Ad 1) A viszonynak nincs mindig neve (az „entitáshoz tartozik” nem név). Ad 2) A nevek lehetnek egyezők és nem-beszélők („Azonosítója” - ami ráadásul elvi hiba is.) Ad 3) Az opcionalitás („esetleg - mindenképpen”) csak felülről adható meg, alulról nem. Ad 4) A kapcsolatokkal („egy vagy több - egyetlen egy”) még nincs baj, viszont nincsen lehetőség az altípus explicit kifejezésére. A személy-alkalmazott viszony 1:1 fokú, amiből még nem következik, hogy az alkalmazott a személy altípusa. Ezért nem lehet tudni, hogy az alkalmazott specializáció vagy pedig részletezés akar-e lenni. Ad 5) A visszamutatás kifejezésére egyáltalán nincs lehetőség.

A fentiek következtében a CASE-ekkel készített kapcsolatlistákkal nem megyünk sokra. Arra azonban még ezek a kreálmányok is alkalmasak, hogy az egyedek külső struktúráit mintegy keresztellenőrizzük. Ebben a szükségszerűen rövid fejezetben olyan példákat ismertetünk, amelyek kétféle hibát takarnak. Az egyikféle esetben magának a viszonynak a meghatározása tökéletlen. A másikféle esetben a rossz kapcsolatmegadás arra utal, hogy a tervező a kapcsolt jelenségeket - magukat az egyedeket - modellezte rosszul.

### 20.1 Rossz kapcsolatmegadások

**Célkitűzés.** Léteznek olyan adatmodellek, amelyekben a kapcsolatok száma lényegesen meghaladja az egyedekét. Ilyenkor az a sejtésünk, hogy vagy sok a tranzitív viszony, vagy a tervező párhuzamos kapcsolatokat határozott meg. Ez azt jelenti, hogy a viszonyokat nem

generalizálta. Például a személy és a vizsga között létrehoz egy vizsgáztató és egy vizsgázó kapcsolatot is, ami teljesen felesleges, mert a vizsgában amúgy is megtalálható a részvétel minősége. A fentiek miatt keresni és irtani kell a szükségtelen kapcsolatokat.

**A megoldáshoz.** A nem felesleges kapcsolatok megadása többféleképpen lehet rossz. Lásd a bevezetést. Tipikus keresendő hiba a fok és az opcionáltság rossz kijelölése. Igen sokszor elmarad a kapcsolatok közötti (kizáró, bennfoglaló, metsző) viszonyok megadása is. Jellegzetes hiba, hogy az implicit altípushierarchia túlzottan mély. Az is gond szokott lenni, hogy e hierarchiában nem a legfelsőbb elemre mutatnak, amikor kellene és lehetne, hanem egy alacsonyabb szintre hivatkoznak. Például a szerződésben szerződő félként nem a szervezetre vagy annak intézmény specializáltjára, hanem a partnerre kellene utalni.

20.1.1 Egy teljesen általános hibát említünk elsőnek. Ha a CASE-eszközzel kapcsolatokat is megadnak és több „adatmodellt” alkotnak, akkor a CASE kapcsolat-részlistákat ír ki. Egy-egy részlistán minden olyan kapcsolattípus megjelenik, amelyben az „adatmodellbe” felvett egyed-típus alá- és/vagy fölérendeltként szerepel. Így egy kapcsolat akár az összes részlistán is feltűnhet. Mi az elméleti hiba lényege?

20.1.2 A szervezeti egységek az alkalmazottak munkahelyei. A személyek közül némelyik alkalmazott. Az alkalmazottak között vannak egyéb alkalmazottak. Miért nem tetszetős ez a modellrészlet? Gondoljon a külső foglalkoztatottakra és a hierarchia mélységére.

20.1.3 A képzési program a KIFEJLESZTŐJE nevű kapcsolattal kötődik a fölérendelt szervezeti egységhez is és a szintén fölérendelt szektorhoz is. Viszont a szektor az egység kötelező 1:1 fokú alárendeltje. Helyes-e a tervrészlet?

20.1.4 Az ügyfél és az iskolai végzettség a keresett szakma fölérendeltje. Úgy tűnik, hogy a tervező nem gondolta át a valós összefüggéseket. Miért látszik úgy?

20.1.5 Minden egyes szervezeti egység egy vagy több szervezeti egységből épül fel. Ez a tervrészlet több hibát is tartalmaz. Mik azok?

20.1.6 Minden egyes alkalmazott *vezetője* (esetleg) egy vagy több szervezeti egységnek. Ez a tervrészlet a 20.1.2 példára is gondolva három hibát tartalmaz. Mik azok?

20.1.7 A képzési program a *bonyolítója* kapcsolatban a szervezeti egység alárendeltje. Ez a viszonymeghatározás esetleg nem célszerű. Miért nem? Lásd a 20.1.3 példát is.

20.1.8 A tanfolyami vizsga egyed fölérendeltje a modul és a tanfolyam. Ugyanakkor az utóbbi egyed a modul fölérendeltje. A megoldás nem feltétlenül rossz (mikor nem az?), de hiányzik belőle egy modellrészlet (mi az?). Egy sajátos korlátra kell gondolni.

20.1.9 Minden egyes tanfolyami vizsgázást megelőző esetleg egy másik vizsgázás. Erre a részletre a tervező egy visszamutató - bár nem ekként minősített - kapcsolatot épített. Ezt egy picit feleslegesen tette. Miért?



20.1.10 A tervező elképzelte az alkalmazott-vizsgáztató 1:N fokú viszonyt. Noha maga a vizsgáztató egyednév nem szerencsés - jobb lenne a vizsgáztatás -, nem ez a gondunk. A fölérendelt körül kell keresni a hibát.

20.1.11 A helyiség az alkalmassági orvosi vizsgálat fölérendeltje. Valahol megvizsgálták a páciens, ebben nincs is semmi hiba. Csak éppen a modellezés lényegét veszítette szem elől a tervező. Miként?

20.1.12 Az ÜGYFÉL CÍM egyed a szerződés kétszeres fölérendeltje egyszer *Tartalmazva*, egyszer *Teljesítője* néven. Ez a modellrészlet katasztrofális. Miért az?

20.1.13 A szerződés az ÁFA-KULCS egyed alárendeltje. Ez bizony kettős hiba. Miért az?

20.1.14 A szerződéses ár egyed a bizonylat tétel fölérendeltje. Helyes ez a megoldás? A bizonylattétel valójában egy számlasort jelent.

20.1.15 Egy cég rendezvényeket szervez. A rendezvényeket típusokba sorolják. Ez a típus az anyagigény egyed fölérendeltje, vagyis a rendezvénytípus bizonyos anyagokat igényel. Apró hiba van ebben a részletben. Mi az?

20.1.16 A termékjegyzék és az ÁFA a termék két fölérendeltje. Köztük nincs kapcsolat. Mi a hiba ebben a tervrészletben?

20.1.17 A USER nevű egyed a bizonylat egyedhez a *Rögzítője* nevű viszonnal kötődik. A USER egyetlen fölérendeltje a USERGROUP. Milyen hibákat tartalmaz ez a részlet?

20.1.18 A 20.2.6 példában is utalunk arra, hogy az alkalmazott alárendeltje a helyiség. Az utóbbi egyed a szervezeti egységnek is az alárendeltje. Az alkalmazottat szervezeti egység foglalkoztatja. Hibás-e a tervrészlet?

20.1.19 Minden egyes értékcsökkenési mód *Hivatkozv* kapcsolattal az állóeszköz mozgás fölérendeltje. A kapcsolatlista másik részében ez található: Minden egyes ECSOMOD *Hivatkozva* ESZKMOZG. Milyen hibát vél felfedezni?

20.1.20 A helyiség az eszközmozgás fölérendeltje. A tervező azt akarta tükrözni, hogy az eszköz melyik helyiségben található. Jó megoldást alkalmazott?

20.1.21 A tervező az eszközökhöz négy részletező jellegű egyedet képzelt el: épület, gép, jármű és tartozékok. Több hibát is elkövetett. Mik azok? Az osztályozásra gondoljon.

20.1.22 Az ÁFA-kulcs érvényesség az ÁFA-kulcs fölérendeltje. Mi a furcsa hiba lényege?

20.1.23 A szervezeti egység a helyiség, az a leltározott eszköz fölérendeltje. Az utóbbi egyed ugyanakkor az ügyfél alárendeltje is. Nem lát itt némi káoszt?

20.1.24 Lásd a 20.1.17 példát is. A tervező a bizonylaton vezeti, hogy melyik alrendszer keretében rögzítették az adatokat. Milyen hibákat követett el? Keresse a tranzitivitást!

20.1.25 Minden határozathoz tartozik egy elutasító határozat. Ez a kapcsolatdefiníció két komoly hibát mutat. Mik azok?

20.1.26 Az éves tervsor meghatározza a havi tervsort. Ez a tényezőmegadás egy értelmi és egy szerkezeti hibát tartalmaz. Melyek azok? Az értelmi hiba általánosítási jellegű.

20.1.27 A beruházási munka alternatív fölérendeltje a szervezeti egység és a vállalkozó. Az előbbi esetben a beruházási munkát a munkaszám, az utóbbiban a szerződésszám azonosítja. Milyen hatalmas elméleti és gyakorlati hibát mutat az alternatív kapcsolat?

20.1.28 Bizonyos ügyeket egy adott egységünk intéz. Más ügyekben több egységünk is közreműködik. Ezért a tervező létrehozta a közreműködés egyedet, ami nem más, mint az egységek családfája. Jó-e a megoldása, ha létezik az ügy/egység viszony is?

20.1.29 Egyes dolgozókat csak időszakosan alkalmaznak egy-egy szervezetben. Ezért a tervező létrehozta az ügyfél (ez lenne a dolgozó) és munkaviszony szüneteltetés (ez lenne a munkakimaradás) kapcsolatot. Hol követett el hibát?

20.1.30 Munkatársaink a partnervállalataink egy-egy csoportjáért felelősek. Ezért létezik az ügyintéző - partnercég kapcsolat. Ez több szempontból is hibás. Miért az?

## 20.2 Rossz egyedmegadások

**Célkitűzés.** Nagyon sokszor csak a kapcsolatok elemzésekor döbbenünk rá arra, hogy maguknak az egyedeknek a meghatározása volt helytelen. Az alábbi példasorozat olyan eseteket mutat be, amelyekben a viszonyok mérlegelése az egyedek átdefinálásához vezet.

**A megoldáshoz.** Ha a viszonyt 1:N fokúnak adták meg és ön M:N fokút feltételezne, akkor hiányzik a kapcsolóegyed. Fordított helyzetben pedig éppen felesleges. Jellegzetes hiba, hogy az altípus felé mutató is-egy kapcsolatot birtoklásnak állítják be vagy éppen megfordítva. Az is előfordul, hogy a visszamutatást hierarchikus viszonyral váltják ki. Ezekben az esetekben az egyedek minősítései - azok korlátai - lettek hibásan megadva.

20.2.1 Minden egyes ügyfélhez esetleg egy intézmény tartozik. Ez a kapcsolat fogalmi értelmezési problémát mutat. Ön szerint mi a hiba lényege?

20.2.2 Létezik az a kapcsolat is, ami szerint minden egyes ügyfélhez mindenképpen tartozik egyetlen személy. Ez a viszony ellentmond az előbbi példának. Miként?

20.2.3 A szervezeti egység az egyéb szervezeti egység 1:1 fokú fölérendeltje. Milyen hibát követett el a tervező?

20.2.4 A főkönyvi számlának kettős alárendeltje a szervezeti egység. Egyszer úgy, mint költséghely, másszor úgy, mint költségviselő. Helyes ez a kettős kapcsolás?

20.2.5 A tanfolyami vizsga a vizsgáztató fölérendeltje. Mivel egy vizsgán több vizsgáztató is szerepelhet, a megoldás látszólag jó. Nem is a kapcsolattal, hanem az egyeddel van bajunk. Melyikkel és vajon miért? Milyen résztvevők vannak a vizsgán a diák mellett?

20.2.6 Az alkalmazottnak alárendeltjeként adták meg a helyiséget. Nem ismerjük ennek a kapcsolatnak a szemantikai lényegét (*a kapcsolatokhoz soha nem adnak meghatározást*). Lehet, hogy az alkalmazott szobájáról van szó. A viszony ettől függetlenül rossz. Miért?

20.2.7 Az ügyfél a pályázat fölérendeltje. Miért hibás ez a modellrészlet? Egy értelmezési gonddal állunk szemben.

20.2.8 Minden helyiséghez feltétlenül tartozik egy és csakis egy idegen helyiség. Viszont tartozik egy vagy több saját helyiség is. Mi ezzel a részlettel a probléma?

20.2.9 A bankszám nevű egyed (rémes egy név) a szerződés fölérendeltje. A tervező nem sok szerződést láthatott. Mire alapozható ez a vélemény?

20.2.10 Az előző példa folytatása: a szerződésnek az ügyfél is fölérendeltje, mint vevő. Az is egyszerez. Lát-e hibát ebben a tervrészletben?

20.2.11 A főkönyvi tételek a főkönyvi kartonsorok fölérendeltjei. Mekkora hiba ez?

20.2.12 A bizonylat tételek a főkönyvi tételek fölérendeltjei. Mennyire súlyos hiba ez?

20.2.13 Milyen hiányra utal, ha az ÁFA-kulcs a termék fölérendeltje?

20.2.14 A terméktípus a terméknek, ez utóbbi a termékváltozatnak a fölérendeltje. Az összes további kapcsolat magára a termékre vonatkozik, vagyis a termékváltozatnak nincs semmilyen egyéb kapcsolata. Ez a szerkezet rész hibásnak tűnik. Miért?

20.2.15 A medium egyed specializáltja a videoszalag, a magnoszalag és a CD. E három al-típusnak nincsenek saját kapcsolatai, ami gyanús. Mit kell ellenőriznie ahhoz, hogy elbírálhassa: jogos-e a specializáció?

20.2.16 A foglalkoztatási viszony egyed fölérendeltje a műszakrend. Hibás-e a kapcsolat?

20.2.17 A mozgásösszefüggés az eszközmozgás fölérendeltje és az eszközátadásokat írja le. Milyen elemi hibát követhetett el a tervező? Gondoljon a sajátos mozgásnemekre!

20.2.18 Az eszköz a tartozék fölérendeltje. Tipikus szemléleti hibát mutat a terv. Mi az?

20.2.19 Minden egyen1 egyedhez tartozhat egyen2 alárendelt egyed is. Milyen komoly hibát követett el a tervező és vajon miért tehette azt? Miért nevezhető lustának?

20.2.20 Gyanússá vált, hogy a modellben tucatnyi ügyiratjelleg - támogatás-x nevű viszony fordult elő. Az x különböző támogatási formákat jelentett. Milyen absztrakció maradhatott el a modellezés során?

20.2.21 A modellben szerepelt N darab támogatás-x - esemény kapcsolat is. Lásd az előző példát! Milyen nemkívánatos jelenséget mutat a struktúra?

20.2.22 A kérelem egyed fölérendeltje a munkáltató és a munkavállaló. A tervező nem élt a szükséges absztrakcióval. Melyikkel és mi a helyes megoldás?

20.2.23 Az adatszolgáltató az adatszolgáltatás fölérendeltje. Ez így helyes, mégis akad egy absztrakciós hiba a modellben. Mi az?

20.2.24 A kérelem egyed alárendeltje a mellékletek és a nyilatkozat egyed. Milyen hibát mutat ez a tervrészlet?

20.2.25 Az iktatófőszám az ügyirat fölérendeltje. Milyen szemléleti hibát takar a modell?

20.2.26 A megye a „telephelye van” kapcsolat mentén a szállító cég fölérendeltje. Több hibát is tartalmaz ez a tervrészlet. Mik azok?

20.2.27 Minden településhez tartozik egy vagy több helység. Ez a kapcsolatdefiníció egy szemléleti hibát rejt. Mi az?

20.2.28 Ha a külsős személy megbízásos munkát végez, akkor az ügyfélnek kell, hogy legyen *egy* adónyilatkozat alárendeltje. Miért hibás ez a terv?

20.2.29 A tervező megálmodta a szervezeti egység - statisztika kapcsolatot. Ez egy picit rossz álom volt. Legalábbis formailag. Miért az?

## 21. TULAJDONSÁGMODELLEZÉS - MEGOLDÁSOK

### 21.1 Egy átfogó példa

21.1.1 A legalapvetőbb modellezési *axióma* így hangzik: „**Az adatmodell - egy**”. Éppen ezért egyetlen szervezetben sem lehet *több* modellről beszélni és *több* adatmodellt alkotni. Aki ezt teszi, az nem érti az adatmodellezés lényegét. Súlyos szemléleti hiányosság, hogy a tervezők az átgondolatlan terveiket a „modell” megjelöléssel illetik. A modellezés kritériumokra figyelő tervezést jelent. Erről a példa esetében nyilván szó sem lehet.

21.1.2 A „modellek” kifejezés azt sejteti, hogy kísérlet sem történt arra, hogy az ismereteket magas szinten - értsd: az alkalmazások között is - egyeztessék. Ezért az „adatbázisok” fogalmi/tartalmi, szerkezeti és formai értelemben összehangolatlanok. Azért azok, mert elmarad a legfontosabb lépés, az **integrálás**. Vagyis pontosan az, ami minden modellezésnek az alapvető lényege.

21.1.3 A helyes válasz: c) nem lehet tudni. Magyarázat: Két tulajdonság akkor és csakis akkor azonos, ha

- azok fogalmi lényege megegyezik (ez esetünkben feltehető);
- azok neve megegyezik vagy explicit módon jelzett szinonimákról van szó (esetünkben ez a kapcsolat nem történt meg);
- a tulajdonságok értékkészlete azonos (erről nem tudunk semmit);
- a természetes ábrázolása azonos (erről sem tudunk semmit).

Ha például a Munkanélküli Anyja neve 32 karakteres, a Mnélk Anyja neve 40 karakteres adat, akkor a kettő már nem azonos lényeg!

21.1.4 A kettős név arra utal, hogy még egy alkalmazáson belül sem történt meg a lényegek egyeztetése. Ezzel a tervező két elvet is megsértett. Egyrészt adatmodellje **nem egyértelmű**, másrészt az azon alapuló adatbázisa nyilvánvalóan **redundáns**. Mindez arra utal, hogy nem történt meg az adott esetben az állományintegráció.

21.1.5 A helyes válasz: b) egyetlen egyben. A modellben minden valós jelenséget - így a személyt is - csakis egyetlen egyedben tükrözzünk. Ez követeli meg tőlünk a minimalitás elve. Ezért az Anyja neve tulajdonság - ami a személyt jellemzi - kizárólag az egyetlen SZEMÉLY egyedhez köthető. Ezért az ilyen egyszeres, tipikusan leíró tulajdonságok modellezésén szinte nem is kell gondolkodni. A d) választás szándékosan megtévesztő. Természetes, hogy egy adat többféle kimeneten is megjelenhet. Viszont a modellezés idevágó alapelve szerint „**A kimenetek nem képezik az adatmodell részét.**”

21.1.6 A helyes válasz: c) általában nem. Igen ritka az olyan feladat, amikor egy ismerettárban csakis és kizárólag a vérszerinti szülőt kell megjelölni. Ám ilyen helyzet sem elképzelhetetlen. Ekkor viszont nem elegendő az Anyja neve hivatkozás, hanem az annál pontosabb Édesanyja neve megnevezést kell alkalmazni.

21.1.7 A helyes válasz: c) sok is, meg kevés is. A definíció magyarázatra, nem pedig a név megismétlésére való. Ezért a kérdéses leírás - sok. Ha meghatározást látunk, akkor attól többletet várunk. Például az adott esetben egy olyan magyarázatot, hogy az Anyja neve korlátozódik-e a természetes szülőre vagy sem. Ezért a vonatkozó definíció - kevés.

21.1.8 A szinonima jelenségről van szó. Ez az **egyértelműség** elvét sérti. Az elv csak akkor nem szenved csorbát, ha a szinonimák kontrolláltak, vagyis explicit módon kijelentik róluk, hogy egymás névváltozatai. Lásd a következő példát és megoldást is!

21.1.9 Az első válasz: b) Igen, ha kontrollált szinonimákról van szó, azaz valahol rögzítjük, hogy a két név ugyanazt a dolgot jelenti. A második válasz a) Nem, mert ez az adott esetben nem történt meg.

21.1.10 **Technikai szinonimáról** akkor van szó, ha ugyanazt a jelenséget elírás vagy az írásmód különbsége miatt lényegében azonos, de formailag mégiscsak eltérő nevekkal jelöljük. Példánk esetében ilyen volt az Anyja neve - elírt - megjelölés.

21.1.11 A nem kontrollált szinonimák - közöttük a technikaiak is - mindig **redundanciát** rejthetnek. A rejtett, nem szándékolt redundanciákat pedig nem lehet feltárni, ha a nevek írásmódját nem szabályozzák konvenciókkal. Ezért fontosak az írásmód formai szabályai.

21.1.12 A helyes válasz: Nem. Gondolja meg, hogy hány kezdőbetű alatt kell keresni az anyára való utalást!

21.1.13 A helyes válasz: a) Nincs. Ezt a választ a következő kérdésekre adott feleletek igazolják.

21.1.14 Az adatmodellben is be kell tartani a **helyesírási** szabályokat. Az 'Anyjaneve' egybeírás éppen úgy helytelen, mint az 'Anyja Neve' hivatkozásban a második nagybetű alkalmazása, hiszen egyetlen fogalomról - és nem kettőről - van szó. Így csak az első tagot lehet nagybetűvel kezdeni. Ezért az egyetlen helyes megoldás az 'Anyja neve' írásmód lenne. Megjegyzés: Tekintettel a rendezési sorokra is, ne tessék alábecsülni a kis- és nagybetűk ad-hoc használata által okozott technikai problémákat!

21.1.15 Feltehetőleg az anya *nevéről* van szó. De csak esetleg! Bizonyos helyzetekben magát az anyát is le akarjuk írni ismeretekkel. Ekkor az 'ANYJA' lehet az arra való hivatkozás - tehát *azonosító* - is. Ezért a félreértések elkerülése végett a névnek mindig pontosan tükröznie kell a tartalmat, tehát az adott esetben le kell írni a „neve” szót is!

21.1.16 A két be nem tartott elv az **érthetőség** és az **egyértelműség**. Ugyanebben a rendszerben anyagokat is ismeretekkel akartak leírni. Nincs semmi akadálya annak, hogy az Anyag megnevezése korrekt név helyett valaki szintén az 'ANYNEV' rövidítést alkalmazza. Ezzel máris előáll a homonima jelenség. Tehát az 'ANYNEV'-ről egyrészt nem lehet tudni, hogy minek a megnevezése (nem érthető), másrészt pedig több dolgot is jelölhet (nem egyértelmű).

21.1.17 A megbízott egy idő múlva maga is lehet munkanélküli - vagy éppen fordítva. Ezért a **felesleges minősítés** mindig feloldhatatlan zavarokra vezet, ha az időfaktor is lényeges tényező.

21.1.18 Általánosan elterjedt rossz gyakorlat, hogy a neveket **programozói szemléletben** alkalmazzzák. Vagyis a tulajdonságokat a tartalmat befogadó állomány nevének a kezdőbetűjével/betűivel minősítik. Ez az úgynevezett „emlékeztető” név humbug. Először is csak a programozót emlékezteti. Másodszor már őt sem, hiszen számos állomány neve kezdődhet „E” betűvel, ezért az ‘Eanyanev’ még mond valamit, de a ‘Ganyanev’ - az „eGyéb” állományra utaló - már semmit sem. Másrészt az ilyen rövidítés nevetséges megjelölésekre vezethet. Gondoljon például a szerződéses árra vagy a ‘Banyanev’ névre! Harmadrészt a minősítés igen megnehezíti az elemzést, például a szinonimák feltárását.

21.1.19 Sok tervező az adatmodellbe a meglévő *fizikai* állományokban található és/vagy a *papírokon* leírt adatmezők megnevezéseit veszi át tulajdonságnévként, ami számos hiba forrása. Ez a gyakorlat arról árulkodik, hogy a modellezésből tökéletesen kimaradt a legfontosabb lépés: az **elemzés**. A felvethető problémák a következők:

Ad 1) Formai: Ki tudja fejben tartani, hogy mi is az az M109? Ez az adatnév úgy beszél, hogy közben hallgat. Ad 2) Elvi: A modellben nem nyilvántartásokon vezetett adatokat, hanem fogalmakat tükrözünk. Ezért az egyetlen és tiszta - nem homályosan minősített - Anyja neve tulajdonságot szabad csak modellezni. Azt pedig másutt kell leírni, hogy ez a tulajdonság mely nyilvántartásokban fordul elő. Ad 3) Technikai: A rosszul minősített nevek a szinonimák - és azokon át a rejtett redundanciák - feltárásának a legjelentősebb gátjai.

21.1.20 A tulajdonságnevek az egyedekről is árulkodnak. A vonatkozó két név azt sejteti, hogy a modellben nem gondolták át a személy mint jelenség lényegét. Nem gondoltak például arra, hogy az ügyintéző maga is lehet segélyezett. Ezért a szinonim nevek sokszor nemcsak egyes adatok, hanem teljes állományok redundanciáját is sejtetik. És valóban: az adott esetben külön ÜGY-INTÉZŐ és SEGÉLYEZETT állományokkal találkozhattunk.

## 21.2 A tulajdonság lényege

21.2.1 A tételek közül csak a cikkszám alkalmazása elfogadható. Az adatmodellezésnek nem az a feladata, hogy a papírokon fizikailag megjelenő jelsorokat, előrenyomtatott szövegeket stb. leírja. Az szintén nem engedhető meg, hogy tulajdonságként tüntessenek fel olyan tényezőt, ami valójában egyed (lásd Anyagrendelés).

21.2.2 A 21.2.1 feladat első mondata tulajdonságtípusokról szólt. A típus feltételezi az előfordulást, vagyis tulajdonságoknál az **értékkészletet**. A pecsétnek, aláírásnak stb. nincs értékkészlete, mint ahogyan a ‘saját vállalatunk címe’ valaminek sem. Mivel ezek nem típusok, nem is modellezhetők.

21.2.3 A papírokon lévő aláírások a személyek behatárolására szolgálnak. A gépeken az ügyintézőt nem az aláírás, hanem az azonosítója határolja be. Az adott esetben a tervező erről az azonosítóról elfeledkezett. Így tehát a modellje tartalmaz egy felesleges dolgot (aláírás), viszont ugyanakkor hiányos is (nincs azonosító).

21.2.4 Ha mindenáron akarják, akkor az ügyintéző egyedben - tehát egyetlen helyen - elképzelhető a telefonszám modellezése. Az ügyintéző által ügyintézett papírokhoz kapcsoltn az adat nem modellezhető, mivel az óriási redundanciára vezet. Egyébként az ennyire változékony adatot nem szoktuk adatbázisba tenni. Azért nem, mert senkinek sem lesz arra energiája, hogy a telefonszám-változásokat átvezesse. Ha pedig azt nem teszi, akkor az egésznek nincs értelme. Ott van a telefonkönyv!

21.2.5 Az életkort általában nem modellezzük akkor, ha a személyekről tartósan vezetünk ismereteket, hiszen ezt az adatot nem fogjuk minden születésnapon karbantartani. Az életkor csak akkor modellezhető, ha a személy azonosítása szükségtelen, tehát például statisztikai felmérésekben. Egy dolog bizonyos: a kérvényben nem szerepelhet az életkor. Azért nem, mert a kérvény vagy csak egy formális papír, amit nem modellezünk, vagy tényszerű adatokkal leírandó valós egyed, amit viszont a személyhez kell kötni. Az utóbbinál pedig nyilván a fix születési dátumot alkalmazzuk - és nem a változó életkort.

21.2.6 Az első probléma általános elvi jellegű. A modellben egyáltalán nem tükrözzük az úgynevezett *technikai adatokat*. A második gond is generikus, de technikai természetű. A modellben nem egy programrendszer kulcsszavainak megfelelően írjuk le a dolgokat, hanem érthető magyar *fogalmakban*.

21.2.7 Egyrészt a CDV számított, technikai, ráadásul nem is tárolt adat. Másrészt nem állja meg önállóan a helyét, mert nem külön érdemi fogalom. Ezért nem való a modellbe.

21.2.8 Az első válasz: Számos dolognak lehet és van gépi azonosítója, ezért a név nem felel meg az egyértelműség követelményének (potenciális homonima). A második válasz: Bár a modellbe nem vesszük fel a technikai adatokat, e szabály alól kivételt jelent az azonosítópótlék, a mesterséges azonosító. Enélkül ugyanis sok esetben nem volna módunk a kapcsolatok modellezésére és elemzésére.

21.2.9 A papírokon lévő fix szövegeket nem modellezzük.

21.2.10 Ha egy ismeret csak papíron szerepel, akkor jobb, ha ott is marad! Már csak azért is, mert az adott esetben szó sincs tulajdonságról.

21.2.11 Adatkitöltési gond, hogy hová kerül majd az a hozzájárulás, ami pontosan 5 százalékos. Ennél súlyosabb hiba, hogy a tervező *adatértékeket* és nem *adattípusokat* modellezett. Az adattípus a Hozzájárulás mértéke, aminek értéke háromféle lehet a példa szerint. A modellrészlet azért is rossz, mert később ezek a határok módosulhatnak illetve többféle tartományt is felvehetnek. Megjegyzés: Ezt a hibát általában akkor követik el, ha a papíron „kiikszelendő” rovatok szerepelnek. A rossz tervező nem keresi a mögöttes egyetlen valódi tulajdonságot, hanem szolgáian modellezi magát a papírt.

21.2.12 A csatolt papírokat egyáltalán nem modellezzük. Ezért a mellékletek száma adat ez esetben *speciálisan* is felesleges. Ezen túlmenően *általánosan* sem modellezzük az egy fölérendelthez tartozó alárendeltek számát, mert az külön karbantartási igényre vezet.

21.2.13 Ad 1) A pontatlan név miatt még azt sem lehet tudni, hogy az az oldalak darabszámát vagy a kötegen belüli lap sorszámát takarja-e. Ad 2) Az utóbbi esetben a papírok fizikai



elrendezéséhez kötődő adatról van szó, amit csak igen ritka esetben modellezzünk. Ad 3) A név egyébként is aluminósított, tehát homonimára vezet, mert sok mindennek van oldalszáma.

21.2.14 Ad 1) A számított adat tárolása nem elvi, hanem gyakorlati kérdés, az ugyanis mindig redundanciát jelent. Ezt gyakorlatilag a hatékonyság érdekében sokszor el kell tűrnünk. Ezért a kérdéses adat tárolásának nincs elvi akadálya. Ad 2) Mivel a számítás is az ismeretekről szóló tudás (voltaképpen egy sajátos korlát), a számított adatokat éppúgy modellezni kell, mint az alapadatokat. Ad 3) Jelen esetben a megnevezés nem szerencsés. Azért nem az, mert egy adó mértékére - és nem annak fajtájára - utal. Ergo ha a mérték pl. 5.1%-ra változik, akkor baj van.

21.2.15 Ad 1) A név megtévesztő, mert az átlagkereset az adott helyzetben nem számított, hanem alapadat! Ezért modellezésének nincs gátja. Ad 2) A név nem beszélő, nem is magyar (ékezhíány) és a tervezők tipikus hanyagságát tükrözi. A nevekben ugyanis nem lehet betű-írás, karakterfelcserélés stb., mert az megzavarja az elemzést.

21.2.16 Ad 1) Az adott környezetben a Kirendeltség címe nem tulajdonságtípus, mert értéke mindig azonos. Ezért ott nem is modellezendő. Ad 2) Tágabb összefüggésben a vonatkozó adat csak egy egyedet jellemezhet: magát a kirendeltséget. Ezért a sokféle eseményhez, papírhoz kötötten nem modellezhető, mert az redundanciára vezet.

21.2.17 Ad 1) A 'jelenlévő képviselő neve' papírra - és nem adatbázisba - való ismeret. Ad 2) Ha mégis adatbázisba viszik, akkor következtelen. Mert mi van akkor, ha nem egy képviselő van jelen? Ad 3) A „jelenlévő” jelző arra utal, hogy valamilyen eseményről, azaz többszörös jelenségről van szó. Így az adat mindenképpen redundáns.

21.2.18 Ad 1) A technikai műveleti adatok nem tartoznak a modellbe. Ad 2) A kiírandó jelleg nyilván olyan felismerhető állapottól függ, amire az egyed egyéb adataiból lehet következtetni. Tehát ez az ismeret mindenképpen felesleges, redundáns.

21.2.19 Technikai ismeret, ami nem tartozik az adatmodellbe. Ráadásul hibás is, mert egy előfordulásnak nem feltétlenül csak egy állapota lehet.

21.2.20 A 'feltételezett előzetes végzettségek' nem adat, hanem lányregény. Egyrészt többszörös. Másrészt nem sejthető belőle, hogy itt névről, más egyedre való hivatkozásról vagy miről van szó.

21.2.21 A modellben nem szerepelhetnek technikai adatok. Emellett egészen bizonyos, hogy a jelszó nem egyetlen adat, mert annak utalnia kell a tárgyra, a hozzáférési szintre és a műveletre is.

21.2.22 Ad 1) Az 'AF7' túlminősítés, mert a kiállítás mindig azonos dolgot jelent. Ad 2) Mivel a keltezés részeivel semmit sem kell kezdeni, annak szétbontása felesleges. Ad 3) A jegyzőkönyveket egyébként sem modellezzük.

21.2.23 Konstansokat nem modellezzünk az egyedtípusokban. Ha valami mindig N, akkor ezt az értéket minek tárolni és kezelni?

21.2.24 Ad 1) Ilyen módon az adat felesleges, hiszen minden tárolt szerződés élő. Lásd az előző feladat megoldását! Ad 2) A terv nem ettől rossz. Hanem attól, hogy nem feltételezi a megszűnt szerződések tárolását. Tehát nem maga a tulajdonság felesleges, hanem a használati módja nem megfelelő.

21.2.25 Ad 1) Technikai adatokat nem modellezünk. Ad 2) Itt nem adattípusról, hanem értékekről van szó. A 'bevitte' és 'törölte' a Kezelte adattípus két lehetséges értéke.

21.2.26 Nem, mert az kimondottan technikai adat.

21.2.27 Ad 1) Formai hiba, ha a névben definíciószerű részletek is vannak. Ad 2) Ilyen tulajdonságok nem létezhetnek, hiszen egy tulajdonság - karbantartási jog - két értékéről van szó. Ad 3) Technikai adatot nem modellezünk.

21.2.28 Ad 1) A verzió technikai adat, nem szerepelhet a fogalmi modellben. Ad 2) A változatok menedzselése nem így kell, hogy történjen. Az egy helyen őrzött egyetlen verziószám azt sugallja, hogy a régi változatokat törlik - és az hiba.

21.2.29 Ad 1) Az első esetben a meglét vagy kitöltött értéket vagy kapcsolt előfordulást jelent és ekképpen felesleges, redundáns. Ad 2) A második esetben az adat elfogadható, bár a megvalósításban X darab adat helyett inkább egyetlen vektort célszerű használni.

21.2.30 Semmiképpen sem. Nem csak azért nem, mert papíradatról van szó. Azért sem, mert a 'Kapják' után nevek következnek - több is. Ha ezen személyek kiléte valóban fontos a számunkra, akkor nem a neveket, hanem a megfelelő egyedekre utaló hivatkozási adatokat kell alkalmazni. Azt is normalizáltan, vagyis külön egyedben.

21.2.31 Az időtartam a kezdetből és a végből számított adat. Tárolása teljesen felesleges, sőt, káros. Valahányszor a másik két adat valamelyike változik, karban kell tartani. Kivételt képez az az eset, amikor az időtartam előbb ismert, mint a két határoló érték. Ez a helyzet tervezett időtartamok esetében állhat elő.

21.2.32 A jó adatmodellnek mindig része a DÁTUM illetve az ahhoz kapcsolt NAPTÁR egyed típus. A tervező ezekre nem gondolt. Pedig ezek birtokában a göngyölt napok adat vezetése felesleges lenne.

21.2.33 A keltezés helye a legtöbb esetben csak papíradat, ezért nem szerepelhet a modellben. Ha viszont a papír által tükrözött esemény - esetünkben a szerződéskötés - helyének a rögzítése fontos, akkor azt pontosan ennek megfelelően kell modellezni. Vagyis a SZERZŐDÉS egyednek lesz egy Szerződéskötés helye lényegű tulajdonsága.

21.2.34 Ha nem volt sikeres a visszatöltés, akkor mi az adat értéke és azt hogyan lehet megtudni a vissza nem töltött valamiből? Itt egy technikai adminisztrátori üzenetről van szó, aminek nemcsak az adatmodellben nincs keresnivalója, de még az adatbázisban sem.

21.2.35 A részlet duplán is rossz. Ezeknek az adatoknak a tárolásával a tervező nyilván az utókornak akarta megörökíteni a saját igazgatója becses nevét, amire rajta kívül mások bizonyára adatbázis nélkül is emlékeznek. Egyrészt evidens, hogy a névnek nem szabad a többszörös SZERZŐDÉS egyedben szerepelnie, hanem az egyszeres SZEMÉLY-hez kell kapcsolódnia. Másrészt világos, hogy szerződést köthetünk olyan partnerrel is, amelynél - jellegénél fogva - nem is létezik ügyvezetői poszt. Tehát a név megtevesztő is.

21.2.36 Korlátozott műveleti/technikai adatról van szó, ami nem való a modellbe.

21.2.37 Ha valahol annyira elszántsak, hogy magukat a jegyzőkönyveket is adatbázisban rögzítik, akkor léteznie kell egy JEGYZŐKÖNYV egyednek. Ekkor a jegyzőkönyvet teljesen egyértelmű és egyedi azonosítóval kell ellátni és ekkor a Jegyzőkönyv tárgya az adott egyed tulajdonsága lehet. Már az azonosítás problémája miatt is elképzelhetetlennek - és céltalannak - tartom az ilyen adatbázisrészlet.

## 21.3 Tulajdonságnevek

21.3.1 Ad 1) Ez a rövidítés szerencsétlen, mert előbb-utóbb homonimát fog generálni. Például akkor, amikor a tétel azonosítóját is a Tkód névvel illetik a rossz, de következetes névalkalmazás során. Ad 2) A név nem beszél, mert csak az adott egyed összefüggésében lehet sejteni a lényegét. Ámde a tulajdonságokat egyedtől függetlenül is modelleznünk és értenünk kell. Ad 3) A sorszám - nem kód. A név tehát félrevezető, mert kódolási séma szerint kötött értékkészletet várunk mögötte, pedig ilyenről szó sincs.

21.3.2 Az ember nem is hiszi el, hogy valaki ennyire csacsi neveket képes alkalmazni. Az értelmes lény tehát úgy véli, hogy a név mögött valami speciális jelenség van. Pedig nincs más, mint a tervező megfontolatlansága.

21.3.3 A magyar nyelvben alkalmazott ragozások miatt rengeteg technikai szinonima szerepel a terveinkben. A vizsgált modellben a Kezdetének dátuma mellett előfordult a Kezdődátum, Kezdeti dátum, Kezdési dátum stb. megnevezés is. A technikai szinonimák megelőzésére a nevekben kerülni kell a ragozást vagy legalább meg kell állapodni annak közösen alkalmazott módjában.

21.3.4 Nem tudni, hogy hónapról, napról, óráról stb. van-e szó? Ezért a Munka kezdődátuma vagy a Munka kezdési időpontja pontosabban tükrözné a lényegét.

21.3.5 Természetesen a nap is időt jelez, de nem időpontot. Ezért a kezdési időpont megjelölés nem szerencsés, ha az időt csak nap mélységig adjuk meg.

21.3.6 Egyrészt a név csacsog. Az indoklás ugyanis mindig szöveget jelent. Másrészt a „szövegezés” és a „szöveg” nem azonos lényegűek!

21.3.7 Ha két vagy több egymással valamilyen módon összetartozó tényező nevében a megjelölés nem azonos ('Feladat' és 'Felad'), akkor ez nem pusztán félrevezető, hanem az elemzést is gátolja. A tulajdonságlistán már nem egymás mellett fog szerepelni a két név és ez megnehezíti az áttekintést. Emellett technikai szinonimák is generálódhatnak. Például a 'Felad' nemcsak feladatot, hanem pl. feladást is jelenthet.

21.3.8 Tipikusan alulminősített név. Az adott esetben nem árulta el, hogy valójában a tanfolyam kódjáról - és nem például a nevééről - van szó. Ha egy adatot kódolunk, akkor annak nevében mindig célszerű utalni erre a tényre.

21.3.9 Ad 1) A név pontatlan. Nem lehet tudni, hogy az érintett személyek számáról vagy konkrétan az érintett személyekről van-e szó. Ad 2) Utóbbi esetben a név egyedekre utal, vagyis nem tulajdonságnév.

21.3.10 Egyrészt nem definiált, hogy mi is az az 'empír'. Másrészt az mindig kód, tehát a két név technikai szinonima, amelyek közül az egyik nem árulja el a kódolt jelleget.

21.3.11 a) Mivel a FEOR93 tartalma más, mint a régi FEOR-számé, a páros alkalmazása nem hiba, bár a másik adat - sima FEOR - minősítése nem lenne haszontalan. b) A féloldalas minősítés azzal járhat, hogy a terv egyes részleteiben elfeledkeznek a FEOR93 „93” minősítéséről és ezáltal rossz táblához fogják kapcsolni az adott adatot tartalmazót.

21.3.12 A 'HUF' nem típus, hanem érték! Ezért névben nem illik használni.

21.3.13 A név csal, mert nem sorszámról van szó.

21.3.14 Ad 1) A tervező elfelejtett magyarul. Ad 2) A jelző a legtöbbször olyan technikai adat, ami egyáltalán nem való adatmodellbe. Ad 3) A név nem árulja el a tartalmat.

21.3.15 Ad 1) A név különlegesen rossz, mert a Leírás általános név, ami legtöbbször szöveges megjegyzést jelent. Ad 2) Egy név alatt nem illik kétféle tartalmat összevonni. Ad 3) A definíció nem mutatja, hogy mikor van szó összegekről és mikor százalékról.

21.3.16 Az adatelem neve nem utal arra, hogy kódról van szó. Ugyanakkor a név az egyik értékkel egyezik meg, ami megtévesztő. Végül a név sohasem tartalmazhat értékeket.

21.3.17 A terv ellentmondásos. Általános hiba, hogy egy értéksort és annak egy adott részét ugyanazzal a névvel jelölik. Ez pedig kapcsolási gondokra vezet.

21.3.18 Az adatnév nem lányregény és nem rejtvény, tehát nem fogalmazható meg mondatként, például kérdésként. Emellett nem szerepelhetnek benne értékek sem.

21.3.19 Az *aktuális* tanfolyami létszám mindig változik, ezért nem vezethető és felesleges is tárolni, mert a kapcsolódó hallgatók egyedből adódik. Ezért csak a *létszámkeretről* lehet szó, viszont a név nem ezt sugallja.

21.3.20 A tulajdonságtípusnak mindig egyneműnek kell lennie. A „Ha..., ha...” ravasznak hitt programozói megoldásoknak nincs helyük a modellben.

21.3.21 Ad 1) Nem tudni, hogy mit takar a név. Ad 2) Ha egy dologról több mindent mondunk el, akkor nem tehetjük meg, hogy az egyik nevet minősítjük, a másikat nem.

21.3.22 A nevek megtévesztőek. Természetesen az éveket nem fogják kódolni, mint ahogyan a negyedéveket sem. Akkor minek az 'azonosító' szöveg a névben? Az pedig már más kérdés, hogy a két adatot esetleg össze is lehetne vonni.

21.3.23 Erős a gyanú, hogy maguk az alapvető fogalmak nem tisztázottak. A munkakör és a FEOR viszonya éppen úgy nem az, mint a munkaköre és a foglalkozásé. Analóg módon sokszor előfordul, hogy a fejlesztő képzettséget ír, de a végzettségre gondol - vagy éppen megfordítva.

21.3.24 Ön használná-e „Az anyja neve” vagy „Az ÁFA kulcsa” neveket? Névelőt nem szoktuk alkalmazni a megnevezésekben.

21.3.25 Ne a nagyképi angolos (?) megnevezések alkalmazásában éljük ki magunkat.

21.3.26 Az ‘utolsó’ jelző olvastán az ember másikat is várna. A jelzőnek a jelzett dologgal való egybevonása pedig nem szerencsés, mert nehezen fogjuk megtalálni az összefüggéseket a tulajdonságlistán.

21.3.27 A nevekben is törekedni kell a tisztaságra és az egyszerűsége.

21.3.28 Nem. Miért nem fejezte ki a tervező a lényegét, pl. így Személy/Hely jel? A programokban alkalmazott adatnevek hosszán lehet spórolni. Az adatmodellben használt tulajdonságneveken nem.

21.3.29 Magyartalan és nem beszélő név. Emellett ellentmondásos. Ha két lényeg van, akkor két adatot is kell alkalmazni - ezen nem lehet takarékoskodni.

21.3.30 Az akronimákat az adatnevekben is nagybetűkkel illik írni. Különösen akkor, ha tulajdonnevekről van szó. A Pénzügyminisztérium nem nemecsek.

21.3.31 Az adatrénvnek nem illik utalnia az adatnak egy adott papíron elfoglalt helyére. Nemcsak azért nem, mert más papírokon ugyanaz az adat máshol jelenhet meg, hanem azért sem, mert az adott papírt is másféle elrendezésben nyomtathatják ki legközelebb. Ugyanakkor viszont a név nem utal a tényleges tartalomra, tehát lényegét csak az érti, aki ismeri az adott papír mai elrendezését.

21.3.32 Az előző feladattal szemben itt már más a helyzet. A Tételsorszám értékei homogének és nem annyi Tételsorszám(n) adatunk van, ahány tétel van a megrendelésen, számlán stb. Ennek ellenére efféle adatot nem célszerű felvenni a modellbe, mert egy adott nézethez tapadó fizikai sorrendet tükröz és nincs külön fogalmi jelentősége. Pl. az első tételsor időben, fontosságban stb. nem előzi meg a másodikat.

21.3.33 Nem vet jó fényt a modellezőre, ha az általános műveltség hiányára utal az adatmodellje. A média maga is többes szám. A szolgalmi útnak nincs köze a lustasághoz. Az alternatíva pedig mindig pontosan két választást jelent.

21.3.34 A sorszám valami máson belüli lényegre utal. Vö. Tételsorszám. Ezért: Ad 1) Ha a kulcs nem valami máson - pl. a tantárgyon - belüli lényeg, akkor a Modulorszám név megtévesztő. Ad 2) Ellenkező esetben viszont az adat nem lehet azonosító.

21.3.35 A minősített neveknel mindig előre el kell dönteni, hogy milyen legyen a minősítés módja, tehát a minősítő tagot előre vagy hátra írjuk-e. Ezért elvileg az Ezévi támogatás - Múltévi támogatás illetve a Támogatás ezévi - Támogatás múltévi párosok mindegyike elfogad-

ható, de a vegyes minősítés nem. Gyakorlatilag mi a második megoldás mellett kardoskodunk akkor is, ha az picit magyartalannak tűnik. Azért, mert a tulajdonság- és viszonylistákon nem árt az együvé tartozó adatokat együtt is látni.

21.3.36 A tulajdonságok neveiben sohasem szabad utalni azok fellelési helyeire. Azt a metaszótárban kell tükrözni, de ott sem magában a névben, hanem a tulajdonság/forrás viszonyban. (N.B.: A forrás az adat fellelési helye, amiből akár több is lehet.)

21.3.37 Ad 1) A név rossz, mert megengedhetetlen rövidítést tartalmaz; bornírt módon angoloskodik; ékezethibás és nem fejezi ki a lényegét. Ad 2) A meghatározás rossz, mert nem árulja el, hogy pl. hármasként esetén mi a helyzet? Ad 3) Az egész tervrészlet úgy, ahogyan van rossz, mert a modell - egy. Az pedig, hogy annak valamelyik részét kik láthatják, nem fogalmi, hanem hozzáférés-technikai kérdés.

21.3.38 Ad 1) Ha nem mindig következetesen azonos módon rövidít, akkor technikai szinonimákat generál, amik megnehezítik a redundancia feltárását. Ad 2) A rövidítés lehet többértelmű is. Például a 'fogl' jelenthet foglalkoztatást és foglalkoztatottat is. Ekkor technikai homonimák keletkeznek, amik látszatátfedést okoznak.

21.3.39 Vannak olyan közgazdasági, műszaki, jogi kategóriák, amik nem egyértelműek, mert többféle változatuk is létezik. A tervezetben utalni kellene arra, hogy az adott esetben melyikről van szó. Itt például hiba, hogy nem tudhatjuk milyen módon számított létszámra illetve milyen alapú leírási kulcsra gondolt a tervező.

21.3.40 Ad 1) Ha itt valóban százalékos adatról lenne szó, akkor meg kellene adni a számítás alapját és módját. Ad 2) Azonban ez az adat csak jelző: azt mutatja, hogy a hiteltérítés nem egy összegben, hanem százalékban történik. Ezt viszont senki sem sejtetheti a névből. Ezért a definíció nagyon hiányzik.

21.3.41 A terv hibás. Az osztályozó ismérveket - típus, jelleg, fajta, kategória stb. - még viszonylag szűk környezetben is másként értelmez(het)ik. Így ezek esetében a definíció nélkülözhetetlen. Már csak azért is az, mert általában az értékeket is meg kell adni.

21.3.42 Az adott esetben a felhasználók egészen mást értettek adástípuson. Ezért az Adástípus adatnév rejtett homonima amellett, hogy nem is érthető az érintett felhasználók számára. A tervező sokszor kényszerül új fogalmak - itt: adatnevek - bevezetésére. Ezt azonban nem teheti meg a felhasználók legszélesebb körével való egyeztetés nélkül még akkor sem, ha definícióban magyarázza az új fogalom lényegét.

## **21.4 Ami a nevek mögött van**

21.4.1 Ad 1) A telefonszám ebben a modellben nem egy lényeg, hiszen egy tulajdonság nem lehet egyszerre kód- és számkategóriájú is. Ad 2) A tervező nem normalizált, mert nem egy helyen vezette a személyek telefonszámait.

21.4.2 Ebben az esetben csak az értékkészlet árulta el, hogy az adat nem homogén, vagyis nem egyetlen fogalmat takar. Az ilyen vegyes kód magában az adatbázisban sem kívánatos, a fogalmi modellben pedig egyenesen megengedhetetlen. A nem és az életkor ugyanis két eltérő fogalmi lényeg. Ne feledkezzünk el az adott esetben az időfaktorról sem. Arról, hogy a fiú illetve a lány holnap férfi illetve nő lesz.

21.4.3 Tipikus tervezési hiba, amikor az értékkészlet nem egyértelmű és/vagy nem teljes. Jelen esetben nemcsak az 'y ideje regisztrált munkanélküli' hiánya okoz fejtörést, hanem az átfedés is. Hová sorolandó az 'x ideje regisztrált pályakezdő munkanélküli'? Csak a definíció árulta el, hogy a tervező két ismérvet vont egy adatba, ami mindig hiba. Az egyik a munkanélküli állapot időtartama. A másik annak jellege (pályakezdő-e az illető).

21.4.4 A részlet nem teljes. Már tudjuk, hogy mi az adat értéke, ha az illető átképzett. De sohasem tudjuk meg, hogy mi az érték, ha nem az. Lásd a következő példát is!

21.4.5 Az „igennek” a „nem” a párja, nem pedig az üres! Általában vigyázni kell az üres értékek használatára. Annak nem szabad jelentést tulajdonítani, főleg statisztikákban, állapotjelzőkben nem. Az üres, sajnos, ma több szerepet lát el. Lehet valós adat is, de használják akkor is, ha az adat nem értelmezhető vagy nem ismert.

21.4.6 A tervező két hibát követett el. Először: rossz adatnevet alkalmazott és a definíciót csak arra használta, hogy pótolja a név hiányosságait. Másodszor: a meghatározás semmit sem ér, mert éppen a lényegét nem árulja el. Vagyis azt, hogy mit jelent a 'hasznló'.

21.4.7 A 'kód' és a 'természetes' szó ellentmond. Ha egy adat kódolt, akkor az már nyilván nem természetes.

21.4.8 Nem elég, hogy egy név két fogalmat takar, közülük az egyiket még nem is fejti ki. Ugyanis ív nem csak egyféle van.

21.4.9 Az adattípust két szinten kell meghatározni. Az adatoknak van egy természetes, a **fogalmi** szintre tartozó típusa. Például ilyen a szám. Ezen kívül van egy mesterséges, a **fizikai** szintre tartozó típusa. Például ilyen a float, hiszen semmi közöm sincs ahhoz, hogy az adatokat mikor tárolják bizonyos okok miatt lebegőpontosan. A baj tehát az, hogy a CASE-ek fogalmi szintre emelik a saját fizikai adattípusaikat és a modellben a tervező csak erre a fizikai típusra utalhat.

21.4.10 Vagy a tervező volt csacsi, vagy a CASE nem adott kellő támogatást. A szám ugyanis nem mértékegység...

21.4.11 A névnek tükröznie kell a tartalmat. Nálunk a Megyekód értékei közé sorolják Budapestet is, holott az nem megye, hanem város. Itt nem kukacskodásról van szó. Gondolja meg, hogy adott esetben az adatnak csak tényleg megyék felé szabad kapcsolnia. Ekkor vajon mit tesz? Felvesz még egy arra utaló adatot?

21.4.12 A terv rossz, mert az adat sérti a homogenitás elvét.

21.4.13 Az Iktató alszám nem külön tulajdonság! Csak akkor lehetne az, ha a felépítése és az értékkészlete mindig azonos lenne. Ezért a főszám és az alszám külön-külön modellezése hibás - elegendő csak magát az Iktatószámot modellezni.

21.4.14 Ad 1) Az alkalmazott és a foglalkoztatott nem azonos fogalmak, tehát a név és a definíció ellentmond. Ad 2) A név nem utal arra, hogy itt nem magáról a végzettségről, hanem annak csak a szintjéről van szó. Ad 3) A kódok a szintnek ellentmondanak, mert hol iskola-típusra (általános), hol azon belüli fokozatra utalnak. Ad 4) A kód nem bővíthető, jóllehet ma igen sokféle iskola van még az általánosan belül is. Pl. van ahol nincs is általános hetedik, mert az már gimnázium harmadiknak számít.

21.4.15 A részlet hibás. Egy hely nem lehet szakmai gyakorlat, legfeljebb a helyen végezhetnek ilyesmit. A meghatározások sem lehetnek az érthetlenségig magyartalanok.

21.4.16 A tervező mindenkit jól becsapott, még saját magát is. A két névből mások azt sejtethnék, hogy kétféle - pl. ideiglenes/állandó - címről van szó. Ezért bár sorszámokat egyébként is csak ritkán szabad alkalmazni a nevekben, azokat feltétlenül kerülni kell, ha a fogalmi tartalmak nem egyneműek. A tervező saját magát is megtévesztette, mert a modellelemzés nem fogja kimutatni a Cím1 adat kapcsoló szerepét. Ezért hiányozni fog a települések felé mutató kapcsolat.

21.4.17 Lásd az előző feladat megoldását!

21.4.18 A tervrészlet gyakorlatilag helyes, bár a különböző papírokon nem mindig azonos mértékű a helyszűke. Elvileg azonban nem helyes, mert a rövid név nem fogalmi lényeg.

21.4.19 Ebben a változatban a terv már kimondottan hibás, mert redundanciát tartalmaz. A rövid név ekkor már szerkesztéssel képezhető származtatott adat.

21.4.20 A többes szám eleve alárendelt egyed típusra utal. Ráadásul az adott esetben lehet, hogy nem is egyre. Ez azonban nem dönthető el, hiszen a tényszerűen vezetendő ismeretekről semmit sem tudunk meg. Például arról, hogy a gépjárművezetés léte vagy kategóriája fontos-e?

21.4.21 A terv hibás. A definíció léte nem mentesít az érthető nevek használatának a kötelezettségétől. A Nyitóösszeg stb. név nemcsak beszélne, hanem ekként feleslegessé is tenné a meghatározást.

21.4.22 A tulajdonságok értéktartománya (doméjn) mindig egyértelmű kell, hogy legyen. Ezért nem a definíciót kell összevonni „Ha..., ha...” módon, hanem ki kell küszöbölni a homonimákat a nevek minősítésével. Pl.: X határozat típusa és Y határozat típusa.

21.4.23 Itt az iratkezelés egyik részletéről van szó. Ezért nem feltételezhető, hogy a küldőt valamelyik partnerrel kívánják azonosítani. Már csak azért sem, mert a küldő lehet számunkra eddig ismeretlen valaki is. Így a tervrészlet nem hibás. Viszont a nevek következtlen írása miatt arra ne is számítsunk, hogy ki fogjuk tudni keresni azt, hogy milyen küldemények érkeztek egy adott nevű küldőtől.



## **22. EGYEDMODELLEZÉS - MEGOLDÁSOK**

### **22.1 Csoportok és rejtett egyedek**

22.1.1 Ön sem ismeri a Cím1 és a Cím2 lényegét. Állandó és ideiglenes címről van-e szó? Mindkét címet ki kell-e tölteni? Adott esetben a két cím megegyezik-e? Ezek a kisebb bajok. Általában a neveket nem szabad számmal ellátni, mert akkor programfüggő lesz a tervünk: ez a nagyobbik baj. Az már további kérdés, hogy a többféle cím valóban csak maximum kettő lehet-e és nem érdemes-e külön egyedet alkalmazni a címekre.

22.1.2 A programozók gyakran alkalmaznak sorszámozott neveket, ami már önmagában is hiba, amint arra az előző példa utalt. Ennél még súlyosabb baj, hogy a feleslegessé vált ismétlést nem távolítják el a tervből, hanem azt más - a nevének meg nem felelő - célra kezdik el használni, hogy megspóroljanak egy változtatást. Ez káoszra vezet.

22.1.3 Első ránézésre a probléma nem súlyos. A terv ugyan slendrián, mert hol marad belőle a Jogcím2 adat, amit az olvasó joggal várna. Azonban ahol ilyen változékonyak az ismeretek, ott fix számokat nem célszerű alkalmazni.

22.1.4 Számított adatokat ritkán vágunk le külön egyedbe. Azért nem alkotunk külön egyedet, mert az ismétlés csak látszólagos, hiszen az összegek szemantikailag más tartalmúak. Ezt viszont megfelelőbb nevekkal lehetne és kellene tükrözni.

22.1.5 A terv nem kidolgozott. Az Előtanulmányok többes száma arra utal, hogy azokat valamilyen ismeretekkel le kell írni. Tehát az előtanulmány nem tulajdonság, nem adat, hanem külön egyed.

22.1.6 Itt ismét rejtett egyedről van szó. A problémát súlyosbítja a definíció hiánya. Mit jelent a jelenség nevében az „alap” jelző? És vajon az alap/nem-alap munkaköröket együtt kell-e modellezni? Ehhez pedig bizonyára szükség lenne egy külön, a munkakört minősítő adatra, végeredményben pedig egy külön egyedre!

22.1.7 A tervezők sokszor tulajdonságként veszik fel azt a dolgot, ami valójában egyedeket takar. A RENDELÉS egyednek nem tulajdonsága a Számla, ami maga is egyed.

22.1.8 Ha egy ismeret egy egyedben ismétlődik és az egyik tagot minősítik (Jogdat1), akkor a másik nem lehet minősítés nélküli (Jogdat). Ha pedig csak egyetlen tag szerepel, akkor azt felesleges számozni (Jogdat1). A modellnek tisztának és érthetőnek kell lennie.

22.1.9 Az eszköz nem tulajdonság, hanem kapcsolódó egyed.

22.1.10 A baj a felesleges minősítés. Ha nincs pl. tervezett és felhasznált eszköz, akkor a „használt” jelző éppen úgy felesleges, mint a Jogdat1 adatban az „1”, ha nincs Jogdat2.

22.1.11 Valakinek több egyenrangúan magas végzettsége lehet. Melyiket írják be az egyetlen adatba?

22.1.12 A helyzet most kicsit más. Közvetlenül előző munkahely csak egy van. Tehát a megoldás jó. De csak akkor az, ha a még korábbi munkahelyek vezetésére nincs szükség.

22.1.13 Ad 1) Csak a legutolsó dátumot vezetni nem elegendő, hiszen egy napon belül is történhet több változás. Ekkor az ismeret semmitmondó. Ad 2) Az adat „önmeghatározó”. Ha kitöltik, akkor volt módosítás és akkor volt módosítás, ha ki van töltve. Magyarul: Csak az adatrögzítőn múlik, hogy a tényező módosítottként szerepel-e az adatbázisban.

22.1.14 Sokszor előfordul, hogy a csoportot a tervezők hol impliciten (vagyis a tagokat fel nem sorolva), hol expliciten (azaz a tagokat külön-külön) adják meg. Ezzel megsértik a szerkezeti egyértelműség elvét, amiből pl. redundancia fakadhat.

22.1.15 Ad 1) Ha a Kod(n) nevek mögötti tartalmak eltérőek, akkor három külön kifejező nevet kellett volna alkalmazni. Így nem teljesült az érthetőség elve. Ad 2) Ha a tartalmak egy fogalomkörbe tartoznak, akkor a bővíthetőség/változtathatóság elve szenved csorbát, mert ismétlésről van szó és annak száma nem növelhető/csökkenthető rugalmasan. Ezt ugyanis csak külön egyeddel lehet megtenni.

22.1.16 Ezeket az adatokat nem kell ismétlődésekként felfogni. Viszont miért rejtegeti a tervező a lényegét? Miért nem írja ki, hogy januári, februári stb. összegekről van szó?

22.1.17 Ha a vizsga több részből áll és a részeket külön adatokkal kell leírni (ez a jogos feltételezés), akkor expliciten adatcsoportot vagy külön egyedet kellene meghatározni, tehát a Vizsga részei nem tulajdonság. Ha nincs szükség a részek formális kifejtésére, akkor pedig a Vizsgarészek leírása név fejezné ki pontosan a lényegét.

22.1.18 Két eset lehetséges: az Alszámlaszám vagy a Számlaszám része, vagy önálló jelentésű adat. Az utóbbi esetben külön egyedelőfordulásra utal és ekkor nincs hiba a tervben. Az előbbi eset viszont tipikus tervezési hiba. A tervezők egy modellen belül hol egyben, hol tagokra bontva modellezik a csoportokat, ami sokféle zavar forrása.

22.1.19 Ezek a nevek megtévesztőek. A többes szám két dologra is utalhat. Egyrészt arra, hogy az ismeretből több van (egyedelőfordulás). Másrészt arra, hogy az ismeretféléből van több (tulajdonságtípus). A megnevezés egyik esetben sem valóságű. Ha viszont csak leírásról van szó, akkor ezt a névben is tükrözni illenék. Pl. Egyéb feltételek leírása.

## **22.2 Az egyed lényege**

22.2.1 Ez az egyed nem egyed, mert nem akarják ismeretekkel leírni. Azért nem, mert a kérelem informális papír, ami nem is modellezhető.

22.2.2 A címlistákat nem adatbázisból szokták kezelni. Ahhoz rendelkezésre állnak a megfelelő hivatali szoftverek. A CIMLISTA és a viszonyegyed egyébként is kényelmetlen. Egyrészt senki

sem fogja karbantartani őket, másrészt a tételszám állandóan nőni fog, mert a törlés senkinek sem jut majd eszébe. A részlet nem rossz, de teljesen felesleges.

22.2.3 A HAVIJÁRULÉKOK tipikus kimenet, ami olyan számított adatokat tartalmaz, amelyek - az időszakon kívül - nem köthetők semmilyen más érdemi egyedhez. Ez tehát nem is egyed, hanem eredmény. Nem való az adatmodellbe akkor sem, ha számítógépen akarják kezelni.

22.2.4 A feldolgozásközi átmeneti állományok két ok miatt sem tükrözhetők külön egyedként. Egyrészt fogalmilag sem, mivel nem önállóan leírandó jelenségekről van szó. Másrészt az elemzés miatt sem, hiszen az átmeneti állományok teljes átfedésben vannak az érdemi egyedekkel. Ezért a be-, ki- és átmenetek nem tartoznak a fogalmi modell körébe akkor sem, ha annak tényezőivel összefüggenek.

22.2.5 A tábla nem való az adatmodellbe. Itt ugyanis nem egy valós egyedről, hanem egy technikai célokat szolgáló állományról van szó. Egyébként is alapvetően téves felfogás az egyedet a táblával összekeverni, hiszen az utóbbi fizikai szerkezeti tényező.

22.2.6 A modellben fogalmakat és nem rekordokat tükrözünk. A rekord logikai/fizikai tényező és ezért még utalni sem illik rá a modellben.

22.2.7 A megoldás csodás, csak éppen semmi köze sincs az adatmodellhez.

22.2.8 Már csak az hiányzik, hogy a megszűntek mellett felsorolják azokat az egyedeket is, amiket sohasem szándékoztak felvenni... A modell éppen eléggé bonyolult akkor is, ha csak az értelmes dolgokat öleli fel. Ne terheljük az elemzését felesleges tényezőkkel!

22.2.9 A modellezés során - tehát átmenetileg - létezhetnek tulajdonság nélküli egyedek is. Azonban a kész modellben ilyesmi nem fordulhat elő. Lásd a következő feladatot is!

22.2.10 A BÉRKARTON nem egyed. Nem ezt a papírt, hanem az azon szereplő adatok által leírt jelenségeket kell modellezni. Ilyen pedig az adott esetben tucatnyi is van (személy, havi bér, pótlékok stb.). Tanulság: Az egyes papírokon megjelenő ismeretek gyakran több egyedet rejtenek. Az ezek összevonásából képzett ismeretsor nem tekinthető egyednek, mert az egyed és a valós jelenség egymással mindig 1:1 viszonyban áll.

22.2.11 Azért nem, mert a PARAMETEREK nem egyed, hanem egy technikai célú állomány, aminek a leírása nem való az adatmodellbe.

22.2.12 Az egyed típusnak mindig több előfordulása kell, hogy legyen. Kivételt képez az itt nem részletezendő ún. szinguláris egyed. Itt azonban nem arról van szó. A FEJLÉC egyetlen konkrét adatsor, amit nem kell és nem is lehet modellezni. Többek között azért sem, mert elvileg minden más egyedhez kapcsolódik, gyakorlatilag viszont nem. Azért nem, mert nem is azonosítható.

22.2.13 Az adott egyednek egyetlen célja az országok nevének a kontrollálása. Általában a tulajdonságok validálására nem illik külön egyedet tervezni, mert erre a célra szolgál az értéktartomány (domain). Speciálisan ebben az esetben az ORSZÁG egyedet átmeneti állapotnak tekintjük, mert azt feltételezzük, hogy előbb-utóbb akad hozzá értelmes adat is.

22.2.14 A modell méretét csökkenteni lehet ezen felesleges egyed végleges kiirtásával.

22.2.15 Ritka körülménynek számít, ha minden szobában csak egy ügyintéző ül. Az már más kérdés, hogy az egyed teljesen felesleges és nem való adatmodellbe. Azért nem, mert nem lényeges és tartós ismereteket rögzít.

22.2.16 Ahol van TANF1, ott van TANFOLYAM is. Csodának számítana, ha a két egyed nem lenne átfedésben vagy nem lépne fel egyéb zűrzavar. Erre már csak a név miatt is számíthatunk, mert feltehetően egy Clipperes programozó alkalmazta ezt a megnevezést.

22.2.17 A műszaki dokumentáció rengeteg adatszerűen nem tükrözhető tényezőt ölel fel (pl. rajzokat, leírásokat). Ez az ismeretkör nem való az adatszerű modellbe.

22.2.18 Az adatlap az, amit a neve is sugall. Nem egyed, hanem papír, amin adott esetben számos valódi egyedre utaló ismeret található.

22.2.19 A jelenléti ív nem egyed, hanem egy darab papír. Nem modellezhető. Viszont modellezhető a jelenlét. Ez nem kukacoskodás. Nem papírokat, hanem jelenségeket kell modellezni és aki ezt a szemléletet nem teszi magáévá, az mindig a papírokat fogja imitálni az adatbázisban is.

22.2.20 A jegyzőkönyv olyan tétel, amit azonnal ki lehet dobni, mivel a modellezése tökéletesen felesleges.

22.2.21 Itt is arról van szó, hogy a tervező nem a valós jelenséget - a cikket - modellezte, hanem az adathordozón megjelenő adatok együttesét tekintette egyednek. A 22.2.10 példával szemben itt nem több egyedet vont össze, hanem egyet bontott szét részekre. Ez hiba. Tanulság: A nézetek sohasem tekinthetők valódi egyedeknek.

## 22.3 Rossz egyedmegadások

22.3.1 Két jelenség akkor azonos, ha tulajdonságaik nagymértékben egyezők. Teljesen azonosak nem lehetnek, hiszen akkor egyedtypust egyáltalán nem lehetne kreálni. A szóbanforgó modellben helytelenül fogták meg a ...-PÁLYÁZAT egyedeket. Ez három problémát is okoz. Ad 1) Többszörösen jelölik ki a pályázó, szervezet stb. egyedeket is. Ad 2) Ennek következtében a modell rettenetesen bonyolulttá válik. Ad 3) A megoldás mindenféle inkonzisztencia forrása nemcsak amiatt, hogy például a pályázók ismereteit egymásnak meg nem felelő módon vezetik a több pályázó egyedben, hanem amiatt is, hogy az egymást kizáró/feltételező pályázatokat nem lehet együtt látni.

22.3.2 Lásd az előző példát! A biztosító elfeledkezett arról az axiómáról, ami szerint „az adatmodell - egy”. Ennek köszönhető, hogy például az ügyfél nem látja mindig azonosan a saját adatait.

22.3.3 A redundancia és az abból fakadó inkonzisztencia.

22.3.4 Az egyetlen valós jelenség tulajdonságainak a több táblában történő szétosztása a partitionálás. Ezt elérési/hatékonysági/biztonsági okokból szokták alkalmazni. Ezek a szempontok másodlagosak. Nem a fogalmi, hanem a logikai tervezés szintjére tartoznak. Végeredményben az EGYEN1 és az EGYEN2 csak nézet. Ezért rossz a modellrész.

22.3.5 A megoldás rossz. Egyetlen SZERVEZETI EGYSEG-et kellett volna alkalmazni és a Szervezeti szint adattal kellett volna minősíteni az előfordulásokat. A baj az, hogy a központnak éppen úgy vannak pl. dolgozói, mint a kirendeltségnek és az irodának. Ezért a dolgozónál három opcionális kapcsolatot kellene felvenni. Mivel ugyanez a kitétel az eszközökre, beszerzésekre stb. is áll, vagyis mindenféle viszonyt meg kell háromszorozni, mindez rettenetesen elbonyolítja a modellt. Ezen kívül gondolni kell a változásokra is. Sokszor előfordul, hogy egy-egy részleget magasabb szintűvé minősítenek át. Például a „kisfiókból” „nagyfiók” lesz. A jó megoldás esetében ez egyetlen jelző átírását jelenti. A példa szerintiben viszont egyedtörléssel és egyedbevitellel jár.

22.3.6 Igen kivételes esetben altípusról lehet szó, ha a várakozó ügyfelet sok olyan ismeret jellemzi, ami nem vonatkoztatható általában az ügyfélre. Azonban inkább feltételezhető, hogy az ügyfél előforduláshalmazát a tervező oktanul bontotta meg és ezzel problémákat generált. Ha nincs külön ügyfél egyed is, akkor a modell hiányos és a minősítés felesleges. Ha van, akkor a modell redundáns. Továbbá ha az ügyfél megszűnik várakozónak lenni, akkor az előfordulást át kell tenni az egyik egyedtípusból a másikba. A „várakozó” ezért nem külön egyedet, hanem állapotot - vagyis tulajdonságot - jelent.

22.3.7 Nem csak olyan szervezetekkel állunk kapcsolatban, akik támogatást kérnek. Akik mégis ezt teszik, azoknál pedig minden évben elveszik a korábbi támogatás alapja, mivel az eredményeket karbantartják. Ezáltal lehetetlenné válik a támogatás kiadásának az utólagos ellenőrzése. Ergo az egyetlen helyes megoldás az, ha az EREDMÉNYEK-et így, vagyis külön egyedtípusként modellezzük. A tervező tehát azt a hibát követte el, hogy két jelenséget egyetlen egyedbe összevonva modellezett.

22.3.8 A szervezeteknek többszörösen több számlája lehet. Egyidejűleg is lehet több és az idők során is lehet több. Ezért a megoldás akkor sem lenne jó, ha egy adott időpontban minden szervezetnek csak egyetlen számlája volna. Ugyanis változáskor elveszne az a történeti információ, hogy milyen számlán fizetett a cég a múltban, ami pedig törvényesen megőrzendő adat. Tehát az egyetlen helyes megoldás, ha létrehozzák a SZERVEZET SZÁMLÁI egyedet, ami mindkét fajta többszörösséget támogatja. A tervező itt is abban hibázott, hogy összevont két egyedet.

22.3.9 A tervezőnek tudnia illenék, hogy az Átváltás adat nem egyértelmű, mert eltérő pénzügyintézetekben egyazon napon is más-más az átváltási arány. Ezért a DEVIZANEM-nek utalnia kell a pénzügyintézetre is. A tervező helytelenül - nem valósághűen - fogta meg a lényegét. Az egyedtípus előforduláshalmazát a kelletnél szűkebbre szabta. A tanulság az, hogy a modellezés során nemcsak a nevek, hanem az előfordulásokra is ügyelni kell.

22.3.10 A kórház is intézmény, ezért annak altípusaként modellezendő. Ez a megállapítás az intézmény és a szervezet viszonyára is vonatkozik. A szervezet-intézmény-kórház altípus-hierarchiát viszont csak akkor szabad így kialakítani, ha az intézménynek a szervezethez, a kórháznak pedig az intézményhez képest jelentős számú sajátossága van.

22.3.11 Olyan jelenségekről van szó, amelyek ismeretei igen nagy hányadban azonosak. Ezért azokat csak a SZERVEZET altípusaiként szabadna modellezni. A gyanú megerősítésére elegendő a neveket keresni. És valóban, a modellben szerepelt Szervezet, Szállító stb. „neve” adat. Ezek pedig nyilvánvalóan szinonimák.

22.3.12 Ad 1) A tervezett jelleg nem ad okot külön egyedtípusra, mert az csak egy állapotot jelez. Ha a tervezett modulnak vannak a modultól eltérő sajátosságai is, akkor azokat altípusban kell tükrözni. Ad 2) Csak ekkor engedhető meg, hogy két egyed azonosítója megegyezzen. Azért, mert különben a kapcsolati rendszer felborul. Ad 3) Ha nem altípusokról van szó, akkor a MODUL és a TERVEZETT MODUL redundánsan őrzi pl. a modul nevét. Ez ellentmondásra is vezet, ha az M1 azonosítójú modulnak és az M1 azonosítójú tervezett modulnak nem azonos a neve. Ad 4) Ha a tervezett modulból tényleges lesz, akkor komoly áttételi műveleteket kell végrehajtani. Általános tanulság: Az egyetlen egyed előforduláshalmazából nem szabad ilyen módon, az állapot szerint több egyedtypust alkotni. Az állapotokat legfeljebb altípusként modellezzük.

22.3.13 A tervezők sokszor következtelenek az érintett dolgok körét illetően. Az adott esetben hol egy, hol több konferenciában gondolkodtak. Az előbbire utal a hivatkozott leírás, az utóbbira az összetett azonosító. A terv inkonzisztens, ergo rossz. Ha ugyanis egy konferenciáról van szó, akkor KONFERENCIA egyed nem is létezhetne és a SZEKCIÓ egyed kulcsa nem lehetne összetett. Tanulság: Mindig figyelni kell az egyedek érvényességi körére, végeredményben az előforduláshalmazra.

22.3.14 Ha nincs külön SZEMÉLY egyed, akkor az ‘MSZ’ minősítés felesleges. Ennek léte éppen azt sugallja, hogy van másik is, ami viszont alapvető hiba, hiszen redundanciát és inkonzisztenciát okozhat.

22.3.15 Ad 1) A speciális probléma az, hogy nyelvtudása a személynek van és nem a hallgatónak illetve a tanárnak. Mellesleg azt, hogy az alárendelt milyen fölérendeltekhez kapcsolódik nem az egyed meghatározásában, hanem a kapcsolatban kell leírni. Ad 2) Az általános probléma most is a valóság-hűséggel és az előforduláshalmazzal kapcsolatos. A valós jelenségeket nem szűkíthetjük le az általunk látni kívántakra, vagyis a mi korlátos nézetünkre. Mivel a valóság az, hogy az emberek nyelveket beszélnek, nem mondhatjuk, hogy bennünket most ez nem érdekel.

22.3.16 Az önadózási jelleg évről-évre változhat, de nem szüntethetjük meg azt az információt, hogy az előző évben milyen volt a jelleg. Ha az adat a személyt jellemezné, akkor csak egy értéke lehetne, amit évenként nyilván felülírnak. A modell nem teljes, mert nem vezeti a történeti ismereteket.

22.3.17 Itt is a dinamikával van baj. Ha az időt is tekintetbe vesszük, akkor már az adatok ismétlődnek - tehát külön egyedeket kellett volna kreálni SZABADSÁG, BETEGSÉG és EGYÉB TÁVOLLÉT névvel. (Persze e feladat megoldása több változatban lehetséges. Itt csak arra célunk, hogy az időfaktort figyelembe kell venni.)

22.3.18 Ha egy tulajdonságot több névvel jelölünk, akkor a modellben rejtett redundancia lesz, de az csak egyetlen adatot érint. Ha viszont az egyedet jelöljük többféle névvel, akkor többféle egyedet is hozunk létre és ekkor azoknak már számos tulajdonsága lehet átfedő és inkonzisztens.

22.3.19 A minősítést az adott esetben jól alkalmazták. A két jogcímnek nincs köze egymáshoz. A definíciók tisztázhatják azt az apró félreértést is, ami abból fakadhat, hogy nyelvünkben a „fizetés” szó kettős jelentésű.

22.3.20 A kifizetőhelynek is van neve, azonosítója, helye stb. Ott is dolgoznak emberek. Oda is be kell szerezni eszközöket stb. Tehát a kifizetőhely nem más, mint egy sajátos szervezeti egység, aminél a Kifizetőhely jel tulajdonság értéke igen, míg a többinél nem.

22.3.21 A redundancia óriási. Egy valaki egyszerre tartozhat mindhárom egyed típusba! Két dolgot kell megoldani. A DOLGOZÓ-t is SZEMÉLY-ként kell felfogni, szükség esetén altípust alkotva. Az ALKALMAZOTT-ra pedig egyáltalán nincs szükség, mert az ALKALMAZÁS a lényeg. Az alkalmazott név önmagában véve sem szerencsés, mert az a dolgozó fogalmi szinonimája egyes környezetekben. Tanulság: minden valós jelenségnek az adatmodellben csak egy egyedelőfordulás felelhet meg. Ezért nem szabad az egyedeket úgy meghatározni, hogy azok előforduláshalmaza átfedő legyen.

22.3.22 A szándék jó, de a megoldás rossz. A jogcímkód nem egyed, hanem tulajdonság.

22.3.23 A tervtípus és a változat nem fogható fel egymást kizáró két jelenséggént. Az élő, az aktuális, a legújabb, az elsődleges stb. jelzőjű tervtípus maga is egy változat, ha léteznek változatok. Ezért az utóbbi esetben a terv mindig csak változathoz kapcsolódhat.

22.3.24 Az altípustól eltekintve két egyednek sohasem lehet azonos a kulcsa. Itt tehát a Tanfolyamszám homonima. Mármint az elemzés során nagyon sokszor a közös azonosító árulja el, hogy két eltérő nevű egyed vagy azonos, vagy az egyik a másiknak altípusa. Hiszen alapvető modellezési szabály, hogy az azonosítók és az egyedek kölcsönösen és egyértelműen meg kell, hogy feleljenek egymásnak. Ebben a példában a megtevesztés lényege az, hogy a figyelmetlen elemző az AKTTANFOLYAM-ot a TANFOLYAM egyik állapotának, változatának stb. gondolhatta, holott nem erről van szó.

22.3.25 A hallgató nem a tanfolyamon, hanem az aktuális tanfolyamon vesz részt.

22.3.26 A modellrészlet hibás. A tervező két dolgot tévesztett össze: a *vizsgát*, ami több személyre is vonatkozhat és a *vizsgázást*, ami egy személyt érint. Így a vizsga helyét és körülményeit sokszorososan rögzíti. A tervezők olykor az alapvető fogalmakat sem látják tisztán, nem megfelelő egyedneveket használnak és amikor az E1 egyedhez kellene kötni egy tulajdonságot, a megtevesztő név miatt azt az E2 egyedhez kapcsolják.

22.3.27 A modellben nem egyes emberek mai szándékait kell tükrözni. A modellnek a valós és tartós összefüggéseket kell leírnia. Ezért akkor is bővíthetőnek kell lennie, ha egyelőre a megvalósítás korlátos. Tehát a tervező nem jelentheti ki, hogy őt csak adott szintig érdeklik a dolgot, afölött és azalatt nem.

22.3.28 A modell redundáns és ellentmondásos, mert kétszer, de más néven - és így feltehetően más attribútumokkal - tükrözi ugyanazt a jelenséget.

22.3.29 A szerződés és a rendelés nem azonos jogi/gazdasági kategóriák, nem azonos valós lényegek. Összevonásuk megengedhetetlen akkor is, ha mai adataik nagymértékben azonosak.

Azért az, mert a modell nem bővíthető, márpedig holnap a két tulajdonságsor már eltérhet egymástól.

22.3.30 Egy szervezetben ezerféle bizonylat létezik. A név biztosan homonima.

22.3.31 Ad 1) Mindkét tulajdonság homonima. Ez a Megnevezés esetében nem vészes, mert csak leíró tulajdonságról van szó. A Kód esetében viszont az, mert annak szerepe azonosító. Ad 2) Éppen ezért ha például a NYELVTUDÁS-ban is Kód a hivatkozás neve, akkor kapcsolati homonima alakul ki és nem tudni, hogy a sokféle Kód azonosítójú segédegység közül melyik felé kapcsolódik ez az egyed. Azt csak a tervező tudhatja - fejből. Ha pedig nem Kód a hivatkozás neve, akkor a kapcsolat hiányozni fog, mert nem felismerhető azért, mert a kapcsoló adattétel szinonima. Ad 3) Mindez abból fakadt, hogy a tervező össze akarta vonni a Kód/Megnevezés párokat tartalmazó táblákat. Ez helyes szándék. Azonban ezt nem teheti meg a fogalmi szinten. Az összevonás és a megbontás hatékonysági okokból történik és ezért az ilyen döntés a logikai tervezési szintre tartozik.

22.3.32 A *pótlék* az az összeg, amit a dolgozó kap egy adott időszakban és/vagy feladatra egy adott *pótlékféle* szerint. A Kód/Megnevezés páros nem nyilván nem magát a pótléket (konkrétum), hanem a PÓTLÉKTÍPUS vagy hasonló nevű egyedet (általános) jellemzi. Nem győzzük eléggé hangsúlyozni a nevek pontos használatának a fontosságát. A modell alkalmazója keresni fogja a személy és a pótlék kapcsolatát, de ilyen nem létezik.

22.3.33 Az egyednév, a meghatározás és a tulajdonságok nem felelnek meg egymásnak. Az attribútumok nem a nyelvtudást, hanem a nyelvet jellemzik. A nyelvtudásnak utalnia kellene a személyre, viszont vagy csak a kódot, vagy csak a megnevezést tartalmazhatja. A meghatározás a nyelvtudást a hallgatóra szűkíti be, holott a hallgató is egy személy és a nyelvtudás ezt az egyedfőtípust jellemzi, nem pedig a hallgató egyedaltípust.

22.3.34 Önnek már tudnia kell, hogy tulajdonság nélkül nem egyed az egyed.

## 22.4 Azonosítók

22.4.1 Bizonylatszámra rengeteg mindennek van és nem minden bizonylat jelent készletmozgást. Ezért az azonosító neve rossz, mert homonimát okoz és téves kapcsolatokat eredményez az elemzés során. Ráadásul a bizonylatszámok felépítése még egy-egy mozgásnem esetében sem homogén és azt igen gyakran változtatgatják. A mesterséges kulcs az ilyen esetekben elkerülhetetlen. Hasonló a helyzet a határozatszám, az iktatószám, a jegyzőkönyvszám stb. adatokkal, vagyis minden olyan ismerettel, ami kimondottan papírazonosítóként szolgál.

22.4.2 Az azonosítók az alárendelt egyedekben kapcsoló szerepet látnak el. A homonima arra vezet, hogy az elemzésben - a közös név miatt - olyan egyedek között is kapcsolat mutatkozik, amik valójában nem állnak összefüggésben.



22.4.3 A Belső azonosító kapcsoló adattételként szükségszerűen megjelenik az azonosított egyed alárendeltjeiben is. Mármint ön mire gondolna, ha egy egyedben olvasná a mondott adatnevet? Mindenre, csak arra nem, hogy a belső azonosító külső kulcs ...

22.4.4 Az elemzés kapcsolathiányt fog kimutatni, mert a kapcsolat létét nem lehet feltárni a nem egyértelmű megnevezések miatt.

22.4.5 Mivel az iktatószám felépítése nem homogén, azonosító és kapcsoló szerepre nem alkalmas. Be kell vezetni egy mesterséges belső kulcsot és az iktatószámot csak külső azonosítóként szabad használni. Az előbbire kell építeni a kapcsolatokat.

22.4.6 A Modulorszám név megtévesztő, hiszen nincs valódi sorrendről szó. Ugyanis az M modul a T1 tárgyon belül nem ugyanannyiadik, mint a T2 tárgyon belül. Ez a szinte nüansznyi tévedés egy sokkal súlyosabbat rejt. A tervező a tantárgy - modul kapcsolatot 1:N fokúnak feltételezte (erre utal a *sorszám* név), holott az valójában M:N fokú.

22.4.7 A *szerepekről* van szó. A tervező nem adta meg még azt sem, hogy melyik tulajdonság vagy tulajdonságok alkotják a kulcsot. Enélkül pedig az elemzés lehetetlen.

22.4.8 Az az egyed, aminek egyetlen leíró tulajdonsága van, nem egyed. Az már más kérdés, hogy a „mi oszlik meg” kérdésre sem kapunk választ, mert nem tudjuk, hogy mennyi az összes megosztható idő.

22.4.9 Ad 1) A tervező nem írta le a az azonosítás módját, vagyis fontos ismeretet rejtett el a modell használói elől. Ad 2) Fogalmi szinten a kapcsolattal való azonosítás kizárt. Az ugyanis tárolási illetve hatékonysági kérdés, ami a logikai tervezési szintre tartozik.

22.4.10 Lásd az előző feladat megoldását! A tulajdonságokból nem tudjuk meg, hogy minek az érvényességéről van szó. Az előbbi bajt egy újabb is tetézi. Az Érvényesség kezdete nyilván az azonosító része kell, hogy legyen. Ezért itt még arra sem lehet hivatkozni, hogy az azonosítás majd kapcsolattal történik, hiszen azzal nem történhet.

22.4.11 Aranyszabály, hogy rendkívüli esetektől eltekintve sem a papíron lévő azonosítókat, sem a neveket nem használhatjuk a modellben egyedazonosítókként illetve azok részeként. Az adott példában az Iktatószámot azért nem, mert ezerféle papírnak van iktatószáma, tehát a név homonima. Egyébként is II. József korába tartozó felfogás az, hogy valami azáltal létezik, hogy iktatják. Az ügyfél esetében ez biztosan tévedés. Az adott példában a név nem egyértelmű (sokaknak lehet azonos a neve) és mivel a kulcs más egyedekben kapcsolóként szolgál, a név nem praktikus azonosító(rész).

22.4.12 Ad 1) A hallgató is ügyfél. Ha nem úgy fogják fel, az baj. Ha igen, akkor meg nem érthető, hogy hirtelen miként lehetett előkeríteni az előző példából hiányzó Ügyfél azonosítót. Ad 2) Ha bizonyos ügyfeleknek az azonosítója az előző példa mintájára alakul, akkor a pályázatok nem kapcsolhatók az ügyfelekhez.

22.4.13 Ad 1) Általános hiba a megnevezések azonosítóként való használata. Ad 2) Speciálisan most ez megmutatkozik abban is, hogy a kiküldetés nyilván több kiküldött fölérendeltje is lehet, tehát az azonosító kapcsoló szerepet is betölt.

22.4.14 Az azonosítók és az egyedek kölcsönös és egyértelmű viszonyban kell, hogy álljanak. A tervező ezt az elvet megsértette, mivel több egyedben eltérő értelemben használta az Iktatószámot (homonima). Ebből pedig az következik, hogy valaki a hallgatót a szerződéshez akarja kötni akkor is, ha ez a viszony a valóságban nem létezik.

22.4.15 Gondolni sem jó arra, hogy ugyanazt a modult több papíron is iktatják, mert akkor teljes egyedelőfordulások ismétlődnek meg azonos módon, csak az iktatószámban különbözve. Ha viszont ez nem fordulhat elő, akkor minek az Iktatószám az azonosítóba?

22.4.16 Ha a modulnak nem kell hivatkoznia a tanfolyamra, akkor miért van szükség az azonosítójában a Tanfolyamszámra? Ha viszont kell, akkor azt nem teszi, mert viszont nem tartalmazza a Partnerszámot, ami a tanfolyam azonosítójának a része. Ha viszont az Iktatószám és a Partnerszám azonos, akkor a terv a szinonima miatt félrevezető.

22.4.17 Magyar helységek esetében a törzsszám látszik megfelelő választásnak. Viszont az irányítószám előnye, hogy a címzésekben is használható. Ugyanakkor ez az adat csak az egyik irányban egyértelmű: az irányítószámból következtethetünk a helységre, de megfordítva nem. (A nagyobb helységeken belül több szám használatos). Egyik azonosító sem vonzó akkor, ha külföldi vagy ismeretlen helységek is szóba jöhetnek. Ilyen esetben nem kerülhető el a mesterséges Helység azonosító alkalmazása.

22.4.18 Ad 1) Egy tulajdonságnak a modellben nem lehet többféle tartalma. Ad 2) Ha a körzet változik, akkor módosítani kell az azonosítót is.

## **22.5 Teljesség és minimalitás**

22.5.1 a) ... csak személyi számos egyénekről vezetünk adatokat. b) ... ha például külföldi személyekről is vezetünk adatokat, akiknek nincs személyi számuk.

22.5.2 Ha a személyi szám kitöltött, akkor a nemet és a születési dátumot automatikusan abból kell átvenni.

22.5.3 A megoldás nyilván hibás, mert redundanciás, ami ráadásul inkonzisztenciára is vezethet. Pl. egymásnak nem megfelelő munkakör-kód és FEOR értékeket visznek be. A hiba gyökere a rossz kódtervezés, amin az elemző már nem igazán tud segíteni. Ezért eljárással kell biztosítani a konzisztenciát.

22.5.4 Szép kis dilemma! Mivel a tantárgy tervezését követően az óraszámok nem fognak sűrűn változni, ezt a redundanciát nyugodtan el lehet tűrni.

22.5.5 A január havi zárás és a február havi nyitás adat megegyezik. Ez akkor is így igaz, ha a zárásadatokat nem tároljuk. Ezért meg kell alkotni az átvezetés tárolt eljárását.

22.5.6 A terv nem teljes, mert a logika szerint hiányzik belőle a Támogatás vége adat.

22.5.7 A modell normalizált és az adatrév is helyes (talán a rövidítéstől és a ponttól eltekintve), hiszen kellően tükrözi a tartalmat. Az ellenőrzéskor foglalkoztatott létszám ugyanis nem azonos a mindenkori létszámmal, mert az utóbbi változik és így a korábbi - az ellenőrzéskori - ismeret elveszik.

22.5.8 Az összes kiadott tétel mennyisége származtatott adat, ami nem a diszpotételt jellemzi, hanem magát a diszpozíciót. Ezért nyilván redundáns a diszpotételekben.

22.5.9 Ad 1) A mi központunkat úgyis ismerjük, tehát annak a címét teljesen felesleges kezelni. Ad 2) A partner központjának a címét nem a szerződésben, hanem a partner egyedben - azaz egy helyen - rögzítjük.

22.5.10 Ezek a nevek alulminősítettek. Nagyon sok egyednek van típus, dátum, leírás stb. tulajdonsága. Ezért az ilyen neveket minősíteni kell, különben biztosan homonimákat kapunk. Ha ma nem is, holnap bizonyára. Azt pedig nem fogjuk tudni eldönteni egy-egy konkrét esetben, hogy a név homonimát vagy valódi redundanciát takar-e.

22.5.11 Néhány esetben egyszerűen csak fecsegő névről van szó. Azonban igen sokszor a bőbeszédű név normalizálatlan modellre hívja fel a figyelmet. Például „A munkát végző dolgozó neve” egészen bizonyosan redundanciát rejt, mert a modellben kell, hogy legyen Személy neve tulajdonság, ami magát a személy egyedet írja le.

22.5.12 Ha a számított adat és annak tényezői egy egyedet jellemeznek, akkor az előbbi modellezzük ugyan, de az adatbázisban nem fogjuk tárolni. Ha ezt mégis megteszük, az sem számít komoly hibának.

22.5.13 A redundancia óriási. Hány száz hallgatónál vezetnek feleslegesen azt a képzési díjat, amit egyszer már rögzítettek a tanfolyamnál?

22.5.14 Ad 1) Ha az egyéni képzési díj nem mindig azonos a tanfolyamival, akkor két eltérő adatról van szó, tehát a képzési díj homonima, mert kettős a jelentése. Ad 2) A diáknál nem a képzési díjat, hanem magát a kedvezményt kell vezetni. Ha ezt nem teszik meg, akkor a modell nem teljes a valósághűséget tekintve. Ha megteszik, akkor pedig redundáns, hiszen az egyéni képzési díj a tanfolyamiból és a kedvezményből számítható.

22.5.15 A helyzet erősen homonimagyánús. A modell elvileg lehetne hibátlan is abban az esetben, ha a sorszám egy egyedben azonosító, az összes többiben pedig leíró. Ennek az esélye azonban gyakorlatilag nulla.

22.5.16 A redundanciák és/vagy homonimák feltárásához szükséges a több egyedben is előforduló tulajdonságok **szerepeinek** az ismerete. Annak tudása, hogy azonosító, kapcsoló, vagy csak leíró szerepű tételről van-e szó. Mivel ezt az információt a CASE-ek nem vezetik, nem a szükséges mértékben támogatják az elemzést. (Megjegyzés: A CASE-ekben csak azt lehet megadni, hogy egy tulajdonság az adott egyeden belül kulcs-e. Azt nem, hogy az egész modellt tekintve az-e. Valószínűleg nem jelölhető ki az abszolút szerep.)

22.5.17 Ad 1) A partner neve egyik egyedben sem fordulhat elő, mert ellenkező esetben a kérdéses egyedek nem normalizáltak. Ad 2) Vajon miért nincs a partner neve az egyetlen természetes és jogos helyén, nevezetesen a partner egyedben?

22.5.18 Ad 1) Ha csak saját dolgozókról van szó és a gyerekekről magukról is vezetünk adatokat, akkor az adat felesleges. Ad 2) Minden más esetben a megoldás elfogadható.

22.5.19 Az ÁFA-% cikkcsoportot - és nem egyedi cikket - jellemez. Ha van cikkcsoport egyed, akkor a terv normalizálatlan és ezért hibás. Ha nincs, akkor az a baj. Azonban még magát a cikkcsoportot sem írhatja le a kérdéses adat a változások miatt. Lásd az állapotok modellezését! (Létre kell hozni egy cikkcsoport/ÁFA dinamikus kapcsolóegyedet.)

22.5.20 Az állomány és a tárterület két egymással viszonyban álló dolog. Ez a viszony csak ma és csak adott helyen N:1 fokú. Általában M:N fokú. Ezért külön technikai állományban kell leírni és a Tablespace nem lehet az állomány fix adatmezője.

22.5.21 A felhasználót nyilván az érdekli, hogy az álláskeresés idején az adott szakma hiányszakmának minősült-e. Bár jobb megoldásnak látszik a jelzőnek a szakma egyedben való egyszeri vezetése, az mégsem elfogadható, hiszen az mindig a legújabb állapotot mutatja. Tökéletes megoldás az lenne, ha a szakmának lenne szakmaállapot alárendeltje is. Ámde a legtöbb esetben ez az ismeret felesleges. Ezért a látszólag nem normalizált megoldás talán elfogadható.

22.5.22 Sokszor előfordul, hogy a tervezők nem magát a jelenséget, hanem annak ad-hoc módon kiszakított valamelyik tulajdonságát kötik egy egyedhez. Ha példánkban az utolsó határozat - értsd: a közvetlenül megelőző előbbi határozat - valamilyen másik adatára is kíváncsiak, akkor bajban vannak. Ezért helyesebb ilyenkor szerepnév kapcsolót használni: Utolsó határozat száma. Ezzel elérhető az összes korábbi adat.

22.5.23 Az adat nem a megfelelő egyedben van. A tervező összetévesztette a munkakör és az álláshely fogalmát. A munkakör egy generikus dolog, ami a szervezettől független. Az álláshely pedig olyan viszony, ami egy szervezeti egység és egy munkakör kapcsolatát mutatja. Ezt jellemzi az az ismeret, hogy az állás betölthető-e pályakezdővel és nem a független munkakör egyedet.

22.5.24 A levágásra nincs szükség, hiszen az adatmodell - és az adatbázis - viszonylag könnyen bővíthető új leíró szerepű adatokkal. Viszont tisztességes - nem számjegyes, a tartalmakat pontosan kifejező - neveket kell alkalmazni.

22.5.25 A tervrészlet rossz. Nem a pályázót, hanem a végzettséget jellemzi a fok. A pályázóhoz magát a végzettséget kell beírni. Ez az egyetlen jó megoldás már csak azért is, mert annak az értelmezése, hogy mi „középfokú”, időben változhat.

22.5.26 A megnevezés pontatlanságától eltekintve a modellrészlet normalizált is lehet! Itt ugyanis nem arról van szó, hogy milyen a pályázó mindenkori végzettsége, hanem arról, hogy a pályázat beadásakor mi volt az. Tehát ha nem akarjuk vezetni a pályázó végzettségeinek a történetét - amit nem tehetünk meg józanul -, akkor az adatnak tényleg a pályázatban van a helye. Persze ebben az esetben is kerekébb, pontosabb lenne magának a végzettségnek a megadása.

22.5.27 A modell hiányos és nem egyértelmű. Ha a mértékegység mindig azonos és az is marad, csak akkor hagyható el a mennyiség adat mellől ez a minősítő adat. Ha pedig használjuk, akkor külön ellenőrzőlistát kell felvenni, különben hol „hl”, hol „hekto” módon írjuk például a hektolitert.

22.5.28 Biztosan nem. A név valamilyen eseményre - lásd: megbízás - utal, amiben nem szerepelhet a pénzüintézet neve, legfeljebb a hivatkozása. Azért nem, mert a nevet természetesen egy helyen - magánál a pénzüintézetnél - fogjuk tárolni.

## 23. KAPCSOLATMODELLEZÉS - MEGOLDÁSOK

### 23.1 Rossz kapcsolatmegadások

23.1.1 Egyes CASE-ek működési logikája hibás. Egy modellrészbe nem akkor tartozik egy kapcsolat, ha annak eleme a fölé- *vagy* az alárendelt egyed, hanem csak akkor, ha annak eleme a fölé- *és* az alárendelt egyed is. Tehát például ha az eszköz egyed eleme a modellrésznek, de az eszközmozgás nem, akkor az eszköz - eszközmozgás kapcsolatnak nem kell ebben a modellrészben megjelennie és azt az áttekinthetlenségig terhelnie.

23.1.2 A szervezeti egységek és az alkalmazottak M:N fokú kapcsolatát mindig egy külön munkaviszony egyeddel kell tükrözni (idődimenzió!) és azt nem szabad mesterségesen 1:N fokú hierarchiára redukálni. A jó modellben a munkaviszony a személy és a szervezet között áll fenn úgy, hogy az utóbbi két egyed a partner altípusa. Ilyen módon nemcsak a saját foglalkoztatást tükrözhetjük, hanem a külsősök alkalmazását is. Egyes személyek az alkalmazottaink. A partner-személy-alkalmazott hierarchia már éppen elég mély. Azt nem igen célszerű még tovább terhelni az egyéb alkalmazott alábontással. Ez már azért sem okos dolog, mert az egyéb *alkalmazottnak* aligha lehet több személyes adata, mint az alkalmazottnak. Nem őt, hanem az egyéb *alkalmazást* kell többletadatokkal leírni.

23.1.3 Két hiba is fellép. Ad 1) Két kapcsolatnak nem lehet azonos a neve. Ad 2) Egy alárendelt elvileg kapcsolódhat egy fölérendelthez és annak fölérendeltjéhez is. Azonban csak akkor, ha a két viszony opcionális és kizáró. Esetünkben még ekkor sem igazán jó a megoldás, mert a két fölérendelt 1:1 kötelező kapcsolatban áll. Ezért a KIFEJLESZTŐJE mindenképpen tranzitív kapcsolat.

23.1.4 Az ügyfelek személyek és szervezetek lehetnek. Nem világos a részletből, hogy a szervezet vagy a személy keres-e egy adott szakmát, ami megfelelő végzettséget igényel. (Az is kérdés, hogy mindig csak egyet?) A tervet pontosítani kellene.

23.1.5 A „minden egyes” kitétel biztosan hibás. Vannak olyan egységek, amikhez már nem tartoznak alárendelt egységek. Különben a hierarchia végtelen mély lenne. Az is baj, hogy a viszonyt hierarchikusan fogták fel, mert az időben és funkcióban nyilván hálós.

23.1.6 A „minden egyes” kitétel biztosan hibás. Nem minden alkalmazott vezető. Ezen túlmenően a modell képtelen nyomon követni a vezetőváltásokat. Ráadásul redundáns is. Lásd a 23.1.2 megoldást. A szervezeti egységek és az alkalmazottak kapcsolatát a külön munkaviszony egyeddel kell tükrözni. A *vezetője* kapcsolat sem más, mint egy ilyen viszony. Tehát a tervező az egyedelőfordulás helyett egy rossz kapcsolatot használt.

23.1.7 A szervezeti egység (csak egy?) a képzési program kifejlesztője és bonyolítója. Ez a két egyed közötti statikus többszörös kapcsolatot jelent. Éppen a statikus jelleg jelenti a fő gondot. Biztos az, hogy ugyanaz marad a bonyolító az idők végezetéig? Biztos az, hogy a program

sohasem szorul továbbfejlesztésre? Célszerűbb megoldás lett volna a többszörös kapcsolatot külön viszonyeggyeddel tükrözni.

23.1.8 A tanfolyami záróvizsga nem kötődik modulhoz, ezért a megoldás jó. Viszont nem tükrözi azt a korlátot, hogy csakis a záróvizsga kötődhet a tanfolyamhoz, a többinek pedig csak a modul lehet a fölérendeltje.

23.1.9 Nem szabad a vizsgát és a vizsgázást (valaki leteszi a vizsgát) összekeverni. A vizsgáknak lehet egy előírással sorozata, amit esetleg visszamutatással tükrözhetünk. A vizsgázásban viszont feltétlenül van egy dátum, amiből megtudhatjuk, hogy melyik előzte meg a másikat. Erre kapcsolatot építeni enyhe túlzás.

23.1.10 Ha a tervező azt feltételezte, hogy mindig belső vizsgáztat, akkor aligha tükrözte korrekten a valóságot. Ha külső is vizsgáztathat, akkor zavarban vagyunk az alkalmazott fogalom lényegét illetően. Általában az alkalmazott nem külön egyed (az alkalmazás az), hanem a személynek mint partnernek az egyik időleges jellemzője.

23.1.11 A modell az általános, lényeges és tartós jelenségeket tükrözi. Legyen szabad kifejeznünk kételyünket az orvosi vizsgálat helyének a lényegességét illetően.

23.1.12 Ad 1) A szerződést nem egy cím teljesíti, hanem maga az ügyfél. Vannak olyan szerződések is, amikben nem egyetlen ügyfél szerepel. Ad 2) Ha a szerződésben mind az ügyfelet, mind az ügyfélcímet megadjuk, akkor az redundancia. Ad 3) A szerződésben néha valóban utalnak a teljesítés helyére, ami pusztán formai dolog. Ezt az ismeretet meg lehet adni egy leíró adatban is és teljesen felesleges rá egy lényegtelen kapcsolatot húzni.

23.1.13 Ad 1) Bizonyára vannak olyan szerződések, amikben nem egy ÁFA-körbe tartozó cikkek/szolgáltatások szerepelnek. Ezért hiba, hogy a hálót a modell hierarchiára szűkíti le. Ad 2) Az ÁFA-kulcs egyébként sem közvetlenül az egyedi termékeket, hanem azok csoportjait jellemzi. Vagyis a kapcsolat „átugrott” egy szintet.

23.1.14 A szerződéses ár nem a számlázáskor, hanem a megrendeléskor válik ismertté. Ezért értelemszerűen a rendeléstételt jellemzi, a számlasorban pedig redundáns. Tehát a vonatkozó kapcsolat végeredményben tranzitív.

23.1.15 Az anyagigény nincs minősítve. A rendezvénynek van egy előfeltételezett és egy tényleges anyagigénye. Mivel a modell erre nem utal, féloldalas.

23.1.16 Lásd a 23.1.13 példát. A terméket kell besorolni termékjegyzékbe és az utóbbi lesz az ÁFA alárendeltje. Ebből a hierarchiából a tervező egy szétmutató hálót kreált.

23.1.17 A USER technikai egyed. Nem szerepelhetne a modellben. Ha viszont megteszi, akkor érdemi egyedhez - vagyis a partnerhez - illenék kapcsolni.

23.1.18 Ad 1) Gazdag cég lehet az, ahol egy alkalmazott van egy helyiségben. Ha a foglalkoztatás helyszínét jelöli az alkalmazott - helység kapcsolat, akkor azt tévesen teszi. Ad 2) A szervezeti egység - alkalmazott - helység hierarchia miatt a viszony tranzitív.

23.1.19 A technikai szinonima ördögével állunk szemben. A tervező kétszer határozta meg ugyanazt a kapcsolatot úgy, hogy nevük csak egy betűben tér el. Ha netán más lenne a helyzet, akkor a terv még rosszabb: akkor ugyanis érthetetlen a kettősség.

23.1.20 A helyiségben nem az eszközmozgás, hanem az eszköz van. A helyiséget ezért leltár-helyként kellene felvenni. Azt, hogy hol található egy vagyonelem - ilyen az eszköz is - a leltárhely/vagyonelem dinamikus viszonytal szoktuk tükrözni.

23.1.21 Ad 1) A tervező nem jelentette ki, hogy egymást kizáró altípusokról van szó. Nem minősítette „is egy” kapcsolatnak a viszonyokat. Ad 2) Az altípusok körét mindig egy helyen illik megadni. Itt az altípusok köre nem teljes (lásd pl. ingatlan). Ad 3) A tartozék nem tekintendő vagyonelemnek, ezért inkonzekvens az implicit osztályozás.

23.1.22 A tervező megfordította a kapcsolatot. Vannak ÁFA-kulcsok, amelyek -tól/-ig időszak-határok között érvényesek. Tehát az érvényesség az ÁFA-kulcs alárendeltje. Ha nem az lenne, akkor a kulcs időbeli története nem volna nyomon követhető.

23.1.23 Nem tudni, hogy mit fejez ki az ügyfél - leltározott eszköz kapcsolat. Azonban egyrészt a „leltározott” minősítés gyanús, mert az eszköz az eszköz. A leltározás viszont nem tárgy, hanem esemény. Mivel pedig a szervezeti egység is és az ügyfél is partner (sőt, ezesetben az ügyfél a partner szinonimája) a kettős - közvetlen és közvetett - kapcsolat megmagyarázhatatlan.

23.1.24 Ne törődjünk most azzal, hogy a hozzáférési rendszer elemei nem modellezendők együtt a felhasználói ismeretekkel! Ha viszont már megteszik, akkor azoknál is törekedni kell az egyszerűségre. A 23.1.17 példa szerint a felhasználó is a bizonylathoz kapcsolt. Evidens, hogy a hozzáférési sémában a felhasználó/alrendszer viszonyt vezetni kell. Ergo a bizonylatot ahhoz kellene egyszerűen kötni, nem pedig külön-külön a felhasználóhoz is és az alrendszerhez is. Tipikus hiba az A és B felé történő kimutatás az AB helyett.

23.1.25 Az elutasító határozat is egy határozat. Itt tehát nincs szükség külön egyedre, hanem visszamutatást kellett volna alkalmazni. Az már más kérdés, hogy ha minden határozat egyben elutasító is, akkor milyen szolgáltatást kaphat az ügyfél?

23.1.26 Sohasem az éves adat határozza meg a havit, hanem fordítva. Ez csak formai, megfogalmazási hiba. Tartalmi probléma az, hogy a tervező nem ismeri a tervek, tények, statisztikák modellezésének a módszerét (*Idősorok*  $\Leftarrow$ ). Nem alkalmazunk külön éves, negyedéves, havi stb. idősorokat, mert azokkal minden kapcsolatot megtöbbszörözünk. Az egyetlen idősorban az időszakot minősítjük éves, havi stb. módon. Tehát lényegében csak például egy terv-egyedet alkalmazunk, aminek van éves, havi stb. fejezete.

23.1.27 A tervrészlet rettenetesen hibás. Nem ragadja meg magát a munkát mint valós jelenséget. A munka független attól, hogy belső szervezeti egység vagy külső vállalkozó végzi. Az elvégző jellege és kiléte csak leíró adat lehet. A munkát mindig munkaszámmal kell azonosítani. Ha nem így teszünk, akkor bajba jutunk. Például a munkát valamilyen ok miatt a belső egység nem tudja befejezni és rábízza egy külső vállalkozóra. Ettől fogva a munka nem a volt munkaszámon fut. Tehát senki nem tudja már nyomon követni, hogy maga az elvégzendő feladat mennyibe került.



23.1.28 A megoldás nem jó. A közreműködés egyed csak annyit árul el, hogy bizonyos egységek együtt dolgoznak, de azt nem, hogy milyen ügyön. Szerencsésebb ilyen esetben felvenni egy mesterséges egységet és ahhoz kötni az ügyet és az együttműködőket is. Lásd a 17.2.3 Finanszírozás cím alatti megoldást. Ezen a mesterséges egységen pedig nem kell megütközni. Például a projekt is egy hasonló, többrésztvevős mesterséges szervezet.

23.1.29 Ha egy dolgozó nem egy foglalkoztatónál állt munkaviszonyban, akkor nem a szüneteltetést, hanem a korábbi munkaviszonyait kellene vezetni (azokból adódik, hogy mikor nem állt munkaviszonyban, ami viszont fogalmilag nem szüneteltetés). Ha valódi a szüneteltetés, akkor azt nem az ügyfélhez, hanem a munkaviszonyhoz kellene kötni.

23.1.30 A partnercég rossz specializáció. Ugyanis vannak partnereink (főtípus), akik ill. amik közül egyesek szervezetek (altípus). Ezek mellett nem lehet még külön partnercég inhomogén jellegű alosztály is. Ettől függetlenül az érintett cégeknél sohasem szabad vezetni a mi ügyintézőnk hivatkozását. Gondoljunk a kilépésre, a helyettesítésre stb. Az ügyintéző - szervezet viszony alkalmas a valóság dinamikus tükrözésére.

## **23.2 Rossz egyedmegadások**

23.2.1 A „minden egyes” kitétel megint csacskaság. Rossz megfogalmazás a „tartozik” megjelölés is. Arra utal, hogy a tervező nem alkalmazott explicit specializációt. A helyes megoldás szerint léteznek olyan partnerek (ügyfelek), amik egyben intézmények is.

23.2.2 Ha az ügyfél mindenképpen személy, akkor annak hogyan lehet az intézmény az alárendeltje? Ez a modellrészlet roppant zavaros.

23.2.3 Több eset lehetséges. Ha az egyéb szervezeti egység is egy normális egység, akkor nincs szükség a külön egyedre és a tervező nem alkalmazta a szükséges visszamutatást. Ha az egyéb szervezeti egység netán külső - más szervezet részlege -, akkor nem hihetünk abban, hogy a kapcsolat 1:1 fokú. Ha az egyéb szervezeti egység netán a normálisnak csak a részletezése, akkor a modell elrejti ezt a fontos tényt.

23.2.4 Szerintünk a viszonyt megfordították. Ezen kívül mindig vigyázni kell a kettős kapcsolatokkal. Ezért célszerűbb lett volna egy viszonyegyed alkalmazása.

23.2.5 A vizsgán szerepelhet elnök, megfigyelő stb. is. Lásd érettségi. Ezért a vizsgát - pontosabban a vizsgáztatást! - az általánosabb személyhez kellene kötni és a vizsgáztató mint olyan a személy felesleges specializációja. (NB.: A vizsga és a vizsgáztatás éppúgy nem azonos dolog, mint a tanfolyamtípus és a tanfolyam. A vizsgáztatás egy adott időpontban végrehajtott vizsga. Az utóbbi egyed saját ismeretekkel is leírható.)

23.2.6 Mindenféle szempontból kizárt az, hogy a személyek és a helyiségek kapcsolata valamelyik irányból hierarchikus legyen. Ez a viszony mindig dinamikus hálót alkot.

23.2.7 Lásd a 23.2.5 megoldást. Most a pályázat és a pályázás fogalmát keverték össze. A pályázatot egy cég írja ki (egyszeres) és arra több ügyfél pályázhat (többszörös). Tehát az ügyfél nem a *pályázat*, hanem a *pályázás* fölrendeltje.

23.2.8 A saját helyiségek valamilyen elrendezését meg lehet adni (bár ez felesleges, mert lényegtelen ismeret), de aligha képzelhető el, hogy ez a viszony hierarchikus. Bár nem tudjuk, hogy mit jelent az idegen helyiség, szinte kizárt, hogy kapcsolata kötelező legyen. A kölcsönös 1:1 fokú viszony aligha tükrözi helyesen a valóságot.

23.2.9 A szerződésekben nem bankszámot, hanem bankszámlaszámot jelölnek meg. Több szerződésben nem is egyet, mert például az előleget máshonnan szokták fizetni, mint a többi tételt. Ezért a bankszámlaszám és a szerződés viszonya nem mindig hierarchikus.

23.2.10 Már utaltunk arra, hogy nem ritka a többügyletes szerződés. Ha a szerződés az ügyfélre és annak bankszámlájára is hivatkozik, akkor a viszony tranzitív. (A bankszámla adott ügyfélhez tartozik!) Ezt a redundanciát el lehet viselni.

23.2.11 A tervező papírokat modellezett. Ám lehet, hogy nem tehetett mást.

23.2.12 A bizonylat tételek és a főkönyvi tételek viszonya adott esetben hálós is lehet. Most azonban nem ez a fő gondunk. A bizonylat tétel mint egyed nem megfogható dolog. Rengeteg mindent bizonylatolnak. Tehát a modell nem egyértelmű.

23.2.13 Lásd a 23.1.16 feladat megoldását.

23.2.14 Ha a termékváltozat semmihez sem kapcsolódik a terméken kívül, akkor igen kérdéses, hogy egyáltalán szükséges-e. Azért kérdéses, mert a változatokat általában nem külön egyedben tükrözzük. Ha van egy alapcikkünk és annak pár változata, akkor azok mindegyikét szintén cikként vesszük fel az alapcikkhez kapcsolt módon és a változatokra külön jellel utalunk.

23.2.15 A tulajdonságokat. Esetünkben ki is derült a turpisság. A tervező ugyanazokat a tulajdonságsorokat alkalmazta az altípusokban. Vagyis nem tartotta be a specializációnak azon alapszabályát, ami szerint az altípusokban nem lehetnek közös tulajdonságok.

23.2.16 Ha a történeti ismeretek is szükségesek, akkor a kapcsolat hibás. Például az év közben műszakrend-váltás történik és az egyik hónapban még így, a másikban már úgy kell elszámolni a műszakot. A tervezett modell erre nem alkalmas.

23.2.17 Egyes tervezők kivételezésre és bevételezésre bontják az eszközátadást, ami pedig gazdaságilag egyetlen egy mozgásnemet képez. Tehát a terv nem valósághű.

23.2.18 Az eszköz csak időlegesen a tartozék fölrendeltje. Ugyanaz a tartozék áttehető másik eszközhöz is. Ezért az eszköz - tartozék viszony valójában hálós.

23.2.19 Az egyén egy valós jelenség és nem szabad két egyedben tükrözni. Ezért már az is hiba, ha az egyednév kvázi sorszámot tartalmaz. Feltehető, hogy a tervezőnek az volt a feladata, hogy több alkalmazást integráljon (egyben ugyanis nem lehet eleve két egyén). Lusta volt és az

aggregálás helyett megőrizte a két „egyedet” - valójában táblát - azért, hogy ne kelljen pár programot újraírnia. Ez bizony halálos bűn.

23.2.20 A tervező nem hozott létre egyetlen generalizált egyedet a támogatásokból.

23.2.21 A generalizáció elmaradása miatt fellépett a „pókháló” jelenség. Ahelyett, hogy az egyetlen általánosított támogatás egyedhez kapcsolódnának az egyébként teljesen azonos jellegű események (illetve más további egyedek is), minden ilyen viszonyt meg kellett ennszerezni.

23.2.22 A munkáltató és a munkavállaló egyaránt partner. Az egyik szervezet, a másik személy. Külön munkáltató és munkavállaló partner altípust nem illik alkalmazni. Ha pedig a munkáltató és a munkavállaló explicit egyedként került megfogalmazásra, akkor a megoldás végleg rossz, mert teljesen elmaradt belőle a partner specializációja.

23.2.23 Az adatszolgáltató nem lehet explicit egyed, mi több, egyáltalán nem is egyed. A partner egyed egyik osztályozó tulajdonságtípusa.

23.2.24 Papírokat nem modellezzünk. A papírok meglétét elegendő leíró adatban megadni.

23.2.25 Technikailag a részlet nem hibás. Azonban számok és papírok helyett jobb volna inkább eseménytípusokat és eseményeket modellelni.

23.2.26 Ad1) A partnerben lévő vagy ahhoz kapcsolt cím már utal a településre, az pedig a megyére. A szállító cég is egy partner. Ad 2) A modell fogalmilag zavaros. Ugyanis nem csak egy „telephelye” lehet egy cégnek. Ha pedig több is van, akkor a megye nem a szállítóhoz, hanem annak telephelyéhez kapcsolódik.

23.2.27 Ad 1) Számomra a település és a helység általában szinonima. Ad 2) Vannak ún. településrészek, amik nem önálló helységek. Ilyenekből viszont több is tartozhat egy önállóhoz. Ad 3) Ha a település önálló, akkor minek ahhoz helységet is kapcsolni?

23.2.28 Az adónyilatkozat érvénye korlátos. Ha egy illető egy évben többféle munkát is vállal, akkor lehet, hogy több nyilatkozatot kell benyújtania. Tehát az ügyfél viszonya a nyilatkozathoz nem egyszeres, hanem hierarchikus.

23.2.29 Nem valószínű, hogy egy szervezetre csak egyféle statisztika vonatkozik. Ezért az alárendelt egyed megnevezése félrevezető.

## A. FELADATMEGOLDÁSOK

- 1/1 **A.** A közlés nem tartalmaz új ismeretet.
- 1/2 **A.** A leírt ismeret mindig „csak” adat. Ha valaki nem tudja, hogy mi a rendszám vagy a kocsitípus, akkor benne nem születhet információ.
- 1/3 **S.** Tudja Ön, hogy ki az a Sári? Személy vagy ló?
- 1/4 **Aki** olvasni tud, az észleli és érzékeli a jelsort. Ha nem ismeri a szót, akkor itt meg reked (3). Aki tud eszperantóul, az ismeri a magyar megfelelőt: „navigálás”, és a közlést felfogja (6). Csak az képes értelmezni is a közlést (10) aki azt is tudja, hogy a szó mit jelent.
- 1/5 **2, T, 1, 3**
- 1/6 **L, V, V, L.** A férőhely és a hídhossz nem vezet sehová. Viszont a cikkszám adat azt sejteti, hogy a cikkről is van mondanivalónk. Ha Gábor szervezői tanfolyamra jár, akkor a tanfolyam maga is fontos, Gábortól függetlenül leírható jelenség.
- 1/7 **H, I, I, I**
- 1/8 **H, T, E**
- 1/9 **1.** Csakis egyetlen egyedtypusról lehet szó, mivel az orvos, a beteg és a lehetséges beteg egyaránt SZEMÉLY. Több egyedtypus kreálása az adott esetben az ismeret redundanciájával járna. Hiszen az orvos is biztosított és ő is lehet beteg.
- 1/10 **M.** A pótlék a személy bérét írja le, tehát tulajdonságtípusként tükrözendő. Viszont több személy is ugyanazt a pótlékot kapja feladatától függően. Így a pótlék általánosan is leírható a Pótléknév, Pótlékösszeg, Pótlékfeltétel adatokkal. Ezért egyedtypusként is megfogalmazandó.
- 1/11 **A Rendelésdátum** nem vezet sehová. Ezzel szemben a Vevőkód egy másik egyedre, a konkrét vevőre utal a VEVŐ - RENDELÉS általános kapcsolat szerint. Ezért a két tulajdonság között eléggé lényeges az eltérés.
- 2/1 **H, H, I, H**
- 2/2 **T.** A duplapontosságú adat kimondottan tárolási „trükk”.
- 2/3 **L.**
- 2/4 **I, I, I, I**
- 2/5 **L.** Itt nem a fogalmi lényegen, de nem is a tároláson van a hangsúly, hanem a tartalmon.
- 2/6 **V, E, N, V, N**
- 2/7 **3.** Külön egyed a kocs, a káresemény és a kocsit ért kár. Egy kocsi akárhányszor is szenvedhet, ezért a kár adata nem kapcsolható a kocsihoz. Egy balesetben több kocsi vehet részt, így az előbbi állítás megfordítva is igaz. Vegyük észre, hogy más jelenség a kocsi, a valahol történt baleset és a kocsit ért kár.
- 2/8 **R, R, H**
- 2/9 **2**
- 2/10 **H, H, H, I**
- 3/1 **H, H, I.** Ha nem így gondolkodnánk, Cliff Richard dalait összekevernénk Little Richard nótáival.
- 3/2 **L, S, H**
- 3/3 **AA, LL, AK, AA, LL**
- 3/4 **3**

- 3/5 1. Ismétlődő csoportot nem szabad alkalmazni. A „legmagasabb” képzettség pedig azért csacsiság, mert valakinek lehet több ilyen képzettsége is.
- 3/6 3
- 3/7 Számlaszám: AA. Vevőkód: AK. Sorszám: RR.
- 3/8 4. Sokan a 3. megoldást választják. Nem jutnak vele messzire, mert kiderül, hogy téves az egész ismeretsor. Minden jelenségnek ténylegesen azonosíthatónak kell lennie. Ameddig nem az, addig nem lehet vele számolni.
- 4/1 N, H, N. Általában a kulcsokat nem minősítjük, a leírókat viszont igen.
- 4/2 H, I, I, I
- 4/3 I, H, H
- 4/4 R, J, J
- 4/5 R, R, J
- 4/6 F, P
- 4/7 2, 4
- 4/8 3, 4
- 6/1 1 - A Vevőcím nyílt logikai átfedést okoz.
- 6/2 3, 5 - A Vevőnév és Vevő megnevezés szinonima. A két egyed nem kapcsolható.
- 6/3 7 - A részlet egyensúlytalan. Könnyen kikereshető az, hogy egy személy milyen nyelveket beszél. Viszont nehezen keressük ki, hogy ki tud angolul.
- 6/4 K - Mivel bármikor szükség lehet újabb keltezésre (például szállítás dátuma), az általános Dátum név alkalmazása nem célszerű. A leírókat minősíteni illik. Ezért a Rendelésdátum név használata helyesebb lenne.
- 6/5 K - A terv technikai szinonimát tartalmaz. Gondolkozni kell azon, hogy a két adószám azonos lényeg-e. Az azonosítókat kapcsoló szerepük miatt nem illik minősíteni. Itt helyesebb lenne az egyszerű Adószám név használata.
- 6/6 B - Tipikus bizonylatszemplélet. A személy törzslapján szerepel a foglalkoztató szervezeti egység kódja és neve. Ezért a tervező mindkettőt felvette a SZEMÉLY állományba, holott külön SZERVEZET állományban lenne csak szabad a nevet rögzíteni. Ha a Szervezetnév változik, akkor azt így minden foglalkoztatottjánál karban kell tartani, tehát a feldolgozás többszörös.
- 6/7 K - Tipikus kimenetorientált terv. Nem magára a lényegre (cikk) utal annak azonosítójával (Cikkszám), hanem osztályozási ismérveket sorol fel.
- 6/8 Első ránézésre az Ár nyílt logikai átfedés. Azonban lehet, hogy homonimáról van szó. Létezik a cikkekre jellemző Egységár, de egyes rendelésekben kialakított árat alkalmaznak. Ha a kedvezményes árak alkalmasszerűek, akkor az árak a RENDELÉSTÉTEL-ben van a helye, de mindenképpen más - minősített - névvel.
- 7/1 1,4
- 7/2 E - Ez becsapás volt. A TELEPÜLÉS egyeden belül minden településnek van neve és irányítószáma. (A többszörös irányítószámmal itt nem foglalkozunk.)
- 7/3 A - Gyerekkód → Apakód  
B - Gyerekkód <-/-> Apakód
- 7/4 Raktárkód <-/-> Cikk-kód  
Raktárkód ← Cikk-kód  
Rendelészám → Cikk-kód  
Rendelészám <-/-> Cikk-kód

- Irányítószám → Kerületkód  
 Férjkód ↔ Feleségkód  
 Férjkód <-/-> Feleségkód
- 7/5 **J** - Minden településnek van irányítószáma, de nem minden ilyen szám azonosít települést.
- 7/6 **3**
- 7/7 **3** - Két tulajdonság viszonya nem lehet ilyen is, meg olyan is. Ha millió rendelés között csak egy is van, amelyben több cikket kérnek, akkor már a RENDELÉS és a CIKK egyed M:N-es viszonyú, tehát kulcsaik egymástól függetlenek.
- 7/8 **S** - A terv rejtett ismétlődést tartalmaz, amit sohasem szabad álnevekkel leplezni.
- 7/9 **SZEMÉLY (Törzsszám, ...)**  
**SZÁMLA (Törzsszám+Hónap, ..., Számlaösszeg)**
- 8/1 **0NF** - Az egyed ismétlődést tartalmaz.
- 8/2 **1NF**
- 8/3 **1NF**. Az Egységár mindig a cikket jellemzi, tehát a Cikkszámától függ.
- 8/4 **CIKK (Cikkszám, Egységár)**  
**SZÁMLATÉTEL (Számlaszám+Cikkszám, ..., Tételérték)**
- 8/5 **2**. Sohasem szabad előfeltételezésekkel élni. Kétség esetén kérdezni kell.
- 8/6 **3NF** - Minden gépkocsinak van kötelező biztosítása. Ezek szerint a Rendszám és a KB-kötvényszám függés kölcsönös. Az utóbbi tétel valódi „kulcs”, aminek függéseit nem vizsgáljuk a 3NF alakig bezáróan.
- 9/1 **I, I, I, H** - Az utolsó tétel azért hamis, mert elméletileg egy kétoszlopos csupakulcs táblázat lebontható két unáris táblára. Gyakorlatilag ennek nincs jelentősége, ezért nem is részleteztük a témát.
- 9/2 **H** - Az egyed nemcsak funkcionális, hanem többértékű függés esetén is meg lehet bontani kapcsolható módon. Tehát a kitétel így helyes: az E (A, B, C) egyed csak akkor egyenlő az E1 (A, B) és E2 (A, C) lebontásainak az összekapcsolásával, ha fennáll az  $A \Rightarrow B$  és  $A \Rightarrow C$  többértékű függéspáros.
- 9/3 **I, I, I, H**
- 9/4 **I, H, I**
- 9/5 Az első egyedben körkörös JD fedezhető fel, tehát az egyed csak 4NF alakú és hármas megbontást lehetne alkalmazni. Viszont a második egyedben az első három tétel funkcionálisan meghatározza a negyediket (melyik csapat, melyik asztalnál, milyen leosztásban milyen eredményt ért el). Ezért ez az egyed nem bontható meg ismeretvesztés nélkül. A példa jól mutatja, hogy az 5NF csak elméleti trükk. A gyakorlatban igen ritka a megbontható, csupakulcs, leíró adat nélküli egyedtípus.
- 10/1 **I, I, I, I, I**
- 10/2 **A TERMÉK** egyedben tetszőleges - létező - dolgozóra és gépre lehet utalni úgy, hogy a dolgozó nem is kezeli azt a gépet. Hiányzik a két egyed közti hivatkozási integritási korlát. A helyes megoldás:
- HASZNÁLJA (Használja azonosító, ..., Műszak)**  
**TERMÉK (Termék azonosító, Használja azonosító)**  
 Használja azonosító {Dolgozó azonosító+Gép azonosító}

- 10/3 A Dolgozó azonosító impliciten redundáns. Azt a Helyettesített dolgozó azonosító tétellel kell kiváltani a homonima elkerülésére. Az egyik azonosító Szabóra, a másik Kovácsra fog utalni. Lásd a csoportfüggés esetét.
- 10/4 A Rendelésszám  $\rightarrow$  Számlaszám függés miatt pszeudo-tranzitivitás áll fenn. Az Érték vagy pszeudo-tranzitív függésű, vagy homonima. Feltehetőleg az előbbi.
- 10/5 A probléma a következő: a Szerződésszám-ot megadhatjuk elemi tulajdonságként, de akkor nem tudunk a Vevőkódon a VEVŐ felé kapcsolni és nem érvényesíthetjük a hivatkozási integritást. Vagy részeire bontjuk, de akkor meg procedurálisan kell ellenőrizni a Kötésszám Szerződésszámon belüli egyediségét. Nem deklarálnak egyetlen helyen az egyediséget. Természetesen a célszerű megoldás a csoport lenne. A fogalmi tervezési szinten feltétlenül az, hogy mindegyik integritási korlátot ki tudjuk fejezni.

## B. FOGALOMJEGYZÉK

Az alábbi listában vastag szedetű oldalszámnál találhatók azok a fogalmak, amelyekre formális meghatározást is adtunk. Normál szedettel utalunk az egyéb fontos fogalmakra.

0NF	<b>92</b>	Denormalizálás	108
1NF	<b>100</b>	Doméjn	<b>48,87</b>
2NF	<b>104</b>	Egyed	<b>15</b>
3NF	<b>105</b>	Egyed belső szerkezete	<b>36</b>
4NF	<b>121</b>	Egyed külső szerkezete	<b>36</b>
5NF	<b>124</b>	Egyed-tulajdonság-viszony	<b>17</b>
Abszolút szerep	<b>35</b>	Egyedaltípus	<b>54</b>
Adat	<b>10</b>	Egyedelőfordulás	<b>15</b>
Adatábrázolás	23	Egyedelőforduláshalmaz	<b>15</b>
Adatállomány	12	Egyedek közötti függés	91,140
Adataimodell	<b>31</b>	Egyeden belüli függés	91
Adatbank	12	Egyedtípus	<b>15</b>
Adatbázis	<b>29</b>	Elemi azonosító	38
Adatdimenziók	12	Elemi függés	90
Adatmodell	<b>30</b>	Elhelyezési mód	23
Adatmodell-diagram	31	Elsődleges kulcs	87
Adatszerkesztés	24	Extenzió	<b>164</b>
Adatszervezés	23	Értékhalmoz	<b>16</b>
Adatszótár	63	Értéktartomány	<b>47,87</b>
Adattétel	10	Fejlesztési adatbázis	59
Additivitási szabály	88	Feltételes függés	<b>157</b>
Aggregálás	<b>165</b>	File	10
Alaptulajdonság	160	File szervezés	23
Alhalmaz függés	<b>153</b>	Fizikai adatfüggetlenség	<b>24</b>
Alkalmazási adatbázis	59	Fizikai adatszerkezet	<b>24</b>
Alséma	<b>31</b>	Fizikai átfedés	<b>78</b>
Alternáló kulcs	87,112	Fogalmi séma	<b>31</b>
Atomi egyedtípus	<b>118</b>	Fogalmi adatszerkezet	26
Attribútum	48,87	Fonál	<b>149</b>
Azonosítás	<b>17</b>	Funkcionális függés	<b>88</b>
Azonosító	<b>17</b>	Függéserő	90
Azonosító szerep	37	Független lebontás	111
Álegyed	<b>68</b>	Generalizáció	159
BCNF	<b>116</b>	Globális nézet	28
Belső kulcstörő függés	<b>130</b>	Hálós viszony	42
Birtoklási viszony	40	Házastárs viszony	53
Családfa	<b>52</b>	Hierarchikus viszony	42
Csoport	24	Hivatkozás integritás	87
Csoportfüggés	<b>134</b>	Hozzáférési mód	23
Csupakulcs egyed	120	Idegen kulcs	87



Információ	<b>11</b>	Nem-értelmezhető tulajdonság	157
Információs erőforrás szótár	<b>63</b>	Nem-független lebontás	111
Intenzió	<b>164</b>	Nem-normalizált egyedtípus	<b>92</b>
Inverzió	<b>174</b>	Nézet	28
Is-egy kapcsolat	40	Normalizálási alap	139
Ismétlődő csoport	<b>42</b>	Normalizálási eljárás	139
Kapcsolásfüggés	<b>123</b>	Nyílt logikai átfedés	<b>73</b>
Kapcsolat	<b>18,51,57</b>	Összekapcsolás	110
Kapcsolaterő	<b>40</b>	Összetett azonosító	38
Kapcsolatfok	<b>40</b>	Összetett függés	90
Kapcsolatelőfordulás	<b>19</b>	Parciális nézet	28
Kapcsolatelőfordulás-alhalmaz	<b>19</b>	Projektivitási szabály	128
Kapcsolat előforduláshalmaz	<b>19</b>	Pseudo-tranzitív függés	<b>136</b>
Kapcsolathiány	<b>77</b>	Reflexivitási szabály	88
Kapcsolattípus	<b>19</b>	Rejtett logikai átfedés	<b>75</b>
Kapcsolótulajdonság	<b>39</b>	Rekord	10
Kapcsoló szerep	35	Relációs adatbázis	<b>86</b>
Kiegyensúlyozatlanság	<b>79</b>	Relatív szerep	<b>36</b>
Kivetítés	110	Részleges függés	90, <b>101</b>
Konkatenált kulcs	38	Specializáció	<b>54,158</b>
Korlát	30	Származtatott tulajdonság	<b>160</b>
Kölcsönös függés	88	Szerepnév	<b>48</b>
Kölcsönös viszony	45	Tartományfüggés	<b>91</b>
Kulcsjelölt	87	Technikai homonima	<b>76</b>
Kulcshelyettesítő	183	Technikai szinonima	<b>76</b>
Kulcsmátrix	144	Teljes függés	90
Kulcsrész	38	Többértékű függés	<b>119</b>
Külső kulcsörő függés	<b>115</b>	Többszörös kapcsolat	<b>151</b>
Látszólagos logikai átfedés	<b>74</b>	Tranzitív függés	<b>105</b>
Leíró szerep	37	Tranzitivitási szabály	105
Lineáris viszony	42	Triviális függés	128
Logikai adatfüggetlenség	<b>26</b>	Tulajdonság	<b>16</b>
Logikai adatszerkezet	<b>24</b>	Tulajdonság általánosítás	<b>175</b>
Logikai átfedés hiánya	<b>77</b>	Tulajdonságérték	<b>16</b>
Metaadat	<b>59</b>	Tulajdonságtípus	<b>16</b>
Metaadatbázis	<b>60</b>	Univerzális reláció	141
Metaadatmodell	<b>60</b>	Veszteségmentes dekompozíció	111
Mező	10	Visszamatató kapcsolat	<b>50</b>
Metszetfüggés	<b>132</b>	Visszamatató viszony	<b>154</b>

## C. IRODALOMJEGYZÉK

- [1] IBM: Vocabulary for Data Processing, Telecommunications and Office Systems. 7. kiadás, 1981. július.
- [2] Idegen szavak kézikönyvtára. Szerk.: Bakos Ferenc. Terra, Budapest, 1967.
- [3] Halassy Béla: Az adatbázis-tervezés alapjai és titkai. IDG. 1994.
- [4] Halassy Béla: Ember - Információ - Rendszer. IDG. 1996.
- [5] Bachman, C.W.: Data Space Mapped into Three Dimensions. Infotech State of the Art Report 15., 1973.
- [6] Bachman, C.W.: Data Structure Diagrams. Data Base (ACM SIGBDP) Vol. 1, No. 2, 1969.
- [7] Codd, E.F.: Extending the Database Relational Model to Capture More Meaning. ACM TODS, Vol. 4, No. 4, 1979.
- [8] Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. CACM, Vol. 13, No. 6, 1970.
- [9] Armstrong, W.W.: Dependency Structures of Data Base Relationships. Proc. IFIP Congress 1974.
- [10] Date, C.J.: An Introduction to Database Systems. 3.kiadás, Addison-Wesley, Reading, Massachusetts, 1981.
- [11] Codd, E.F.: Recent Investigations into Relational Data Base Systems. Proc. IFIP Congress, 1974.
- [12] Rissanen, J. : Independent Components of Relations. ACM TODS, Vol. 2, No. 4, 1977.
- [13] Fagin, R.: Multivalued Dependencies and a New Normal Form for Relational Databases. ACM TODS, Vol. 2, No. 3, 1977.
- [14] Aho, A.V. és mások: The Theory of Joins in Relational Databases. ACM TODS, Vol. 4, No. 3, 1979.
- [15] Halassy, B.: Normal forms and normalization: practical designer's view. Information and Software Technology, Vol. 33, No. 6, 1991.
- [16] Fagin, R.: The Decomposition Versus the Synthetic Approach to Relational Database Design. Proc. 3rd International Conference on Very Large Data Bases, 1977.
- [17] Bernstein, P.A.: Synthesizing Third Normal Form Relations from Functional Dependencies. ACM TODS, Vol. 1, No. 4, 1976.