GÖNDÖR GYÖRGY

Adatszervezés – adatstruktúrák

I

Számoljunk: Egy modern számítógép 1 mp alatt kb. 100 000 karaktert képes elolvasni. Ennyi gépidő ára 1–2 forint. Adatfeldolgozási feladatoknál a számolási idő általában még ennyit sem tesz ki. 1 karakter adatrögzítésének költsége 2–8 fillér, tehát százezer karakteré néhány ezer forint. Úgy tűnik, a feldolgozás költsége elenyésző az adatrögzítéséhez képest. Ez igaz is, ha az adatok struktúrája egyszerű, a rekordok egymástól függetlenül értelmezhetők. Ha azonban az adatrekordok között valami kapcsolat van, pl. minden rekordhoz meg kell találni a vele kapcsolatban levőket, akkor minden egyes rekord feldolgozásakor végig kellene olvasni már az egész file-t. Ez már többezerszeres olvasást jelentene (amelynek költsége egyáltalán nem elhanyagolható), hacsak nem alkalmazunk valamilyen „trükköt".

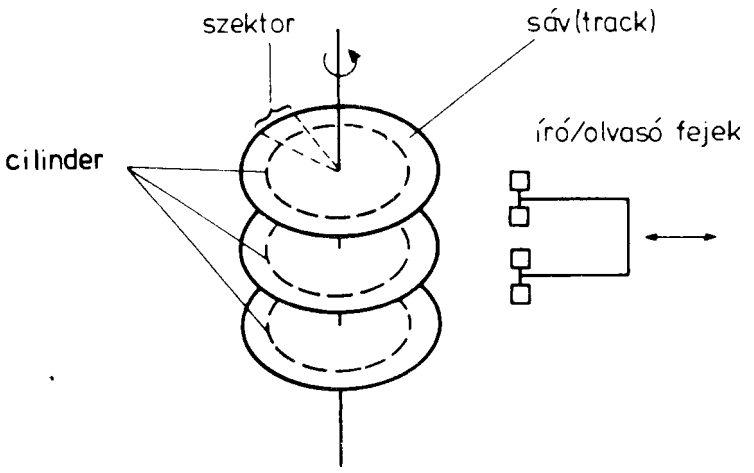
A probléma kifejtésében bátran használtam az olyan fogalmakat, mint rekord, file, hiszen ezekkel a történészek és más társadalomkutatók lassanként már kezdenek megbarátkozni. A baj sajnos éppen az, hogy e fogalmak szerepe ma már meglehetősen vesztett jelentőségéből. A korszerű adatbázisszemlélet nem az adatok fizikai képéből indul ki, hanem a valós világ viszonylatait próbálja az adatokkal modellezni. Ezt a szemléletet szeretném az olvasókkal kissé megismertetni. Nem célom a legújabb módszerek bemutatása – inkább csak azokra a lehetőségekre térnék ki, amelyek már valóban hozzáférhetőek, s éppen emiatt ezúttal nem foglalkozom a fogalmak részletes magyarázatával sem.

II

Az adatok fizikailag továbbra is rekordokba rendeződnek, és ezek file-okat alkotnak. (Adaton, az egyszerűség kedvéért, valamely objektum egy attributumának értékét értjük, amely lehet numerikus (tehát egy vagy több szám) vagy szöveges. Később utalunk rá, hogy a fenti értelemben felfogott adatok közötti kapcsolatokat is hasonlóan lehet kezelni, mint az adatokat magukat.) Az adatbáziskezelő rendszer olyan eszköz, amelynek „optikáján" keresztül a felhasználó több file különböző rekordtípusának mindig egy-egy előre definiált kombinációját látja. Az adatbáziskezelő rendszer a megfelelő rekordokat, ill. ezeknek szükséges mezőit esetleg több file-ról és egy-egy file-nak különböző részeiről szedegeti össze. Ennek műszaki feltétele valamilyen közvetlen címzésű tömegtároló eszköz (lemez vagy dob).

A klasszikus adathordozók, a lyukszalag és lyukkártya csak egyirányban, sorosan, tehát rekordonként egymás után olvashatók. Hiába tudom, hogy csak az 1000-dik rekordot akarom elolvasni, ezt a gép csak úgy tudja megkeresni, hogy végigolvassa az előtte levő 999-et. A mágnesszalag is lényegében ilyen szekvenciális eszköz, bár rendelkezik bizonyos lokálási lehetőségekkel. Nem kell minden rekordot programilag végigolvasni, a szalagot fizikailag mégiscsak át kell csévélni, ami elég sok időbe kerül, és sokszori átcsévélés a szalag állagának se használ. A lemezek műszaki adottságai ezzel szemben olyanok, hogy az úgynevezett fizikai blokkok közvetlenül címezhetők: az olvasó fej tehát kívánságra a megadott fizikai blokk elejére „áll be". Nem tartozik közvetlenül a tárgyhoz, de talán nem haszontalan kicsit jobban megismerni a lemezcsomag működését. A lemezcsomag – típustól függően – 10–20 egymással párhuzamos és egymás fölött elhelyezett kör alakú mágnesezhető anyaggal bevont lemezből áll, amely a meghajtóegységre helyezve nagy sebességgel forog. Minden lemezoldalhoz tartozik egy-egy fej (a meghajtóegységbe van építve), amely sugárirányban mozgatható. Egy fej egy helyzetben a lemez

forgása közben egy ún. sávot (track) érint. Az egymás fölött levő sávok cilindert alkotnak. A sávok szektorokra vannak osztva: ezek a legkisebb címezhető egységek – a fizikai blokkok. Egy szektor megtalálása két mozgásból áll:



1. ábra. lemezcsomag szerkezetének vázlatos rajza

a lemez megfelelő szögelfordulásából (amely viszonylag rövid idő kb. 10^{-5} sec nagyságrendű) és a fejmozgásból (ez már hosszabb). A rekordok ügyes elhelyezésével a fejmozgás minimálisra csökkenthető (pl. az egymás után használandó rekordok egy cilinderen). Egy fizikai rekordba több logikai rekordot lehet összezsúfolni a jobb helykihasználás érdekében (blokkolás), ill. egy hosszú logikai rekord esetleg csak több fizikai rekordba fér bele. Hogy ezt a program is „megértse", a fizikai rekordnak különféle úgynevezett kontroll információt kell tartalmaznia. Pl. a fizikai rekord elejére odaírjuk, hogy az hány logikai rekordot tartalmaz, a végére meg azt, hogy hol található a folytatása.

A következő lépés során bevezetünk egy újabb kontrollmezőt arra a célra, hogy az rámutasson egy, az avval a rekorddal kapcsolatban álló másik rekordra (pl. tartalmazza a rekord fizikai címét: tehát melyik sáv melyik szektora). Ezzel a technikával a rekordok logikai kapcsolatba hozhatók. Természetesen semmi akadálya, hogy többféle ilyen kapcsolómezőt (pointert) is bevezessünk. Ily módon több különböző szempontú rendezettség valósítható meg anélkül, hogy adatainkat többször kellene tárolnunk.

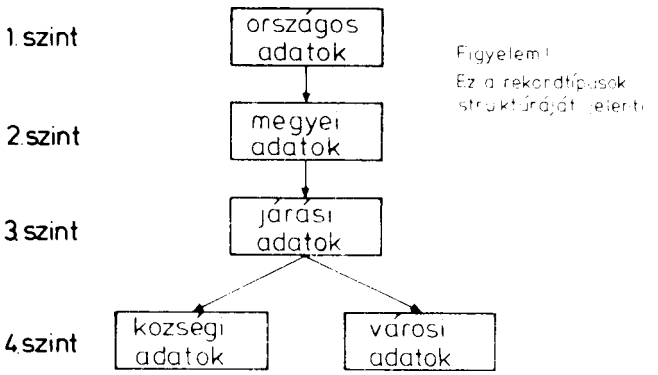
Az utóbbi fontos körülmény. Nemcsak arról van szó, hogy a többszörösen tárolt adatok helyigénye is többszörös, hanem az is, ilyen esetben gondoskodni kell arról, hogy minden példány egyforma legyen: ha tehát egyiken változtatunk, ezt a változást mindegyiken végig kell vezetni. Ez nem látszik nehéz feladatnak, pedig valójában bonyolult követelményeket támaszt.

III

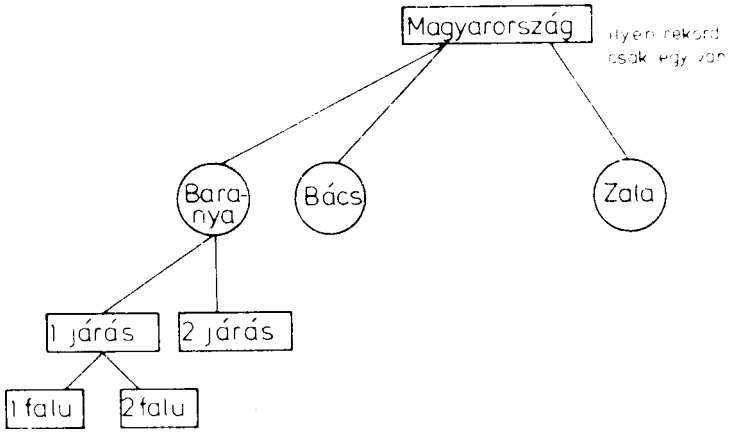
Lássunk egy példát, mire is akarjuk a közvetlen címezhetőségben rejlő lehetőséget felhasználni. Tekintsünk egy képzeletbeli népszámlálás kötetet, amely községi részletességű országos adatokat tartalmaz. Megadja a járási és megyei összegeket, és a városokról részletesebben számol be, mint a falvakról. Közöl további olyan járási, ill. megyei adatokat, amelyeket községi bontásban nem ad meg, és tartalmaz néhány csak országosan értelmezhető adatot is. Induljunk ki a felhasználásból, és próbáljuk meg adatainkat a felhasználáshoz idomuló struktúrával modellezni.

Az ember először is a tartalomjegyzéket nézi meg, hogy megtalálja-e benne a keresett adatokat, ill. egyáltalán milyen adatokat talál. (Ennek számítógépes megfelelője az adatszótár. Lásd később.) A továbbiakban már csak azért nézi meg a tartalomjegyzéket, mert nem akarja „végiglapozni" az egész könyvet. A tartalomjegyzékben talál ugyanis egy oldalszámot (ez egy pointer), és a könyvet már ott tudja kinyitni. A kötet minden táblázatának van címe, amelyből egyértelműen kiderül, megyei, járási vagy községi adatról van-e szó, valamint az is, konkrétan melyik megyéről, járásról, községről. A táblázatok fejléceiből tudjuk meg, hogy az alatta olvasható számok a település mely attributumára vonatkoznak. Az attributumokat általában csoportosítják, tehát pl. az ipari és mezőgazdasági adatokat esetleg külön táblán, hasonlóan a termelést és a fogyasztást is külön. (Ritkán közölnek egy adatot kétszer.)

Öt rekordtípussal van tehát dolgunk: országos, megyei, járási, városi, községi. Az országos kivételével mindegyik típus természetesen többször fordul elő, az egyes előfordulásokat a nevük és/vagy kódjuk azonosítja. A struktúra a következő lehet:



Az előfordulásokat a következő ábra illusztrálja:



A 2–3. ábrán jól látható, hogy a struktúrának különböző szintjei határozhatók meg, mégpedig úgy, hogy minden magasabb szintű rekord előforduláshoz több alacsonyabb szintű rekordelőfordulás tartozhat, de fordítva nem, tehát egy rekord fölött több magasabb szintű rekordelőfordulás nem lehet. Ezt hívják fa szerkezetnek. Két rekordtípus viszonyában a magasabb szinten lévőt szülőnek, az

alacsonyabb szinten lévőt gyereknek, az egy szinten lévő különböző rekordtípusokat, tehát amelyek ugyanazon szülő gyerekei, testvéreknek nevezik. A struktúránál egy szülőnek több gyereke lehet, de egy gyereknek nem lehet több szülője. Abban a speciális esetben amikor minden szülőnek csak egy (típusú) gyereke van, egyszerű hierarchiáról beszélünk.

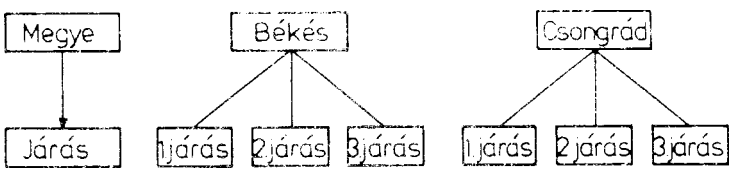
Világos, hogy ez a struktúra nem tükrözi azt a tényt, egy város közigazgatásilag járási vagy megyei szinthez kellene hogy tartozzék, ugyanakkor más logika szerint – a város település, tehát a községekkel kellene egy kategóriába kerülnie.

Ha az országos szint a megyékre, a megyei szint a járásokra, a járási szint a településekre vonatkozóan tartalmaz valamiféle pointereket (ez jelenti a struktúra megvalósítását), akkor egy települést a közigazgatási beosztása alapján három kereséssel találunk meg: az országos szinten megkeressük a megyét, megyei szinten a járást és járási szinten a települést. Tehát a harmadik elolvasott rekord már a keresett település valóságos adatait tartalmazza (az első kettőből csak a pointereket használtuk). Ha az adatainkat mágnesszalagon szekvenciálisan tároltuk volna, a rekordokat sorban kellett volna olvasnunk addig, amíg a kívánt azonosítójú rekordig el nem jutunk. A mai Magyarország 3000 településével számolva ez átlagosan 1–2000 olvasást jelent.

Bár egy településhez meglehetősen rövid út befutásával jutottunk, érintettük annak a járásnak és megyének adatait is, amelybe településünk közigazgatásilag tartozik, tehát ha a megyei és járási szintű adatokat megőrizzük, könnyen megkaphatjuk a település megyei vagy járási százalékos arányait. (Anélkül, hogy ezeket az adatokat közvetlenül tároltuk volna.) Bármelyik közbülső szinten megállhatunk: pl. ha csak járási vagy megyei részletességű adatokkal akarunk dolgozni. A tapasztalat azt mutatja, hogy bár a fa struktúra nagyon is egyszerű, mégis jól modellez sok valóságos problémát. Éppen e kiemelkedő szerepe miatt (és persze mert egyszerű) a fa struktúra az érdeklődés homlokterébe került – sok hatékony és gazdaságos gépi realizálás született rá. Érdemes megjegyezni, hogy írott szövegekben előszeretettel rendszerezzük gondolatainkat pontokba és alpontokba. Ez a fa struktúrának felel meg.

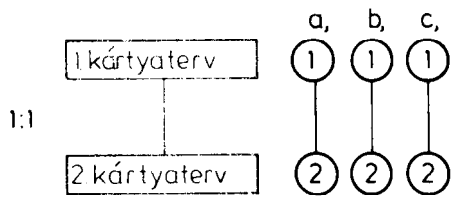
IV

Általánosítsuk a két rekordtípus közötti lehetséges kapcsolatokat. A fa struktúrát nevezik 1 : N fokú kapcsolatnak is, mert egy szülő rekord minden előfordulásához N gyerek rekord-előfordulás tartozik.



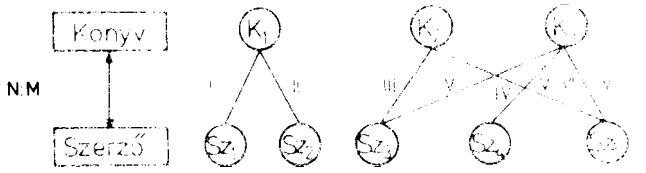
4. ábra. 1 : N fokú kapcsolat sematikus vázlata

Ha minden szülő rekordnak egy gyerek rekordja van: ez 1 : 1 fokú kapcsolat. Könnyű ilyenre példát mondani: ha adatainkat kártyára rögzítjük, de nem férnek el egyetlen kártyára, akkor egy második kártyán folytatjuk. Fölfoghatjuk ezt a viszonyt úgy is, hogy az első kártya a szülő, a második a gyerek és minden első típusú kártyához egy másodiknak is kell tartoznia.



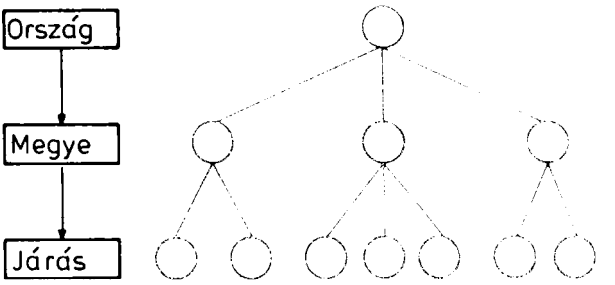
5. ábra: 1 : 1 fokú kapcsolat sematikus vázlata

Végül, néhány esetben az jellemzi a struktúrát, hogy az egyik rekordtípus egy előfordulásához a másiknak több előfordulása tartozik, és fordítva is ez a helyzet. A szokásos példa erre a könyv és a szerző kapcsolata, ahol egy könyvnek több szerzője, és fordítva egy szerzőnek több könyve lehet. Ezt hívják N : M fokú kapcsolatnak.



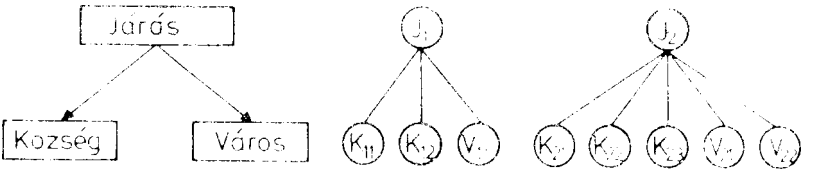
6. ábra: N : M fokú kapcsolat sematikus vázalata

Egyszerű logikai kapcsolatok segítségével összetett logikai szerkezetek alakíthatók ki. Ha csak 1 : N viszonyok léteznek és mindegyik viszonyban a szülő rekordtípushoz csak egyféle gyerek rekordtípus tartozik – egyszerű hierarchiát kapunk.



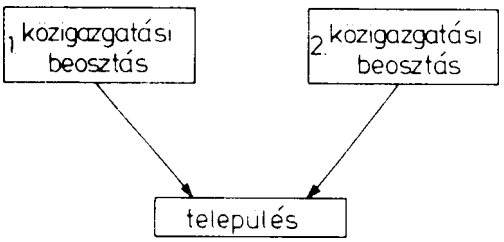
7. ábra: Egyszerű hierarchia

Összetett hierarchiáról beszélünk, ha egy szülő rekordtípushoz több gyerek rekordtípus is tartozhat.



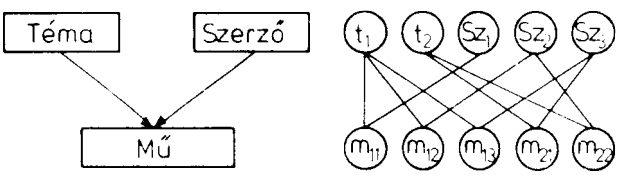
8. ábra: Összetett hierarchia

A hierarchiákra általában az jellemző, hogy egy gyerek rekordtípusnak mindig csak egy szülő rekordtípusa lehet. Kezdetben abból indultunk ki, hogy a valós világ tényleges viszonyait akarjuk az adatstruktúrával modellezni. Előfordul, hogy egy gyerek rekordtípusnak több szülő rekordtípusa van, ilyen esetben hálós szerkezettel van dolgunk. Tekintsük a következő példát. Két időpontból származó országos, községrészletességű összeírást akarunk együtt feldolgozni, de a két időpont között a közigazgatási beosztás megváltozott; a települések még viszonylag jól azonosíthatók. Ezt például a következő struktúrával tudnánk leírni:



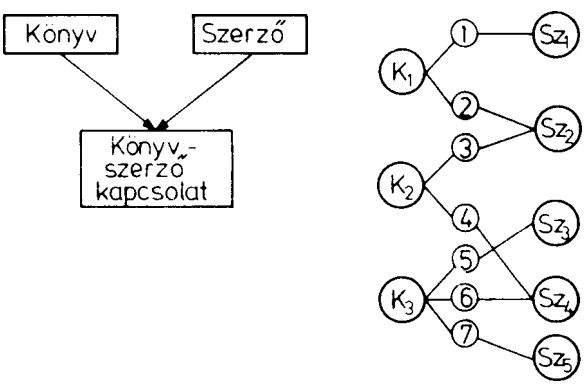
9. ábra: Példa hálós adatszerkezetre

Itt a közigazgatási beosztás további két szintet tartalmaz: ún. megyei és járási szintet; a település rekordtípus tartalmazza mindkét felvétel időpontjából származó adatokat. Másik példa: könyvtári nyilvántartásnál, ha a szerzőcsoportokat egy „szerző"-nek és a művek témáját egyértelműnek tekintem, (pl. mindig megállapítok egy fő témát), tehát beérem az N : M kapcsolatok helyett 1 : N kapcsolatokkal – a következő háló sémát nyerhetem:



10. ábra: Példa hálós adatszerkezetre

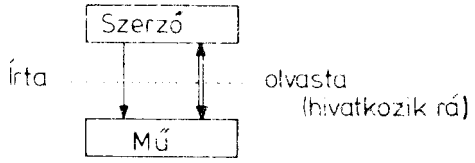
Hálós szerkezeteket is leíró adatbázisok gépi megvalósítása lényegesen bonyolultabb, mint a hierarchikusé, de még így is egyszerűbb, mintha N : M kapcsolatok leírására is alkalmassá akarnám tenni őket. Léteznek viszont kerülő utak. Hierarchikus adatbázissal is leírhatunk hálós kapcsolatot, csak az egyszeres adattárolás nyújtotta előnyöket kell feláldozni. Hálós adatbázissal – ún. kapcsoló rekordok alkalmazásával – N : M fokú kapcsolatokat is tudunk kezelni. Lássunk erre is egy példát. A 6. ábrán sematikusan ábrázolt N : M fokú kapcsolatot így tudjuk átalakítani:



11. ábra: N : M fokú kapcsolat megvalósítása hálós adatszervezéssel kapcsolórekordok segítségével.

A 6. és a 11. ábra összevetéséből jól látható, hogy az ár, amelyet fizetnünk kellett, nem csekély, a 6. ábra minden egyes kapcsolatához egy-egy külön kapcsolórekordot kellett bevezetni.

Többcélú adatbáziskezelő rendszer két rekordtípus között nemcsak egyféle kapcsolatot enged meg. Ilyen összetett hálókapcsolat a következő:



12. ábra: Összetett hálókapcsolat

Itt a kétféle rekordtípus között kétféle kapcsolatot próbálunk kifejezni, egyrészt egy szerző több művet írhatott, másrészt egy szerző több műre is hivatkozhat. Ez az utóbbi kapcsolat N : M fokú, mert egy műre is több szerző hivatkozhat.

Végül még egy – gyakorlatban fontos – kapcsolattípust szeretnék megemlíteni. Azonos típusú rekordok is lehetnek egymással kapcsolatban. Például egy községi anyakönyv esetében: ha csak a férfiakat tekintjük, mindenki fia valakinek, egyúttal apja lehet másoknak. Egyetlen rekordtípusról van szó („férfiak") és ezek között létesítettünk kapcsolatokat (apja, fia).

V

A rekordok közötti kapcsolatokat a gépen – mint arra már a bevezetőben utaltam – pointerek valamiféle rendszere valósítja meg. Ha tehát egy struktúra felső szintjén megtaláltuk valamelyik elemet, innen a pointerek vezetnek el az alsóbb szintekre. De hogyan keresünk a legfelső szinten? Szervezhetjük úgy az adatbázisunkat, hogy a legfelső szinten csak egyetlen előfordulás legyen (mint a példaként említett népszámlálás országos adatai esetében). Ha több, de nem túl sok előfordulása van a legfelső szinten lévő rekordtípusnak, ezek között szekvenciálisan kereshetünk; azaz sorban elolvassuk őket mindaddig, ameddig a megfelelőt meg nem találjuk. Sok előfordulás esetén indexelt szekvenciális vagy random keresést alkalmazhatunk. A két eljárás abban különbözik egymástól, hogy az előbbinél egy táblázat, az utóbbinál egy algoritmus alapján a rekord azonosítójából egy címet tudunk előállítani, és az ilyen azonosítójú rekordot csak ez után a cím után, ennek közelében kell csak keresni.

Elméletben annak sincs semmi akadálya, hogy egy struktúra belsejében szereplő valamely rekordtípust is valamilyen random eljárással is megkereshessünk. Ha pl. a népszámlálás kötetben gyakran kell keresnem községeket a nevük alapján anélkül, hogy ismernénk közigazgatási hovatartozásukat, célszerű egy betűrendes névmutatót is adni a kötet végén.

A rekordtípusok közötti logikai kapcsolaton kívül a keresés módját is meg *kell* tehát adnunk. Sőt egy adott adatbázisnál a különböző felhasználások számára különböző bejárási utakat adhatunk meg. Ez a lehetőség különösen hálós szerkezeteknél jelenthet sokat, mert ilyenkor – szemben a fa struktúrával – nincs egyértelmű bejárási mód.

Már az adatbázis logikai szerkezetének megtervezésekor figyelembe kell venni a várható felhasználás módját. Ez is beleértendő a „valós világba", amelyet modellezni akarunk. A felhasználási igények figyelembe vételének további lehetőségét jelenti, hogy előírhatjuk a keresés módját.

Nem szabad azonban megfeledkezni arról, minél részletesebben írjuk elő adatbázisunk szerkezetét, megvalósítása annál bonyolultabb lesz, ezáltal előállítási költségei mindenképpen megnövekednek. Ezért cserébe akkor kapunk valamit, ha a várható felhasználás szerint használjuk.

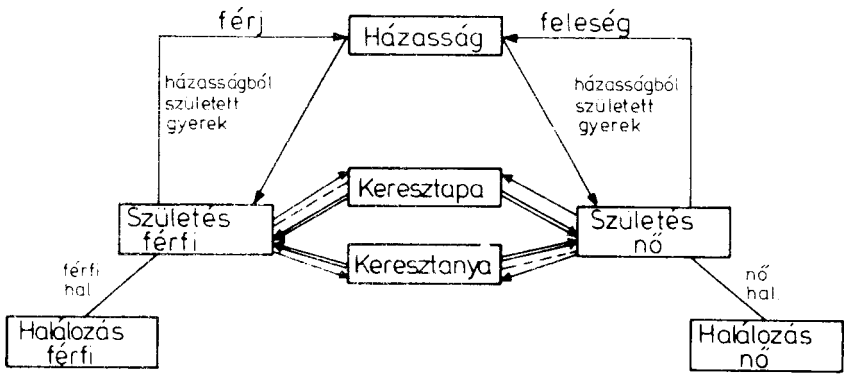
VI

Bemutatnék egy valóságos példát. A példa annyiban reális, hogy az ismertetendő igények tényleg megfogalmazódtak, ily módon tehát egy valóságos munka terve, annyiban azonban nem, hogy a gépi realizálás még kísérleti stádiumban sincs.

Adottak egy község anyakönyvei, meghatározott időintervallumban (mondjuk, 2–300 év). Háromféle anyakönyv létezik: születési, házassági, halálozási. A születési anyakönyvben vezetik a keresztszülőket is – lényegében ez a kutatás tárgya: a keresztszülők választásának rendszere az adott községben, ill. az ebből levonható következtetések. Nem is egy, hanem több községben maradtak fenn hasonló anyakönyvek, a konkrét kutatásba is több községet akarnak bevonni, azonban a községek közötti relációktól eltekinthetünk. Az ismertetendő módszert minden községre külön kell majd megvalósítani.

A személyek azonosítása külön kérdés, erre most nem térek ki, bár a probléma várhatóan súlyos. Egy hagyományos falu társadalma meglehetősen zárt, tehát a szereplők túlnyomó többsége ott született, házasodott és halt meg. Vannak persze kivételek, ilyenkor a nem létező bejegyzést valamilyen módon géppel generáljuk, mégpedig úgy, hogy ebben a fiktív bejegyzésben az egyes adatmezők értéke „ismeretlen". Ennek megvalósítási lehetőségét sem kívánom most tárgyalni. Ily módon az anyagot teljessé tesszük abban az értelemben, hogy minden szereplőnek lesz egy születési, halálozási bejegyzése, és ha gyereke is van, akkor házassági is. Tehát egy házassági bejegyzés nem feltétlenül jelent tényleges házasságot, hanem esetenként csak egy viszonyt, amelyből gyerek származott, ahol esetleg az apa (kvázi-férj) „ismeretlen". Házasságon belül a gyerek apjának mindig a férjet tekintjük.

Milyen kapcsolatokat kívánunk az adatbázisunkba beépíteni? Minden szereplő egyszer születik és egyszer hal meg. Ezek között tehát 1 : 1 fokú kapcsolat van. A házassági viszonynál tulajdonképpen egy adott adatfajtának saját magával való kapcsolatáról van szó. Hogy ezt feloldjuk, külön adatfajtának tekintjük a nőket és a férfiakat. Ilyen módon egy férfi házasságai 1 : N fokú kapcsolatként kezelhetők, hasonlóan a nőkéhez. Csak zárójelben említem meg, hogy egy leányanyához, aki az adott községben szült, ill. anyakönyveztette a gyerekét, generálunk egy fiktív házassági bejegyzést, ahol természetesen jelezzük, hogy ez nem egy valódi házasság. Ily módon minden születést egységesen tudunk kezelni: a házasságok és születések között 1 : N fokú kapcsolat lesz mind a fiúk, mind a lányok esetében. A házassági tanúkat egyelőre nem kívánták a feldolgozásba bevonni. Minden születéshez tartozik azonban N számú keresztszülő – ez 1 : N fokú kapcsolat. A keresztszülők létező személyek, akik valamikor születtek is. Ez utóbbi is 1 : N fokú kapcsolat. Mindeddig nem vettük figyelembe, hogy egy személy több gyereknek lehet keresztszülője. A gyerekek és keresztszülők közötti M : N fokú kapcsolatot, ahol ráadásul a gyerek és a keresztszülő ugyanaz a rekordtípus, a keresztszülő nevű kapcsolórekorddal két 1 : N fokú kapcsolattá alakítottuk. Az ismertetett megoldást sematikusan így ábrázolhatjuk:



13. ábra: Logikai kapcsolat az anyakönyvi bejegyzések között

ahol: ——→ 1 : N fokú kapcsolat a született gyerek keresztszüleire mutat

– – – 1 : fokú kapcsolat a keresztszülő rekordban csak megnevezett személytől a születési rekordban lévő részletes adataira mutat

═══ 1 : N fokú kapcsolat a keresztszülő összes keresztgyerekeire mutat

(a többi kapcsolat az ábrán meg van nevezve)

Látható, hogy ez a megoldás egy meglehetősen bonyolult adatstruktúrát eredményezett, amely azonban már tartalmazza mindazokat a kapcsolatokat, amelyekre a kutató kérdései vonatkoznak. Ha már egyszer az adatbázist létrehoztuk, azt várhatjuk, hogy a „beépített" kapcsolatoknak megfelelő lekérdezéseket a gép gyorsan fogja végrehajtani. Kérdés persze, hogy az adatbázis létrehozásának költsége nem túl magas-e, magyarul a megoldás végső soron gazdaságos-e, figyelembe véve, hogy itt egyszeri felhasználásról van szó. De ez már konkrét tervezési probléma, melyre nem térek ki.

VII

Az adatbázis-koncepció további előnyöket is jelent. Ezek közül hármat szeretnék kiemelni. Mostanáig az *adatleírás programfüggetlenségéről* beszéltem. Érzékeltetni próbáltam, milyen lehetőségek léteznek. A valós világ egyedei között fennálló kapcsolatok maguk is lényeges információt tartalmaznak. E kapcsolatok maguk is adatként kezelhetők. Több felhasználó esetén nincs szükség arra, hogy egy új felhasználó ismerje a kész programok adatkezelő, adatleíró részét, ezt az adatleírást az adatbázis tartalmazza, és az *adatszótáron* keresztül bárki számára hozzáférhető. Az adatszótárban megnézhetjük, hogy az adatbázis milyen adatokat tartalmaz, és hogy ezek között milyen kapcsolat van. Annak, aki az adatbázishoz hozzá akar férni, a kezelő rendszer nyelvén kívül tulajdonképpen csak az adatbázis nevét kell ismernie, minden egyéb információt maga az adatbázis is képes szolgáltatni. Ebben az értelemben az adatbázis *önmagát dokumentálja*, mégpedig a lehető legpontosabb módon, mert az adatbázisra magára vonatkozó információt, az adatok és adatkapcsolatok elnevezését csak egyetlen példányban tárolja. Ide fordul a lekérdező rendszer és adatszótár kezelő rendszer is. A két dolog tehát mindig összhangban van. Ha az adatbázis szerkezetén változtatunk, ez azonnal észrevehető az adatszótárban. Erre a hagyományos írásbeli dokumentáció nem képes. Ott előfordulhat egyszerű elírás, időbeli késés, vagy bármilyen más tévedés. A hagyományos dokumentáció lehet akármilyen jó, de semmiképpen nem azonos a dokumentálandó rendszerrel, mint amilyen az adatszótár. Íme az örökösen visszatérő dokumentációs problémák megoldása.