ADATBÁZISKEZELŐ RENDSZEREK ELMÉLETE

jegyzet

Dominich Sándor

1. ALAPFOGALMAK

Adatbázis (Database, DB)

A számítógépen hosszú távon (permanensen) tárolt és nagy mennyiségű adatok (szervezetről, hivatalról, speciális területről, ..) egy rendszerét adatbázisnak nevezzük.

Az adatbázis

- egy adott szakterületet jellemző adatokból,
- az adatok típusát és kapcsolatát leíró *metaadatokból*,
- és az adatkezelő rendszerből áll.

Az adatbázis létrehozásához szükség van :

- egy adatszerkezet-leíró nyelvre (Data Definition Language - DDL), mely lehetővé teszi, hogy absztrakt adatmodellt definiáljunk;
- a fizikai szerkezetet magvalósító nyelvre (Storage Definition Language - SDL);
- és a tárolt adatok különböző szempontok szerinti visszakeresését, feldolgozását lehetővé tevő nyelvre (Data Manipulation Language - DML).

Adatbáziskezelő rendszer (Data Base Management System, DBMS)

Az adatbáziskezelő rendszer az adatbázis használatát és kezelését lehetővé tevő szoftver (programgyűjtemény), mely az adatleíró és az adatkezelő nyelvet foglalja magában.

Néhány közismert képviselője a relációs adatbázis-kezelésben: DB2, ORACLE, INGRES, MS ACCESS. Az adatbáziskezelő rendszerek fő komponense a szabványosított *SQL* (Structured Query Language) lekérdező nyelv.

Az adatbáziskezelő rendszerek jellemzői:

1) Absztrakt adatstruktúrák
2) Titkosság (Nem minden felhasználó fér hozzá az adatokhoz.)

3) Integritás: az adatoknak konzisztenseknek kell lenniük. A konzisztenciát úgy érjük el, hogy konzisztenciafeltételeket definiálunk, és ellenőrizzük azok teljesülését. Az adatok akkor konzisztensek, ha azok az összes konzisztenciafeltételt kielégítik. A konzisztenciafeltételek meghatározásakor megadjuk, hogy minden paraméter csak az adott értéktartományból vehet fel értéket, miközben az adatok közötti összes kapcsolatot és feltételt értelmezzük.
4) Szinkronizáció: adatokon végzett művelet, mely a tranzakciók egyidejűsége esetén is biztosítja, hogy az adatbázis adatai konzisztensek maradjanak.

Nézzünk egy példát a szinkronizációra! Tegyük fel, hogy

P1 személy pénzt vehet ki egy bankszámláról, és

P2 személy is vehet fel pénzt ugyanarról a bankszámláról.

Pénzkivételkor a következő rutinok hajtódnak végre minden személynél:

tranzakció

.

.

feltételellenőrzés: igényelt összeg ≤ számlaösszeg

kivonás : új_számlaösszeg = számlaösszeg - igényelt összeg

frissítés : számlaösszeg = új_számlaösszeg

Ha P1 és P2 személyek nem egyszerre vesznek ki pénzt, akkor az általuk kivett összegek egymás után kerülnek levonásra a bankszámláról, pl.:

számlaösszeg	: 1000 Ft
P1 kivesz	: 50 Ft -ot
számlaösszeg	: 950 Ft
P2 kivesz	: 70 Ft -ot
számlaösszeg	: 880 Ft

Ha azonban P1 és P2 személyek pontosan egy időben vesznek ki pénzt ugyanarról a bankszámláról, akkor előfordulhatna a következő eset:

számlaösszeg	: 1000 Ft
P1 kivesz	: 1000 Ft -ot
P2 kivesz	: 50 Ft -ot
a számláról kivételezünk	: 1000 Ft -ot
a számláról kivételezünk	: 50 Ft -ot
ekkor a számlaösszeg :	- *50 Ft* lenne!

Ezért, ha az egyik személy megkezd egy tranzakciót, akkor a tranzakció végéig le kell tiltani újabb tranzakciók végzését. Ezt a művelet a szinkronizáció. Tehát az adatok konzisztenciáját a szinkronizáció biztosítja.

5) Rendszerösszeomlás elleni védelem, rendszerfelállás Lehetővé kell tenni a rendszeres backup mentést, és az adatbázis adatainak visszaállítását a rendszer sérülése, vagy összeomlása esetén is.

2. AZ ABSZTRAKCIÓ SZINTJEI AZ ADATBÁZISKEZELŐ RENDSZEREKBEN

Fizikai adatbázis (Physical Data Base)

Lemezen tárolt adatbázis.
Szintjei:

- bitek
- bájtok, és azok címei
- fájlba szervezett rekordok.

Fizikai séma (Physical Scheme) : A fizikai adatbázis leírása, terve.

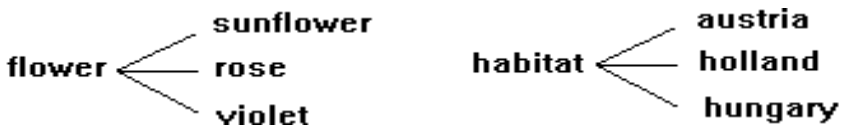
Koncepcionális (elvi, logikai) adatbázis (Conceptual Data Base)

A fizikai adatbázis absztrakt leírása. (Itt már nem adunk meg mezőhosszt, mezőtípust.)

Koncepcionális (elvi) séma (Conceptual Scheme)

Az koncepcionális adatbázis sémája/terve, melynek leírására elvont adatszerkezeteket használunk.

Pl.: flower, habitat, grows_in $\Rightarrow$ flower grows_in habitat

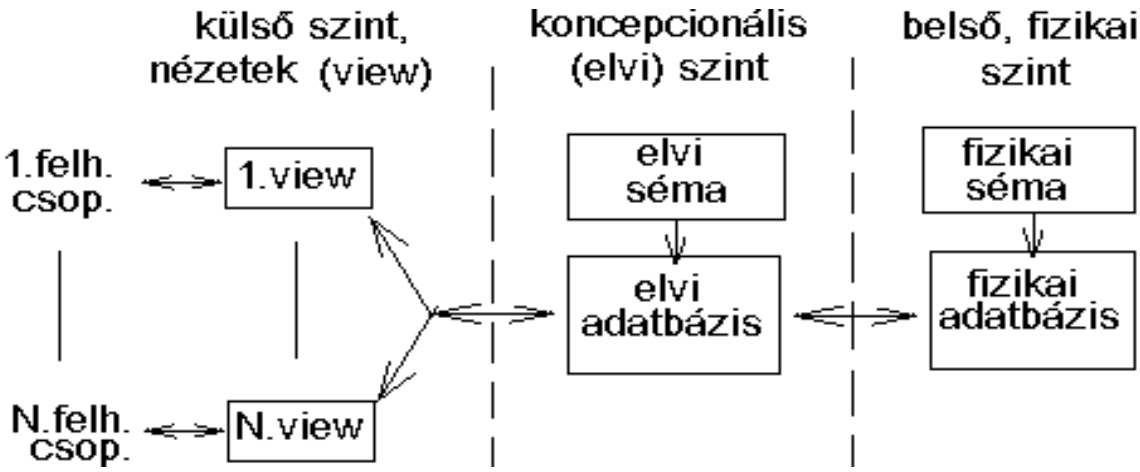


Megj.: A magyar nyelvben a főnevek ragjaiból és toldalékaiból eredő nehézségek az angol nyelvben nem merülnek fel, mivel ott prefixeket használnak. A továbbiakban igyekszem magyar nyelvű példákat használni.

Nézetek (View)

A felhasználók egyes csoportjai nem látják (nem láthatják) a teljes adatbázist, sőt annak részeit is esetleg másképpen látják, mint ahogyan azok a koncepcionális modell szerint felépülnek.

Például az egyik csoport a személyi adatokból nem látja a SZEMÉLY rekord FIZETÉS mezőjét (számára nem jelenik meg), egy másik csoport pedig nem látja a SZÜLETÉSI_IDŐ mezőt. A nézetek kialakítása is adatbázis-tervezési feladat, ami az adatbázis-tervező (DB Designer) és az adatbázis-felügyelő (DB Administrator -DBA) tevékenységei közé tartozik.
Az adatmodell tehát a következő három szintre bontható:



- A külső szintek azt írják le, hogyan látják az egyes felhasználók az adatbázist.
- A középső vagy koncepcionális szint a teljes adatbázis koncepcionális szerkezetét ábrázolja.
- A belső vagy fizikai szint az adatok fizikai elhelyezését és elérési módját írja le.

Az adatbáziskezelő rendszerek tervezése, megvalósítása, fejlesztése és kezelése szempontjából fontos a fizikai és elvi struktúra megkülönböztetése, és ezáltal a munkamegosztás lehetővé tétele.

Az adatfüggetlenség mint az adatbáziskezelés egyik legfontosabb követelménye, a koncepcionális és a fizikai szint éles különválasztásának köszönhető. Megkülönböztetünk logikai és fizikai adatfüggetlenséget.

A logikai adatfüggetlenséget a metaadatok biztosítják számunkra, vagyis nemcsak az adatokat, hanem az adatok jellemzőit (pl. mezők helye és típusa) és az adatcsoportok közötti kapcsolatokat leíró adatokat is tároljuk. Az adatbáziskezelő rendszerek (DBMS) a logikai adatfüggetlenséget alaptulajdonságuknál fogva biztosítják. Az adatszerkezet megváltoztatása (más rekordszerkezet) csak a metaadatokban jelent változást, nem kell a felhasználói programot átírni. Lehetséges koncepcionális szinten változtatásokat végezni anélkül, hogy az adatmodell külső szintjében (nézet/view) változás történne.

Tekintsük a következő példát a logikai adatfüggetlenség könnyebb megértése végett. Tegyük fel, hogy egy általános célú nyilvántartó programot kell írnunk ANSI C nyelven. A felhasználó olyan programot szeretne, ami könyveit, papagályait és kerti virágait egyaránt kezelni tudja. Amíg egy adatbáziskezelő rendszer esetén a kérdés egyszerűen megoldható, addig ANSI C nyelven megírni egy rekordszerkezettől függetlenül használható alkalmazói programot sokkal nehezebb munka.

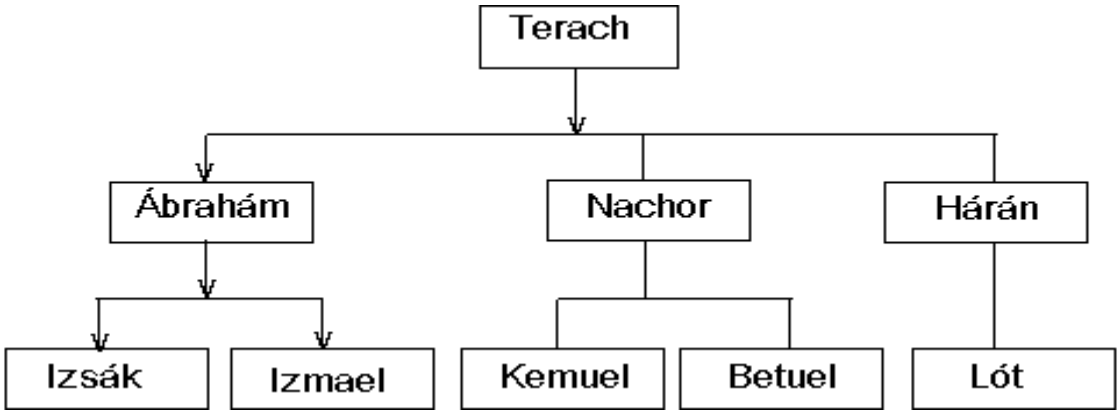
A DBMS ugyanakkor biztosítja a fizikai adatfüggetlenséget is. Az adattárolási szerkezetek és a hozzáférési módok - vagyis a fizikai adatbázis - változtatása nem vonja maga után a koncepcionális séma és az alkalmazói programok megváltozását.
Például egy indexelést a felhasználó legfeljebb oly módon észlel, hogy bizonyos adataihoz gyorsabban hozzáfér, mint korában.

3. LOGIKAI ADATMODELLEK

1) Hierarchikus modell (Hierarchy Model)

A hierarchikus modell *matematikai reprezentációja* a fa (tree), mely egy sajátos esete a gráfnak. A különböző szinten levő csomópontok (rekordok) között hierarchikus, szülő-gyerek kapcsolat van. Az adatfeldolgozási műveletek fa-struktúrák bejárását jelentik.
Pl.: családfa, osztályozás, termelési fázisok, szervezeti felépítés

A hierarchikus modell *grafikus reprezentációjára* nézzük a következő példát, melyből megtudhatjuk, hogy Ábrahám Terach gyereke, de ugyanakkor szülője Izsáknak és Izmaelnek:

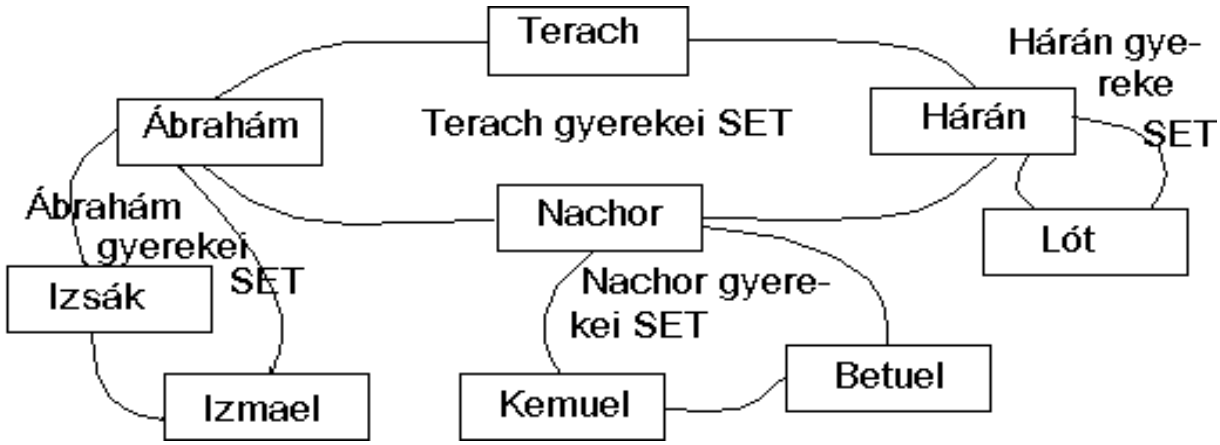


Az adatstruktúra elemei lényegében rekordok, amelyek nemcsak logikai, hanem a programozásból jól ismert mutatókon keresztül fizikai kapcsolatban is vannak egymással.

2) Hálós modell (Network Model)

A hálós modell az adatokat gyűrűsen kapcsolt rekordok hálózataként (set) ábrázolja, ahol az un. tulajdonos-rekordot (owner) körülveszik a tag-rekordok (member).
Matematikai reprezentációja: gráf (graph)
Pl.: úthálózat, számítógéphálózat, szervezeti felépítés

Az előző feladat hálós rekord-előfordulásait bemutató *grafikus reprezentáció*:



A lekérdezési műveletek a rekordok hálózatán értelmezett navigálási műveletekből állnak. Fizikai mutatók segítségével és megfelelő stratégia alkalmazásával gyorsan eljuthatunk a keresett rekordhoz.

3) Relációs modell (Relational Model)

A relációs modell lényegében táblázatok rendszere. Amíg a hierarchikus és hálós modell megvalósításában az egyes rekordok között fizikai kapcsolat van (mutatókon keresztül), addig a táblázatok (relációk) közötti kapcsolat logikai. Az adatbáziskezelő szoftverek számára specifikálni kell a logikai kapcsolatokat (metaadatok), melyek ismeretében a rendszer a logikai hivatkozást fizikai címre képes lefordítani.

Matematikai reprezentáció: relációkkal (relation)
(Reláció: a Descartes-szorzat egy részhalmaza) Pl.: rokonsági kapcsolatok (férj, feleség, hitves, gyerek, fiútestvér, lánytestvér ..), kiadási táblázat (pl. személyi kiadások az oszlopokban, hónapok a sorokban).

4) Entitás-Reláció modell (Entity-Relationship Model - ERM)

Specifikus abban az értelemben, hogy nagyon általános. A modell elemei:

Entitás: minden olyan objektum, amelyet egységként lehet azonosítani egy alkalmazáson belül, pl. személy, autó, árucikk. Az entitást téglalappal ábrázoljuk.

Attribútum: elemi tulajdonság, mely az adott entitást jellemzi. Az attribútumokat körrel ábrázoljuk.

Kulcsattribútum (key): olyan attribútum, melynek értéke egyértelműen azonosítja az adott entitást, vagyis az egyedtípus bármely előfordulását. Pl. az ÁRUCIKK entitás kulcsattribútuma lehet a ‘kód’ attribútum. Amennyiben nem létezik ilyen attribútum és a meglevők kombinálásával sem állítható elő, akkor esetenként az adatbázis-tervezőnek kell létrehoznia erre a célra még egy attribútumot, amely kulcsattribútumként használható.

Reláció: alkalmazás-specifikus kapcsolat, pl. IS_A $\Rightarrow$ Peter IS_A person. Opel IS_A car. A relációneveket deltoiddal ábrázoljuk.
Vonal, ív, nyíl: a modell elemeit köti össze.

PROJEKT 1.

A probléma leírása:

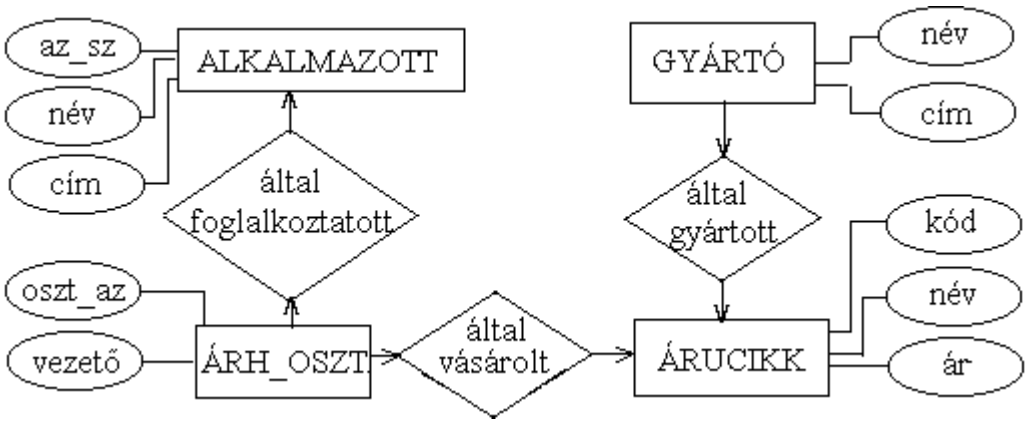
Tervezzünk ER modellt egy áruház számára a következő információk ismeretében:

- *Minden alkalmazott* szerepeljen a nyilvántartásban. Az alkalmazottak adatai: azonosító-szám, név, cím, áruházosztály - ahol dolgozik.
- *Minden áruházosztály* szerepeljen a nyilvántartásban. Az osztályok adatai: alkalmazottak, vezető, az osztály által árusított cikkek
- *Minden árucikk* szerepeljen a nyilvántartásban. Az árucikkek adatai: kódszám, gyártó, név, ár
- *Minden gyártó* szerepeljen a nyilvántartásban. A gyártók adatai: név, cím, árucikk - melyet az áruháznak szállít.

A probléma megoldása:

Mivel az áruházosztály nem mindig egyértelműen azonosítható a vezetőjével (például ideiglenesen lehet két osztálynak egy vezetője), ezért célszerű osztályazonosítót rendelni az ÁRH_OSZT attribútumhoz.

A fenti adatokat a következőképpen alakíthatjuk ER Modellé:



Megjegyzés

Az ER modell használatának előnyei:

- Elkerülhető a redundancia.

- Általánossága miatt kiváló eszköz elsődleges modellezésre.
- Lehetővé teszi a más rendszerekbe való transzformációt; átalakítható a modell.

Lehetővé teszi a mesterséges intelligenciában alkalmazott szakértői rendszerek tudásbázisába történő transzformálást, átalakítást.

4. FIZIKAI ADATSZERKEZETEK

Az adatbázis adatait *fizikailag* alkalmasan választott adathordozókon tároljuk. Az adathordozók közül egyelőre a mágneslemez tekinthető a legelterjedtebb eszköznek (bár a CD, DVD használata egyre inkább terjedőben van). Nagy adatbázisok esetén a ritkán használt adatokat rendszerint mágnesszalagokra mentik, és szükség esetén visszatöltik azokat. A következőekben csak a leggyakoribb adattárolási struktúrákkal és a legfontosabb hozzáférési módokkal foglalkozunk.

Fizikai adatbázis: fájlok rendszere
Fájl (file): rekordok együttese, halmaza
Rekord (record):

- lemezen tárolt értékek (effective record), egy-egy egyed jellemző adatait tartalmazza,
- mezőket tartalmaz. (rekord-struktúra leírása - record format)

A rekordoknak úgy kell elhelyezkedniük a lemezen, hogy könnyen és gyorsan elérhetők legyenek.

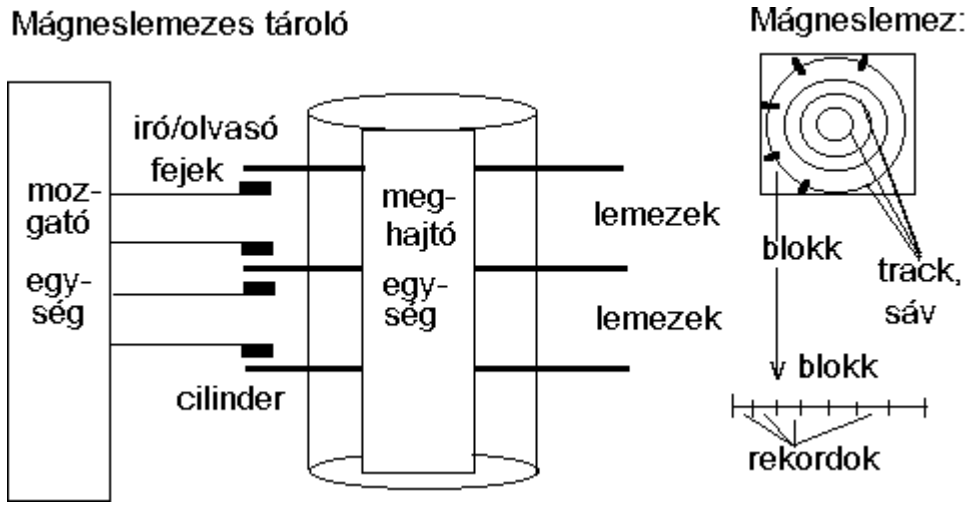
Mező (field):

- meghatározott típusú adatokat tartalmaz (numerikus, karakter, ..),
- hossza meghatározott

Rekordműveletek: módosítás, törlés, beszúrás, keresés

KÜLSŐ ADATTÁROLÓ ESZKÖZÖK

A mágneslemezes tárolóegységen azok az adatok könnyebben elérhetőek, amelyek azonos cilinderen vannak. Egy lemezegység (disc pack) több lemezből (disc surface) áll.



Az egyes lemezeken az adatokat keskeny koncentrikus körök mentén helyezik el. Egy ilyen kört sávnak (track) nevezünk.

A lemezegység ugyanolyan sugarú sávjai egy-egy cilindert alkotnak.

A sávok száma kb. 800 és az egyes sávok tárolási kapacitása 4-50 kbyte. Mivel a sávok információtartalma meglehetősen nagy, ezért a sávokat kisebb, azonos hosszúságú egységekre, ún. szektorokra vagy blokkokra (sector, block) osztják. Ezt a felosztást az operációs rendszer végzi el a lemez formázásakor. Egy blokk, amit lapnak(page) is neveznek, 512-4096 (a 2 n-dik hatványa, fix hosszú) byte információt hordozhat. [2]

A központi memória és a lemezegység közötti információátvitel egysége a blokk!

A blokk hardvercíme a lemezfelület, a sávszám és a blokkszám kombinációja. Olvasási művelet során a kívánt blokk a megfelelő pufferba kerül, íráskor pedig a puffer tartalma kerül a blokkba. A blokkok rekordokból állnak.

A rekordcím meghatározása: fizikai cím alapján vagy blokkcím és az offset alapján.

Az adatelérés folyamata több lépésből áll:

1. fejmozgatás: a megfelelő cilinderre állnak a fejek (lassú),
2. fejkiválasztás: a keresett lemezfelülethez tartozó fej (gyors),
3. forgási idő: a keresett rekord a fejhez kerül (közepes),
4. adatátvitel: elektronikus (a leggyorsabb művelet)

Az 1. és 3. lépés a leghosszabb idejű művelet. Amennyiben logikailag valamely mező szerint növekvő vagy csökkenő sorrendben következő rekordok fizikailag különböző cilinderen vannak, vagy ugyanazon cilinderen belül más-más, de nem egymás után következő blokkban helyezkednek el, a hozzáférés ideje jelentősen megnőhet. Az adatbáziskezelő rendszerek általában lehetővé teszik a rekordok fizikai rendezettségét is. [2]

Optikai lemezek

Vélhetőleg egyre nagyobb jelentőségük lesz az optikai lemezeknek, mert az utóbbi időben egyre olcsóbbak és jobbak lettek. Az optikai elven működő lemezeknél az írást és az olvasást egy lézersugár végzi.

Az optikai eljárásnál meg kell különböztetnünk analóg és digitális eljárást. *Analóg eljárás* szerint működnek a kép- és videolemezek (lézerlemez), amelyek főleg játékfilmek, videoszekvenciák, vagy egyes képek tárolására szolgálnak.

A *digitális technikát*, amellyel az audio-CD-k (CD=Compact Disc) a hagyományos lemezeket kezdik kiszorítani, a PC-kben is használják.

Az optikai tárolólemezeknél írhatóság és olvashatóság szerint három típust különböztetünk meg:

- a felhasználó által nem írható, csak olvasható (CD-ROM);
- a felhasználó által egyszer írható: a CD-Recordable és a WORM (Write Once Read Multiple), illetve a DRAW (Direct Read After Write);
- a felhasználó által többször írható (MOD=Magnetic-Optical Disc).

A fentiek közül részletesebben (a jegyzet terjedelmére való tekintettel) csak a CD-ROM-ot ismertetem:

Csak egyszer olvasható, és nem törölhető. Már kis darabszámban is kifizetődő az előállítása. Tárolókapacitása egyoldalas felvételnél 550-650 Mbyte.

A CD-lemez átmérője 12cm (4,75col). Az információkat egy spirál alakú, belülről kifelé vezető barázdákkal ellátott sáv tárolja, amelyeket a gyártásnál sajtolnak bele. Ezeket a mélyedéseket *pit*-eknek nevezzük, a fényvisszaverő alapot, amelyben ezek a pitek vannak, *land*-nak. A piteket egy áttetsző rétegen keresztül lehet letapogatni. A fény függőlegesen esik, és vagy visszaverődik a felületről, vagy nem. Ez a két eset felel meg a digitális jelek tárolásának. A visszaverődő fényt egy fotocella érzékeli. Ha a lézersugár egy bemélyedésre esik, akkor elterelődik, úgyhogy a fotocella nem kap fényt. Az alumínium réteget védőréteggel borítják. A CD anyaga szánszálas műanyag, 1,2mm vastag és csak az egyik oldalán vannak információk. Az olvasási sebessége állandó. Mivel a sáv belső átmérője kisebb, mint a külső, a forgási sebességet változtatni kell. Az elérési idő kb. 150-300ms között van. Egy pit mélysége 0,1 mikrométer, szélessége 0,6 mikrométer, és a sávok közti távolság 1 mikrométer. A pitek hossza 0,8 - 3,5 mikrométer között változik. A CD-audiolemezeket és a CD-ROM-okat ugyanúgy készítik. [7]

Digital Video Disc (DVD)

Egy-másfél évvel ezelőtt napvilágra kerültek a DVD technika (Digital Video Disc vagy Digital Versatile Disc) részletei, melyek óriási visszhangot keltettek és keltenek ma is. Azt hiszem, forradalmi változásoknak lehetünk szemtanúi az elkövetkező években - ezért fontosnak gondolom megemlíteni a fejlődés jelenlegi állását az adattárolás területén: a DVD-technológiát !

A DVD gyakorlati feljesztése már a nyolcvanas évek végén elkezdődött. Az akkor meghatározónak számító adathordozó-gyártóknak (IBM, 3M, Sony, Verbatim) szembe kellett nézniük a ténnyel, mely szerint a piacon kapható írható CD-k, lemezek, dat-kazetták és hordozható winchesterek néhány éven belül a programok óriási helyigényének köszönhetően elavulttá válnak, és nem lesznek képesek kielégíteni a felhasználók jogos igényeit. 1995-ben a Sony, a Philips és a Panasonic vezérletével megalakult a DVD-Consorcium, amely nyolc multinacionális céget olvasztott magába. A Consorcium által felállított lelkes kutatócsoport kétéves megfeszített munka után felhasználva a foto-CD, CD-ROM és a video-CD fejlesztésénél és használatánál szerzett tapasztalatait - egymás után hozta nyilvánosságra a kutatási eredményeket.

Az igazi újdonság a nagyobb tárolókapacitás és az olvasási gyorsaság volt. A megnövekedett tárolókapacitást kisebb pontmérettel és sűrűbb sávszerkezettel sikerült elérni. Ezenkívül a

lézerfény hullámhosszát is csökkentették: míg a hagyományos CD-meghajtók 780 nanométeres infravörös fénnyel dolgoznak, a DVD rendszer 635 nanométeren üzemel.
Jelenleg a legfejlettebbnek mondható a *17 gigabyte !!! tárolókapacitással* rendelkező DVD lemez.
A szakértők 2-3 éven belüli gyökeres átalakulást jósolnak nemcsak az adathordozó-, de a globális számítástechnikai és informatikai piacon is. [6]

Mutató/Pointer: fizikai cím (blokkra, rekordra mutat)
Ha egy adatstruktúra A elemének egy mezője egy másik B elem első szavának a címét tartalmazza, akkor azt mondjuk, hogy a mező a B elemre irányuló pointert tartalmaz. [3]

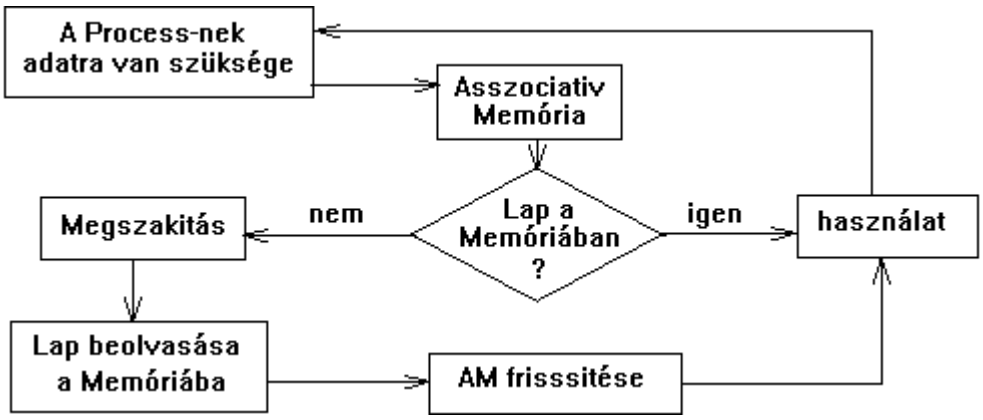
Virtuális Memória (VM)
Olyan rendszer, melyben egy folyamat munkaterülete részben a központi memóriában (High Speed Memory - HSM), részben a valamivel lassúbb háttértárolón (Backing-Store Device) van. [3]

Logikai cím (Logical Address - LA)
A cím alsó 12 bitje a lapon belüli helyet jelöli, míg a felső helyiértékű bit a lap számát jelenti. [3]

Asszociatív memória (AM) vagy tartalomcímzésű memória (Content-Addressable Memory)
Olyan tár, amely képes meghatározni, hogy egy bizonyos adat - a keresőszó - előfordul-e valamelyik címén (a memóriában levő lap címe) vagy tárhelyén (a mágneslemezen levő lapra mutató pointer). [3]

Process: végrehajtás alatt levő program

A virtuális memória kezelésének módja : lapozási technika (paging):



Ha a folyamat egy tárcímre hivatkozik, akkor a rendszer hardvere érzékeli, hogy a kívánt terület épp a központi tárban van-e, vagy sem. Ha nem, akkor megszakításkérést állít elő, ami lehetővé teszi a rendszerfelügyelőnek, hogy a háttértárolóból betöltse a tárba a megfelelő területet. A logikai címet használják az asszociatív tárban való keresésre, ill. annak eldöntésére, hogy a kívánt lap a központi tárban van-e. Ha nincs, akkor az operációs rendszer megkeresi a kérdéses lapot a háttértárolóban, átviszi a központi tárba, majd aktualizálja az asszociatív tárat. [3]

Az adatbáziskezelő műveletek sebessége függ:

- az algoritmus sebességétől: Θ, Ω, O, ... (N: a feldolgozandó adat mérete)
- a mágneslemez és a memória közötti átvitel idejétől (az átvitel egysége a blokk).

HASÍTOTT FÁJLOK (HASHED FILES)

Alapötlet: szervezzük a blokkokat 'bucket'-okba (bucket ~ vödör) !

Hogyan ? Hasító algoritmussal (Hashing Algorithm) !

H: V $\rightarrow$ B V: kulcsok halmaza; B: bucket-számok halmaza

H: v $\rightarrow$ b

H(v)=b v: egy adott kulcs; b:cím, egy bucket száma

Az eljárás lényege, hogy a kulcsmezőből alkalmasan választott függvénnyel címet képezünk, amely cím a rekord blokkjának fizikai helyét jelenti. A leképezés nem egy-egyértékű, ezért két eltérő kulcshoz ugyanaz az egész szám tartozhat.

A hashing néven elterjedt szervezési-hozzáférési módszer közvetlen hozzáférést biztosít, amennyiben a hashing-függvénnyel nyert címérték egyedi, vagyis nincs két olyan kulcsérték, amelyre a leképzés ugyanazt a címértéket adja.

A hashnig eljárás egyik legegyszerűbb változata:

Adattétel beszúrása:

A hasítóérték a bucket táblában elfoglalt elsődleges pozíciót jelöli. Ha ez a pozíció már foglalt, akkor a rákövetkező pozíciókat nézi végig mindaddig, amíg egy szabad helyet nem talál (a táblát cirkulárisnak tekinti). Az adattételt a hozzá tartozó kulccsal együtt ezen a helyen szúrja be a táblába.

Adattétel táblázatbeli lokalizálása:

Kiszámítja a kulcshoz tartozó hasítóértéket, és az annak megfelelő pozícióban levő táblabejegyzést megvizsgálja. Ha az ott talált kulcs megegyezik a keresett tétel kulcsával, a tételt lokalizálta; ha nem, a rákövetkező pozíciókat mindaddig vizsgálja, amíg egy megfelelő kulccsal rendelkező bejegyzést nem talál, vagy üres pozícióhoz nem ér. Ez utóbbi eset azt jelenti, hogy az adott táblában nem létezik a keresett kulcs, mivel a behelyező eljárás biztosan elhelyezte volna az üres pozícióban.

A módszer működéséhez elengedhetetlenül szükséges, hogy lényegesen több pozíció legyen a táblában, mint ahány tételt el kívánnak helyezni. Feltéve, hogy a táblának legfeljebb 60% -a kitöltött, egy tétel táblabeli lokalizálása legfeljebb két pozíció megvizsgálásával végrehajtható.

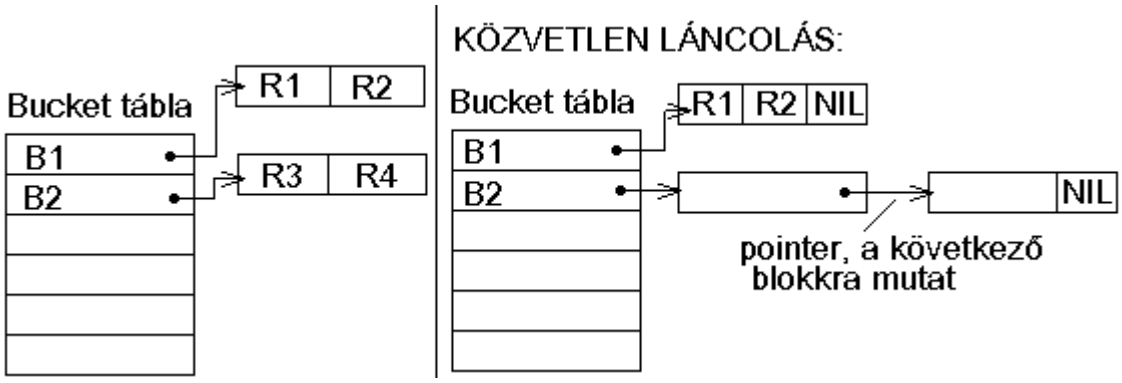
Kifinomultabb technikák alkalmazásával oldható meg az ütközés problémája, amely akkor lép fel, ha a hasítóérték által kijelölt pozíció már foglalt. E technikákkal a tábla átnézését még hatékonyabban lehet végrehajtani. [3]

Ütközés (Collision):

Ütközésről akkor beszélünk, amikor különböző kulcs-értékekhez ugyanazt a bucket-számot (címértéket) rendeljük.

Közvetlen láncolás (Direct Chaining) :

Egyszeresen láncolt listát használ az ütközés feloldására.



HASíTÓFÜGGVÉNYEK (HASHING FUNCTIONS)

- Több hasítófüggvény is létezik.

- A legáltalánosabban használt a moduló függvény. Pl.: 11(mod 3)=2 ; H(v)=v(mod m) ; v: a kulcsnak megfelelő egész típusú szám; m: bucketek száma

Fontos **követelmény**, hogy a *H* függvénnyel meghatározott blokkok **egyenletesen töltsék ki az *m* által meghatározott tárterületet (bucket táblát),** azaz ne maradjanak üres vagy alig kitöltött helyek.

Ennél is fontosabb követelmény, hogy ne legyen két olyan *v* érték, amelyre ugyanazt a *H* értéket kapjuk, azaz teljesüljön: H(K1) ≠ H(K2) feltétel. Más szóval **ne legyen ütközés (collision).** [2]

Hasonlítsuk össze a lineáris és a bináris keresést egy hash fájlban:

a) Lineáris keresés	**b) Bináris keresés**
soros keresés a bucket-táblában	bináris keresés a bucket táblában (mivel a bucketek növekvő sorba rendezettek)
soros keresés a bucketen belül	soros keresés a bucketen belül (ha a blokkok rendezve vannak, akkor a bináris keresés lehetséges a bucketen belül)

Hash függvény meghatározásának esetei:

- Legjobb eset: minden bucketban pontosan egy blokk van.
- Átlagos eset: a bucketekben több blokk van.
- A gyakorlatban rendszerint amikor új rekordokkal bővül az adatbázis, akkor bizonyos bucketek egyre több és több blokkot fognak tartalmazni.

PROJEKT 2.

A probléma leírása

Lehetséges már az adatbázis-tervezés elején olyan hash függvényt meghatározni, hogy az adatbázis növekedése során a későbbiekben is viszonylag egyenletes legyen a bucket-tábla kitöltöttsége, és ne kelljen a hash függvényt újratervezni ?

A probléma megoldása

A hash függvény tervezésekor a cél az, hogy a kulcsértékeket egyenletesen osszuk el a 0 és N-1 egész számok között, ahol N a bucket tábla mérete.
Egy lehetséges megközelítés:
Az *S* sztring *c1, ...,ck* karakterei alapján határozzunk meg egy *h* pozitív egész számot.

- a karakterenkénti konverziót az implementációs nyelvek használják
- a sztring karaktereinek értékét kombináljuk

Hogyan határozzuk meg a sztringet reprezentáló egész számot ?
Rossz módszer:

- csak az első, a középső és az utolsó karaktert vegyük figyelembe az adott sztringben
- a sztringben szereplő összes egész számot adjuk össze

Jobb módszer:

- legyen h(0)=0; h(i)=Ah(i-1)+c(i), ahol $1 \leq i \geq k$. Legyen h=hk, ahol a sztring hossza k...

h meghatározása után konvertáljuk azt egy 0 és N-1 közötti egész számmá, például osszuk el N-nel, és vegyük a maradékot. [4]

Más típusú hasítófüggvény pl. az f(k) függvény, amely adott A konstans esetén a k kulcshoz az Ak szorzat alacsonyabb helyiértékű felének első bitjeit rendeli. [3]

Nézzünk egy példát konkrét hash-függvények alkalmazási korlátaira:
Tegyük fel, hogy a következő 6 sztringre akarjuk alkalmazni a hash függvényt:
alma, banán, citrom, dinnye, körte, meggy. Ekkor egy hatékony hash függvény lenne, ha a sztringek első betűjét vennénk figyelembe. De abban az esetben, ha ezt az aaa0001, aaa0010,

aaa0011, aaa0100, aaa0101, aaa0110 sztringekre akarjuk alkalmazni, akkor a legrosszabb megoldást kapnánk eredményül, mivel mindegyik sztring egy bucketba kerülne.

Megjegyzés, vélemény

Tehát az adatok várható értékétől, és eloszlásától függ a bucket tábla kitöltöttsége. A jövőbeni adatok eloszlását pedig csak becsülni tudjuk. Az adatok egyenletes eloszlása esetén lehetséges jó hash függvényt megadni, sztochasztikus eloszlás esetén pedig lehetetlen. A fenti példa rámutat arra, hogy *nincs olyan hash függvény, amely minden alkalmazásban a legjobb eredményt adná.*

Sajnos *a legtöbb leképező algoritmus sem tesz eleget a bucket tábla egyenletes kitöltése és az ütközésmentesség követelményének. Legjobbnak mondható a következő maradékképző függvény*, mert aránylag egyenletesen tölti ki a tárterületet (bucket-táblát) és kevés az ütközés is.

H(v)=v (mod m), ahol

v a kulcsnak megfelelő egész típusú szám,

m pedig a tervezett bucketek számához legközelebb eső prímszám. [2]

Az ütközések kezelésére legmegfelelőbbek a dinamikus szerkezetek, mint pl. a láncolt lista, vagy faszerkezet.

A hash függvény néhány alkalmazási területe:

a) *Fordítóprogramok*

Az azonosítók véletlenszerűen jelennek meg a fordítóprogramokban, melyek száma programonként változó. A szimbólumtábla hasított fájl struktúrájú is lehet.
Hasítófüggvénye az azonosító első karaktere (- ez meghatározza a bucketek számát).

b) *Soundex kódolás*

Ez a kódolási módszer hasított fájl struktúrát használ.
Pl. a következő angol kifejezések kiejtése azonos: waits, waites, whaits, whates. A magyar nyelvben a tulajdonneveknél előfordulhat a következő eset:

a Desewfy, Dessewfy, Dezsőffy szavak kiejtése megegyezik.

A Soundex kódolásnál az egyforma kiejtésű nevekhez a hash függvény ugyanazt a bucket-számot rendeli. Hashing algoritmusa:

- a magánhangzókat nem vesszük figyelembe
- átugrunk a h-n és az y-on
- a többszörös hangzókat csak egyszer vesszük figyelembe
- a lényeges mássalhangzókat meghatározzuk, és kódoljuk.

Pl. a waits, waites, whaits, whates szavak Soundex kódja: W78; ahol T=7; S=8;
a Desewfy, Dessewfy, Dezsőffy szavak kódja pedig D56, ahol ‘zs’=5, ‘f’=6.

A Soundex-kódolás egy másik lehetséges alkalmazása:
Információ visszakeresés

- Keresés az Interneten (Yahoo, Altavista,...)
- DIALOG, European Space Agency, CARL (CARL kb. 40 adatbázist menedzsel, több mint félmillió indexet használ ...; a DIALOG és az ESA nem nyújt ingyenes szolgáltatást..)

Amíg a relációs adatbáziskezelésben általában a hashing ritkán fordul elő (inkább az indexelés terjedt el). addig a hierarchikus és hálós adatbázisokban jelentős szerepet kap. [2]

Most, a második lecke végén pedig leírok egy a témához kapcsolódó, a hash függvényekkel kapcsolatos viccet, melynek a magyarra fordításától most eltekintek:
ESR once asked a friend what he expected Berkeley to be like. The friend replied, ‘‘Well, I have this mental picture of naked women throwing Molotov cocktails, but I think that’s just a collision in my hash tables’’.

RELÁCIÓS ADATMODELL

Adottak a Di halmazok, ahol ; $2 \leq i \geq n$; N ə n; $n \geq 2$.

Descartes-szorzat:

$$\bigtimes_{i=2}^{n} D_i = D_2 \times D_3 \times \cdots \times D_n = \{(d_2, d_3, \ldots, d_n) \mid d_2 \in D_2, d_3 \in D_3, \ldots, d_n \in D_n\}$$

Ha a (d2, ... ,dn) halmaz

-1 elemű, akkor unárisnak *(unary tuple)*

-2 elemű, akkor binárisnak *(binary tuple)*

-3 elemű, akkor terciárisnak *(terciary tuple)*

-n elemű, akkor n-esnek (*n-tuple)* nevezzük.

Reláció: A fenti Descartes-szorzat bármely *R* részhalmaza; sorokból és oszlopokból álló táblázat.

Pl.: *Nevek* reláció:

biztosításszám	PIN kód	név
110	20	Cirmos Cica
111	30	Bodri Kutya
112	10	Vasorrú Bába

Címek reláció:

PIN kód	cím
10	Veszprém
20	Kecskemét
30	Pápa

(PIN= Personal Identification Number= személyi azonosító szám)

Attribútum: az oszlopok nevei

Értelmezési tartomány (Domain): az attribútum megengedhető értékei

Pl. a fenti *Nevek* reláció egy lehetséges értelmezési tartománya az összes lehetséges magyar név.

Tuple: a táblázat sorai

Reláció foka (Degree): az attribútumok (oszlopok) száma

Számosság (Cardinality): a tuple-k (sorok) száma

Relációs adatbázis: relációk (táblázatok) gyűjteménye

Az egyes kifejezések alternatívái:

mat. modell:	relációs modell:	fizikai modell:
reláció	= táblázat	= fájl(ha minden táblázat külön fájlban van)
tuple	= sor	= rekord egy fájlban
attribútum	= oszlop	= mező

Megjegyzés:

1. Az adatbázisoknál véges halmazokkal foglalkozunk (domain).
2. Lényegtelen az elemek sorrendje egy tuple-ban.

(A matematikában a rendezett pároknál az (1,2) és a (2,1) elemeket különbözőnek tekintjük - de az adatbázisnál nem!)

A reláció tulajdonságai:

- A relációk neve egyedi.
- Az attribútumok neve egyedi.
- Az attribútumok sorrendje lényegtelen.
- Nincs két egyforma sor.

A relációk kulcsai

Szuperkulcs (Superkey):

Az attribútumok olyan halmaza, mely egyértelműen azonosítja a sorokat. Pl. *Biztosításszám, (Biztosításszám, PIN kód)*

Kandidát kulcs (Candidate key):

Olyan szuperkulcs, melynek nincs egyedi azonosítót tartalmazó valós részhalmaza. (Nem tartalmaz redundáns attribútumokat, és a sorok egyértelműen azonosíthatók.) Pl. *Biztosításszám*

Az egyediség tulajdonsága irreducibilis (nem lehet csökkenteni).

Elsődleges kulcs (Primary key):

Olyan kandidát kulcs, melyet kiválasztunk a sorok azonosítására. Pl. a *Biztosításszám* és a *PIN* kód is a *Nevek* reláció egy-egy kandidát kulcsa. Mi döntjük el, melyiket választjuk ki elsődleges kulcsnak!

Idegen kulcs (Foreign key):

Olyan attribútum, melynek értékei megegyeznek egy másik reláció kandidát kulcsának az értékeivel.

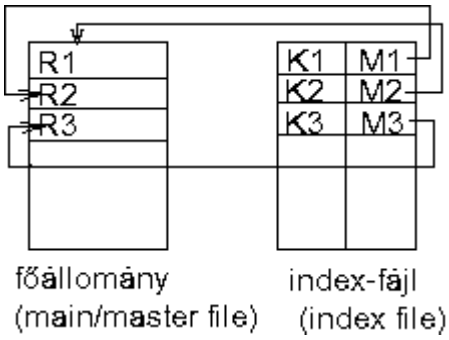
Pl. a *Biztosításszám* a *Nevek* reláció elsődleges kulcsa,

a *PIN kód* a *Nevek* reláció kandidát kulcsa,

a *PIN kód* a *Címek* reláció idegen és elsődleges kulcsa is egyben.

INDEXELÉS (Dense Index)

Hasznos és elterjedt fájl-szerkezet.
Például az Internetes keresőprogramok, vagy a California Research Library (CARL) adatbázis-szolgáltatással foglalkoznak. Adatbázisaikat egyidejűleg több helyről kérdezik le a világban.
Lekérdezésekor fontos tényező a válaszidő (átl.:30-60 sec).A gyors lekérdezést indexelt struktúrával (is) meg lehet oldani.



A főállományban a rekordok az érkezés sorrendjében vannak tárolva (sequential file).
Az index-fájl tartalmazza a főállomány összes rekordjának a kulcsait, és rájuk vonatkozó mutatókat. Többszörös indexelésről beszélünk, ha az indexfájlnak is van indexfájlja.

Konzisztencia-feltételek, melyeket minden adatbázisnak ki kell elégíteni:

1) Tartománybeli integritás elve (domain integrity): az attribútumok meghatározott értékeket vehetnek fel.
2) Entitás-integritás elve (entity integrity): egyetlen elsődleges kulcs attribútuma sem lehet NULL. A NULL ismeretlen, vagy definiálatlan éréket reprezentál
3) Hivatkozás-integritás leve (referential integrity): az idegen kulcs értéke vagy NULL, vagy megegyezik a megfelelő kandidát kulcs értékével.
4) Enterprise integrity: egyéb feltételek, megkötések pl. a kapcsolatokat illetően

PROJEKT 4

KAPCSOLAT AZ ER-MODELL ÉS A RELÁCIÓS MODELL KÖZÖTT

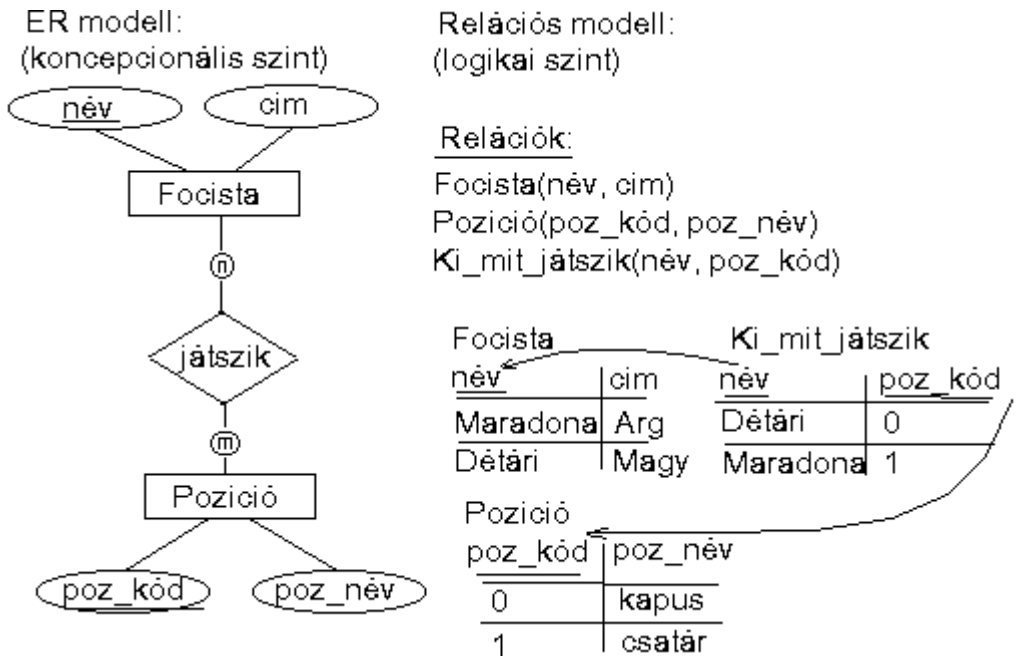
Feladat:

Adjuk meg a következő nyilvántartás logikai szintű relációs modelljét !

Focisták adatainak nyilvántartásánál az ER-modell adatai:

Focista entitás attribútumai: *név, cím*

Pozíció entitás attribútumai: *poz_kód, poz_név.*



Az ábrán aláhúzással jelöltem az elsődleges kulcsokat. Az ER-modell két entitása n:m kapcsolatban van egymással, vagyis egy focista több pozícióban is játszhat, és egy pozícióban több focista is lehet.

A leképezési szabályok alkalmazása után kapjuk a Relációs modell három relációját (tábláját). A *Ki_mit_játszik* reláció mindkét attribútuma elsődleges kulcs és idegen kulcs is egyben, melyek összetett kulcsot alkotnak. Az ábrán nyíllal jelölt módon a *Ki_mit_játszik* reláció

idegen *név* kulcsa a *Focista* reláció elsődleges kulcsa, a *Focista* reláció *poz_kód* kulcsa pedig a *Pozíció* reláció elsődleges kulcsa.

MATEMATIKAI LOGIKA (KLASSZIKUS)

ITÉLETKALKULUS (PROPOSITIONAL CALCULUS)

Itélet (Proposition)

Az ítélet egy egyszerű kijelentés, amely vagy igaz (true) vagy hamis (false).
Nincs harmadik lehetőség, vagyis *Tertium non datur (lat.).* Pl.: 'A nap nem süt.'
Az 'Odanézz!' vagy a 'Jani ezermester a Déli Sarkon.' kijelentések viszont már nem ítéletek, mivel ezek igaz vagy hamis voltát nem tudjuk eldönteni!

Itéletkalkulus (Propositional Calculus)

Jelölés: A, B, P, Q... de: T: igaz (true), F: hamis (false)

- Negálás: ¬, ~
- Konjunkció = logikai ÉS művelet: ∧, &
- Diszjunkció = logikai VAGY művelet: ∨
- Implikáció, következtetés, szubjunkció: →, ⇒
- Ekvivalencia: ⇔
- Ellentmondás (contradiction): P ∧ ¬P. Értéke: F.
- Tautológia: mindig igaz állítás, a komponensei igaz vagy hamis voltától függetlenül.

A műveletek igazságtáblái:

P	Q	¬P	P ∧ Q	P ∨ Q	P ⇒ Q	P ⇔ Q
T	T	F	T	T	T	T
T	F	F	F	T	F	F
F	T	T	F	T	T	F
F	F	T	F	F	T	T

Példa a tautológiára: $P \wedge (P \Rightarrow Q) \Rightarrow Q$

$P \wedge Q \Rightarrow P$

$P \Rightarrow P \vee Q$

$P \Rightarrow P$

KÖVETKEZTETÉSI SZABÁLYOK (INFERENCE RULES)

1) Ellentmondásra való visszavezetés (Reduction to absurd) /Reductio ad absurdum -lat./

$\neg A \Rightarrow B$

$\neg A \Rightarrow \neg B$

A

Használata: Azt bizonyítjuk, hogy A igaz.

1) Tegyük fel, hogy $\neg A$ igaz.
2) Próbáljunk ellnetmondást keresni! ($B \wedge \neg B$ következik az 1)-ből)
3) A igaz

2) Modus Ponens

P

$P \Rightarrow Q$

Q Pl.: $P \wedge (P \Rightarrow Q) \Rightarrow Q$

3) Modus Tollens

$P \Rightarrow Q$

$\neg Q$

$\neg P$

PREDIKÁTUM KALKULUS

Predikátum (Predicate): kijelentés, mely legalább egy változót tartalmaz

Igaz vagy hamis volta azoktól az ítéletektől függ (proposition), melyeket konkrét értékek behelyettesítésével kapunk. Jelölése: A(x)

Pl.:Minden diáklány szép. A szemüveget viselő tanárok nem jól látnak.

Ezek a kijelentések nem ítéletek (proposition), mert több személyt jelentenek.

Kvantorok (Quantifiers)

a) Univerzális kvantor: $\forall$ (minden ..)

b) Létezési (existential) kvantor: $\exists$ (létezik ..)

Pl.: $(\forall x)\ A(x)$ or $(\forall x)\ A$; $(\exists x)\ A(x)$ or $(\exists x)\ A$

Nem mindig könnyű egy mondatból predikátumot készíteni ! Pl.:

a) Nem minden arany, ami fénylik.
b) Minden matróznak van barátnője.
c) Minden 3000 m feletti afrikai országban az évi átlagos csapadékmennyiség a legtöbb európai ország átlagos évi csapadéékmennyisége fölött van.

a) (∃tárgy,ami fénylik) ¬arany(tárgy)

b) (∀matróz): ∃ barátnó (a matróznak)

c) (∃ európai ország) évi átl. csapadékmennyiség < (∀3000m feletti afrikai ország évi átl. csapadékmennyisége) és

(∃ európai ország) évi átl. csapadékmennyiség ≥ (∀3000m feletti afrikai ország évi átl. csapadékmennyisége). A *legtöbb*-et nehéz kifejezni !

A predikátum kalkulus tulajdonságai:

a) (∀x) A T ⇒ (∀x) ¬A F

b) (∃x) A F ⇒ (∃x) ¬A T

c) ¬(∃x) A ⇔ (∀x) ¬A

d) ¬(∀x) A ⇔ (∃x) ¬A

e) (∀x) A F ⇒ (∀x) ¬A nem tudjuk eldönteni

f) (∃x) A T ⇒ (∃x) ¬A nem tudjuk eldönteni

Pl.: Van tanár a teremben. (∃ tanár) teremben T

A fenti állítás alapján nem tudjuk eldönteni az alábbi állítás igaz vagy hamis voltát:

Van tanár a termen kívül. (∃ tanár) ¬ teremben

RELÁCIÓS ALGEBRA

A következő relációk alkalmazásával mutatjuk be a relációs algebrai műveleteket.

R Reláció		
A	**B**	**C**
a	b	c
d	a	f
c	b	d

S Reláció		
D	**E**	**F**
b	g	a
d	a	f

Unió:

$\bigcup_{i=1}^{n} R_i$ Feltétel: az Ri relációk (táblák) aritása (oszlopainak száma) megegyezzen!

Akkor van értelme két reláció unióját venni, ha azonos típusú adatok vannak a táblák egymásnak megfelelő oszlopaiban ! Például az életkor és a hobbi attribútumok uniójának nincs értelme.

Különbség (Difference): R -S

Feltétel: a relációk aritása (oszlopainak száma) megegyezzen!

Ha több egyforma sor előfordul, azok közül csak egyet hagyunk meg. A fejlécbe az első (itt R) reláció attribútumait írjuk!

A fenti relációk felhasználásával kapjuk RUS -t: és R-S -t

R U S		
AUD	BUE	CUF
a	b	c
d	a	f

R -S		
A	B	C
a	b	c
c	b	d

c	b	d
b	g	a

Descartes-szorzat (Cartesian Product)

RxS = { (t u) | t∈R, u∈S, aritás(R)=k1, aritás(S)=k2, aritás(RxS)=k1+k2 }

A fenti relációk alapján:

R x S					
A	B	C	D	E	F
a	b	c	b	g	a
a	b	c	d	a	f
d	a	f	b	g	a
d	a	f	d	a	f
c	b	d	b	g	a
c	b	d	d	a	f

Projekció (Projection)

$\pi p1,...,pm\ (R) = \{ (a1 \ldots am) \mid \exists (b1 \ldots bk) \in R: aj=bpj, \forall j\ \ 1 \leq j \leq m; k=aritás(R) \}$

Szelekció, Kiválaszáts (Selection) σF

Az F feltétel prédikátum ! /operandusz(attr. vagy értékek), operátor(aritm., logikai)/

Példa:

$\pi A,C\ (R)$	
A	**C**
a	c
d	f
c	d

$\sigma B=b(R)$		
A	**B**	**C**
a	b	c
c	b	d

$\sigma F \neq c(S)$		
D	**E**	**F**
b	g	a
d	a	f

Tulajdonságok:

1) $\sigma F\ (\pi P(R)\ ) = \varnothing$, ha F-nek és P-nek nics közös része (attribútuma).
2) $\pi P\ (\sigma F(R)\ ) = \varnothing$, ha $\sigma F\ (R)=\varnothing$, egyébként pedig nem üreshalmaz.
3) $\sigma F\ (\pi P(R)\ ) = \pi P\ (\sigma F\ (R)\ )$, ha P-nek és F-nek van közös része.

Példák a fenti R és S reláció alkalmazásával:

1) $\sigma B = b\ (\pi A,C(R)\ ) = \varnothing$
2) $\pi A,C\ (\sigma B = c(R)\ ) = \varnothing$, ahol $\sigma B = c(R) = \varnothing$,
3) $\sigma A = d\ (\pi A,C(R)\ ) = \pi A(\sigma A = d(R)\ ) = [d\ f]$

Példa:

”DIÁK” reláció			
NÉV	SZÜL_ÉV	NEMZETISÉG	NEM
Mary	1971	német	nő
Anna	1972	magyar	nő
George	1970	francia	ffi
Sophia	1972	bolgár	nő
Olga	1971	magyar	nő

Anton	1973	német	ffi

Szelekció, kiválasztásσF(R)

Adjuk meg minden diáklány adatát !

Ekkor R=”DIÁK”, F=nő, σ =adjuk meg az adatokat. SQL leírással:

SELECT * FROM DIÁK **WHERE** NEM=nő

Projekció πP(R)

Adjuk meg az összes diák nevét és nemzetiségét!

πNÉV, NEMZETISÉG (DIÁK)

SQL: **SELECT** NÉV, NEMZETISÉG **FROM** DIÁK

Kombináció (σ,π)

Adjuk meg az 1971 után született diákok nevét és születési évszámát!

πA (σB(R)) = σB (πA(R)), ha A-nak és B-nek van közös része

A feladatot többféleképpen is megoldhatjuk:

A) πNÉV,SZÜL_ÉV(σSZÜL_ÉV>1971(DIÁK)) SQL:

CREATE VIEW 1971 **AS SELECT * FROM** DIÁK **WHERE** SZÜL_ÉV>1971

SELECT NÉV,SZÜL_ÉV **FROM** 1971

B) σSZÜL_ÉV>1971(πNÉV,SZÜL_ÉV(DIÁK)) SQL:

CREATE VIEW X **AS SELECT** NÉV,SZÜL_ÉV **FROM** DIÁK

SELECT * FROM X **WHERE** SZÜL_ÉV>1971

C)SQL:

SELECT NÉV,SZÜL_ÉV **FROM** DIÁK **WHERE** SZÜL_ÉV>1971

Produktum, Descartes-szorzat

Adjuk meg az összes lehetséges fiú-lány párost! FIÚ x LÁNY

CREATE VIEW LÁNY **AS SELECT * FROM** DIÁK **WHERE** NEM=‘nő’

CREATE VIEW FIÚ **AS SELECT * FROM** DIÁK **WHERE** NEM=‘ffi’

SELECT LÁNY.NÉV,FIÚ.NÉV **FROM** LÁNY,FIÚ

Különbség

Adjuk meg azon lányok neveit, akik nem franciák!

A)**SELECT** NÉV **FROM** DIÁK **WHERE** NEM=‘nő’ **AND** NEMZETISÉG≠‘francia’

A ” NEMZETISÉG≠‘francia’ ” kifejezéssel ekvivalens: NEMZETISÉG **NOT IN** (‘francia’)

B) πNÉV (σNEM=‘nő’ **AND** NEMZETISÉG≠‘francia’(DIÁK)),

Unió

SELECT * FROM LÁNY

UNION

SELECT * FROM FIÚ

Metszet

$R \cap S = R-(R-S)$

Hányados

$R \div S = \{ t \mid \pi t(\sigma tu \in R, \forall u \in S(R)) \}$

‘R’ reláció			
a	**b**	**c**	**d**
a	**b**	**e**	**f**
b	c	e	f
e	**d**	**c**	**d**
e	**d**	**e**	**f**
a	b	d	e

‘S’ reláció	
c	d
e	f

R ÷S	
a	b
e	d

Példa:

Az alábbi LECKE és TANTÁRGY relációk alapján adjuk meg azon diákok neveit, akik két tárgyat tanulnak!

LECKE	
NÉV	TÁRGY
Anna	kémia
Gyurka	kémia
Gyurka	matek
Maris	matek
Olga	kémia

TANTÁRGY
kémia
matek

LECKE ÷ TANTÁRGY SQL:

SELECT NÉV **FROM** LECKE **GROUP BY** NÉV **HAVING count**(*)=2

A művelet eredményeként Gyurkát kapjuk, aki mindkét tárgyat tanulja.

A count() -nál az összel lehetséges tantárgy számát kell megadni, akkor kapjuk a fenti osztást eredményül.

Az osztás tulajdonságai:

a) *R-((R÷S)xS)=Q, ahol Q=maradék (remainder); R=((R÷S)xS) U Q*

A probléma megfogalmazása:

Adottak az A B, C relációk. Adjuk meg D-t, ha D÷B=A, a maradék:C ! D= (AxB) U C

Példa:

A	
a	b
c	d

B
e
f

C

AxB U C = D		
a	b	e
a	b	f
c	d	e
c	d	f
i	j	k

i	j	k
c	d	e

Könnyen ellenőrizhetjük, hogy D÷C=A

b) *Q=∅ ⇒ A=C÷B ⇔ C= AxB (=BxA)*

c) *Legyen A és B olyan reláció, melyre teljesül: B ⊆ A, és C=A-B*

Az osztásra vonatkozó kérdést ált. az alábbi módon tesszük fel:

Adjunk meg minden c∈ C -t, mely tartalmaz minden b∈ B -t.

Példa: C=NÉV; B=TÁRGY; A=NÉV&TÁRGY. Ekkor a lépések:

1. T1=πC(B)
2. T2 = πC((A x T1) - A)
3. R ÷ S = T1 - T2

d) *Adottak az A, B relációk, ahol B ⊂ A*

Osztás: $R \div S = \pi C(A) - \pi C((B \times \pi C(A)) - A)$

Az osztás maradéka: $Q = A - ((\pi C(A) - \pi C((B \times \pi C(A)) - A)) \times B)$

Megjegyzés:

a. A fenti relációk ugyanazt a maradékot adják, mintha az osztást végeztük volna el a relációkon.

b. Nyitott kérdés: adott S relációval való osztás esetén a maradékok száma !

JOIN ⊗F

F feltételre vonatkozó egyesítés: $R \otimes FS = \sigma F(R \times S)$

R		
A	B	C
1	2	3
4	5	6
7	8	9

S	
D	E
3	1
6	2

R⊗B< DS				
A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2

NATURAL JOIN ⊗

A relációkat úgy egyesítjük, hogy az egyező attribútumokat (oszlopokat) csak egyszer íratjuk ki.

$R \otimes S = \pi i (\sigma F(R \times S))$; $F = \wedge i (R.Ai = S.Ai)$; R.Ai = Az R reláció i-dik attribútuma

R		
A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

S		
B	C	D
b	c	d
b	c	e
a	d	b

R⊗S			
A	B	C	D
a	b	c	d
b	b	c	e
d	b	c	d
d	b	c	e
c	a	d	b

Relációs algebra	**Reláció-kalkulus (klasszikus logika)**

R U S	$\{ t \mid t \in R \vee t \in S \}$
R - S	$\{ t \mid t \in R \wedge \neg t \in S \}$
R x S	$\{ t \mid t = (u,v), u \in R \wedge v \in S \}$
$\pi Ai(R)$	$\{ t \mid t \in Ai \in R \}$
$\sigma F(R)$	$\{ t \mid t \in R \wedge F(t)$; /$t \in F(t)$: amikor az F feltétel igaz lesz/ $\}$
$R \div S = \pi t(\sigma tu \in R, \forall u \in S(R))$	$\{ t \mid t \in R \wedge F(u,v); (u,v) \in R, \forall v \in S \}$
$R \otimes FS = \sigma F(RxS)$	$\{ t \mid t \in ((u,v) \wedge F(u,v)), u \in R \wedge v \in S \}$
$R \otimes S = \pi i (\sigma F(RxS))$; $F = \wedge i (R.Ai = S.Ai)$	$\{ t \mid t \in ((u,v) \wedge F(u,v)), u \in R \wedge v \in S$; $F = \wedge i (R.Ai = S.Ai) \}$

PROJEKT 4.

Egy alkalmas pédán mutassuk be a következő relációs algebrai műveletek használatát szavakban, algebrai formában, és esetleg SQL-ben is :
szelekció, projekció, Descartes-szorzat, különbség, unió, hányados, kombináció (szelekció és projekció), join, natural join..

A már korábban is használt áruházas példán mutatom be a fenti relációs algebrai műveleteket. Bár ezek közül soknak nincs értelme, ezért inkább csak a műveletek megértésére szolgáltatnak példát.

Tekintsük az áruházat reprezentáló alábbi relációkat:

ALKALMAZOTT					
alk_az	alk_vez	alk_ker	alk_vár	alk_ut	alk_hsz
A19	Piros	Piroska	Tab	Fő	1
A20	Fekete	Anna	Öskü	Nagy	3
A21	Zöld	Xénia	Márkó	Széles	2
A22	Kis	Virág	Lenti	Kis	2

ÁRH_OSZT		
oszt_az	vez_az	o_nev
o1	A21	Élelmiszer
o2	A22	Virág
o3	A21	Ital

DOLGOZIK	
alk_az	oszt_az
A19	o2
A20	o3
A21	o1
A21	o3
A22	o2

VÁSÁRLÁS	
oszt_az	áru_kód
o1	TT221
o2	VA315
o1	TK329
o3	IB238
o3	IW114

GYÁRTÁS	
gy_kód	áru_kód
G217	TT107
G217	TT221
G332	TF241
G444	TK329
G523	IB187
G523	IB238
G523	IW114
G422	VI105
G422	VA315
G422	VP222

GYÁRTÓ				
gy_kód	gy_név	gy_vár	gy_ut	gy_hsz
G217	Fincsi Sütöde	Tab	Nagy	21
G332	Veszprémtej Rt.	Veszprém	Külső	7
G444	Bajor Pékség	Budapest	Petőfi	9
G523	Zwack	Budapest	Kossuth	41
G422	Kerekes Éva	Tab	Görbe	2
G231	Virágh József	Tab	Fő	9

ÁRU		
áru_kód	áru_név	ár
TT107	sajtos kifli	14
TT221	zsömle	8
TK329	rozskenyér	96
TF241	tejföl	49
IB187	Bólé	312
IB238	Barackpálinka	672
IW114	Whisky	356
VI105	Ibolya	218
VA315	Amarilisz	725
VP222	Páfrány	427

A fenti relációk felhasználásával adjunk példát a relációs algebrai műveletekre !

Descartes-szorzat

Adjunk meg egy olyan táblázatot, melyben minden dolgozó minden áruházosztályon dolgozik ! (Ez hasonló az iraki kibucok munkmódszeréhez.)

Az ALKALMAZOTT és a z ÁRH_OSZT relációk Descartes-szorzata:

ALKALMAZOTT x ÁRH_OSZT :

alk_az	alk_vez	alk_ker	alk_var	alk_u	alk_hsz	arh_az	vez_az	o_nev
A19	Piros	Piroska	Tab	Fő	1	o1	A21	élelmiszer
A19	Piros	Piroska	Tab	Fő	1	o2	A22	virág
A19	Piros	Piroska	Tab	Fő	1	o3	A21	ital
A20	Fekete	Anna	Öskü	Nagy	3	o1	A21	élelmiszer
A20	Fekete	Anna	Öskü	Nagy	3	o2	A22	virág
A20	Fekete	Anna	Öskü	Nagy	3	o3	A21	ital
A21	Zöld	Xénia	Márkó	Széles	2	o1	A21	élelmiszer
A21	Zöld	Xénia	Márkó	Széles	2	o2	A22	virág
A21	Zöld	Xénia	Márkó	Széles	2	o3	A21	ital
A22	Kis	Virág	Lenti	Kis	2	o1	A21	élelmiszer

A22	Kis	Virág	Lenti	Kis	2	o2	A22	virág
A22	Kis	Virág	Lenti	Kis	2	o3	A21	ital

Szelekció

Mely budapesti gyártól vásárol az áruház termékeket?

σ gy_név = 'Budapest' (GYÁRTÓ)

gy_kód	gy_név	gy_vár	gy_ut	gy_hsz
G444	Bajor Pékség	Budapest	Petőfi	9
G523	Zwack	Budapest	Kossuth	41

Projekció: Irassuk ki az áruházban levő osztályok nevét és vezetőjének azonosítóját!

πo_név, vez_az (ÁRH_OSZT)

SQL: **SELECT** o_nev, vez_az **FROM** ÁRH_OSZT

o_név	vez_az
élelmiszer	A21
virág	A22
ital	A21

Kombináció: Adjuk meg azon termékek nevét és árát, melyek 100,- Ft-nál nem kerülnek többe ! πáru_név, ár(σár ≤ 100 (ÁRU))

πáru_név, ár(σár ≤ 100 (ÁRU))	
áru_név	ár
sajtos kifli	14
zsömle	8
rozskenyér	96
tejföl	49

Unió

Hozzunk létre két veiw-t: a 'VEZETŐK' nevű tartalmazza az osztályok vezetőinek azonosítószámát és nevét, a 'MUNKÁSOK' nevű tartalmazza azon alkalmazottak azonosítószámát és nevét, akik nem vezetők! Ezt követően adjuk meg a 'VEZETŐK' és 'MUNKÁSOK' view unióját !

CREATE VIEW VEZETŐK(alk_az, alk_vez, alk_ker) **AS**
SELECT DISTINCT alk_az, alk_vez, alk_ker **FROM** ALKALMAZOTT
WHERE ÁRH_OSZT.vez_az=ALKALMAZOTT.alk_az

Megjegyzés: A DISTINCT kulcsszó az ismétlődések kiszűrésére szolgál.

CREATE VIEW MUNKÁSOK(alk_az, alk_vez, alk_ker) **AS**

SELECT alk_az, alk_vez, alk_ker **FROM** ALKALMAZOTT

WHERE NOT ÁRH_OSZT.vez_az=ALKALMAZOTT.alk_az

VEZETŐK view		
alk_az	alk_vez	alk_ker
A21	Zöld	Xénia
A22	Kis	Virág

MUNKÁSOK view		
alk_az	alk_vez	alk_ker
A19	Piros	Piroska
A20	Fekete	Anna

VEZETŐK ∪ MUNKÁSOK:

CREATE VIEW ALK_LISTA(alk_az, alk_vez, alk_ker) **AS**

SELECT * FROM VEZETŐK

UNION

SELECT * FROM MUNKÁSOK

ALK_LISTA view		
alk_az	alk_vez	alk_ker
A21	Zöld	Xénia
A22	Kis	Virág
A19	Piros	Piroska
A20	Fekete	Anna

Különbség

Adjuk meg azon alkalmazottak adatait, akik nem Tabon laknak!

SELECT * FROM ALKALMAZOTT

WHERE 'Tab' NOT IN alk_vár

alk_az	alk_ker	alk_vez	alk_vár	alk-ut	alk_hsz
A20	Fekete	Anna	Öskü	Nagy	3
A21	Zöld	Xénia	Márkó	Széles	2
A22	Kis	Virág	Lenti	Kis	2

Hányados

Először adjunk meg egy AKCIÓS_ÁRU nevű relációt, melyben a zsömlét és a tejfölt szerepeltetjük, mivel az áremelések ellenére a mi áruházunk még nem változtatott azok árain, tehát más boltokhoz képest azokat igen kedvező áron tudjuk kínálni ! Ezt követően határozzuk meg azon árukat, melyeket nem akciós áron kínálunk !

CREATE VIEW AKCIÓS_ÁRU(áru_kód, áru_név, ár) **AS**

SELECT * FROM ÁRU

WHERE áru_kód=TT221 **OR** áru_kód=TF241

AKCIÓS_ÁRU view		
áru_kód	áru_név	ár
TT221	zsömle	8
TF241	tejföl	49

ÁRU ÷ AKCIÓS_ÁRU :

SELECT * FROM ÁRU

WHERE NOT AKCIÓS_ÁRU.áru_kód = ÁRU.áru_kód

áru_kód	áru_név	ár
TT107	sajtos kifli	14
TK329	rozskenyér	96
IB187	Bólé	312
IB238	Barackpálinka	672
IW114	Whisky	1356
VI105	Ibolya	218
VA315	Amarilisz	725
VP222	Páfrány	427

Join

Az áruházban szeretnénk egy '100Forintos' osztályt kialakítani. Mely cégektől kell árut rendelnünk az új osztályra?

Határozzuk meg, mely gyártótól vásároltunk már 100 Ft alatti terméket, és melyek voltak azok, és mennyi az áruk !

GYÁRTÓ⊗ÁRU.ár≤100 ÁRU = σÁRU.ár≤100(GYÁRTÓxÁRU) :

SELECT * FROM GYÁRTÓ, ÁRU

WHERE ÁRU.ár ≤ 100 **AND** GYÁRTÁS.áru_kód = ÁRU.áru_kód **AND**

GYÁRTÁS.gy_kód = GYÁRTÓ.gy_kód

gy_kód	gy_név	gy_vár	gy_ut	gy_hsz	áru_kód	áru_név	ár
G217	Fincsi Sütöde	Tab	Nagy	21	TT107	sajtos kifli	14
G217	Fincsi Sütöde	Tab	Nagy	21	TT221	zsömle	8
G444	Bajor Pékség	Budapest	Petőfi	9	TK329	rozskenyér	96
G332	Veszprémtej Rt.	Veszprém	Külső	7	TF241	tejföl	49

Natural Join

A Natural Join relációs algebrai művelet segítségével írassuk ki, hogy az áruház egyes osztályai mely termékeket árulják (az árukód elegendő) !

ÁRH_OSZT⊗VÁSÁRLÁS :

SELECT * FROM ÁRH_OSZT, VÁSÁRLÁS

WHERE VÁSÁRLÁS.oszt_az = ÁRH_OSZT.oszt.az **GROUP BY** oszt_az

oszt_az	vez_az	o_név	áru_kód
o1	A21	Élelmiszer	TT221
o1	A21	Élelmiszer	TK329
o2	A22	Virág	VA315
o3	A21	Ital	IB238
o3	A21	Ital	IW114

A RELÁCIÓS ALGEBRAI MŰVELETEK TULAJDONSÁGAI

I. Kommutativitás:

a) RxS = SxR.

Bizonyítás: RxS =/def./= { t | t=(u,v), u∈R ∧ v∈S }=/∧komm./= { t | t=(u,v), u∈S ∧ v∈R } =/def./= SxR

b) R⊗FS = S⊗FR.

Bizonyítás: R⊗FS =/def./= σF(RxS) =/x komm./= σF(SxR) =/def./= S⊗FR

c) R⊗S = S⊗R.

Bizonyítás: R⊗S =/def./=πi (σF(RxS))=/ σF komm./= πi (σF(SxR))=/def./= R⊗S

(F= ∧i (R.Ai = S.Ai)

II. Asszociativitás:

a) (RxS)xT = Rx(SxT).

Bizonyítás: (RxS)xT =/def./= { t | t=(u,v), u∈RxS ∧ v∈T } = /def./={ t | t=(p,o,v), (p∈R ∧ o∈S) ∧ v∈T} =/∧ asszoc./= { t | t=(p,o,v), p∈R ∧ (o∈S ∧ v∈T)} = { t | t=(p,q), p∈R ∧ q∈SxT } =/def./= Rx(SxT)

b) (R⊗FS) ⊗FT = R⊗F (S⊗FT)

Bizonyítás: (R⊗FS) ⊗FT = /def./= { t | t ∈((u,v)∧ F(u,v)), u∈R⊗FS ∧ v∈T } =/def./= { t | t ∈ ((p,o,v)∧ F(p,o,v)), (p∈R ∧ o∈S) ∧ v∈T } =/∧ asszoc./= { t | t∈ ((p,o,v)∧ F(p,o,v)), p∈R ∧ (o∈S ∧ v∈T)} = { t | t∈((p,q) ∧F(p,q)), p∈R ∧ q∈S⊗FT } =/def./= R⊗F(S⊗FT)

c) (R⊗S)⊗T = R⊗(S⊗T)

Bizonyítás: (R⊗S) ⊗T = /def./= { t | t ∈((u,v)∧ F(u,v)), u∈R⊗S ∧ v∈T; F= ∧i (R⊗S.Ai = T.Ai) } =/def./= { t | t ∈ ((p,o,v)∧ F(p,o,v)), (p∈R ∧ o∈S) ∧ v∈T; F= ∧i (R. Ai =S.Ai = T.Ai) } =/∧ asszoc./= { t | t∈ ((p,o,v)∧ F(p,o,v)), p∈R ∧ (o∈S ∧ v∈T); F= ∧i (R. Ai = (S.Ai = T.Ai)) } = { t | t∈((p,q)∧ F(p,q)), p∈R ∧ q∈S⊗T ; F= ∧i (R. Ai = S⊗T.Ai) } =/def./= R⊗ (S⊗T)

III. Kaszkádolás

projekciók kaszkádolása: $\pi A1(... \pi Ai (... \pi An(R) ...) ...) = \pi A1(R)$

ahol $A1 \subseteq A2 \subseteq .... \subseteq Ai \subseteq .... \subseteq An$ (attribútumok)

Bizonyítás: n=2, $A1 \subseteq A2 \subseteq R$; $\pi A1(\pi A2(R))$=/def./= $\pi A1\{ (...ai...) \in A2 \mid \exists (...bj...) \in R: ai= bj \}= \pi A1(R)$

szelekciók kaszkádolása: $\sigma \wedge Fi (i=1...n) (R) = \sigma F1(... \sigma Fn(R))$

Bizonyítás: $\sigma \wedge Fi (i=1...n) (R)$ =/def./= $\{ t \mid t \in R \wedge (\wedge(i=1..n)Fi(t)) \}$ = /asszoc./= $\{ t \mid t \in R \wedge (F1 \wedge (\wedge(i=2...n)Fi(t))) \}$ =/asszoc./= $\{ t \mid (t \in R \wedge F1) \wedge (\wedge(i=2...n)Fi(t)) \}$ =/def./= $\sigma F1(\sigma \wedge(i=2...n) (R))$

IV. A szelekció és a Descartes-szorzat kommutativitása

a) Adott: A,B, F=F(A); Ekkor: $\sigma F(AxB) = \sigma F (A) \times B$

Bizonyítás: $\sigma F(AxB) = \{ t \mid t=(u,v), u \in A \wedge v \in B \wedge F(t) \}$ =/$\wedge$asszoc.,F(t)=F(u,v)=F(u)/= $\{ t \mid t=(u,v), (u \in A \wedge F(u)) \wedge v \in B \} = \sigma F(A) \times B$

b) Adott: $F=F1 \wedge F2$, A,B, F1= F1(A), F2= F2(B); Ekkor: $\sigma F(AxB) = \sigma F1(A) \times \sigma F2(B)$

Bizonyítás: $\sigma F(AxB) = \{ t \mid t=(u,v), u \in A \wedge v \in B \wedge F1(u) \wedge F2(v)\}$ =/$\wedge$asszoc./= $\{ t \mid t=(u,v), (u \in A \wedge F1(u)) \wedge (v \in B \wedge F2(v)) \} = \sigma F1(A) \times \sigma F2(B)$

c) Adott: $F=F1 \wedge F2$, A,B, F1= F1(A), F2= F2(A,B); Ekkor: $\sigma F(AxB) = \sigma F2 (\sigma F1(A) \times B)$

Bizonyítás: $\sigma F(AxB) = \{ t \mid t=(u,v), u \in A \wedge v \in B \wedge F1(u) \wedge F2(u,v)\}$ =/$\wedge$asszoc./= $\{ t \mid t=(u,v), ((u \in A \wedge F1(u)) \wedge v \in B) \wedge F2(u,v) \} = \sigma F2 (\sigma F1(A) \times B)$

V. Attribútumok unióján végzett szelekció

$\sigma F(A \cup B) = \sigma F (A) \cup \sigma F (B)$

Bizonyítás: $\sigma F(A \cup B) = \sigma F (\{ t \mid t \in A \vee t \in B\}) = \{ t \mid (t \in A \vee t \in B) \wedge F(t)\}$=/disztr./= $\{ t \mid (t \in A \wedge F) \vee (t \in B \wedge F)\} = \sigma F (A) \cup \sigma F (B)$

VI. Relációk Descartes-szorzatának projekciója

Adott: R,S, R=R(B), S=S(C), A=BC; Ekkor: $\pi A (R \times S) = \pi B (R) \times \pi C (S)$

Bizonyítás: π_A (RxS) = π_A ({ t | t=(u,v), u∈R ∧ v∈S }) = { q | q= (p,o), p ⊆ u ∧ o ⊆ v ∧ u∈R ∧ v∈S} = { q | q=(p,o), (p⊆u∈R) ∧ (o⊆v∈S) } = π_B (R) x π_C (S)

VII. Relációk uniójának projekciója

π_{Ai} (R U S) = π_{Ai} (R) U π_{Ai} (S)

Bizonyítás: π_{Ai} (R U S) = { q | q ⊆ t ∧ (t∈T v t∈S) } = { q | (q ⊆ t ∈R) v (q ⊆ t∈S) } = π_{Ai} (R) U π_{Ai} (S)

A fenti tulajdonságok felhasználásával csökkenthetjük egy adatbázison végzett szűrés idejét.

Példa:

Q: Írassuk ki minden 100 éves fa nevét, ha az alábbi relációk adottak:

R	
A	B
név	fa-kód

S	
C	D
fa-kód	életkor

A Q: $\pi_A(\sigma_{B=C \wedge D=100}$ (RxS) utasítást kétféleképpen értékelhetjük ki:

a) Először RxS -t számítjuk ki. Ekkor a

N1 = az R relációban levő sorok száma

B1 = rekord/blokk R

N2 = az S relációban levő sorok száma

B2 = rekord/blokk S

jelölések bevezetésével kapjuk az adathozzáférési idő becslését: (N1/B1)x((N2/B2)xB1).

Konkrét adatokat felhasználva nézzük meg a következő esetet:

N1 = 50 000

B1 = 10

N2 = 100 000

B2 = 5 Ekkor a végrehajtási időre kapott becslés: 14 óra !!

b) F2 = „B=C” = F2 (B,C) = F2 (R,S); F1 = „D=100’ = F1(D) = F1 (S)

$\sigma_{F1 \wedge F2}(RxS) = \sigma_{F2}(\sigma_{F1}(R) xS)$ felhasználásával kapjuk:

Q: $\pi_A(\sigma_{B= C} (\sigma_{D=100} (S) xR))$, az adatelérési idő a lemezen: B1(N2/B2); α(N2/B)

D=100, $\alpha \leq$ B1 - kevesebb időt igényel !

Ötlet:

Ha az S reláció D attribútumát indexeljük, akkor közvetlen eléréssel kapjuk meg a fenti feladat kívánt eredményét, ami az adatelérési idő lényeges csökkenését eredményezi !

Tehát az indexek helyes használata nagyon fontos az adatbázistervezésben.

FUNKCIONÁLIS FÜGGŐSÉG

Euklides

posztulátum : magától értetődő állítás, pl. 'a pontnak nincs kiterjedése'

axióma : Olyan állítás, melyet bizonyítás nélkül elfogadunk igaznak egy bizonyos területen belül, pl. A Paralellák axiómája: egy síkbeli egyeneshez egy rajta kívüllálló pontból pontosan egy egyenes húzható.

Gauss, Bolyai Farkas, Bolyai János, Lobacsevszkij szintén foglalkoztak a paralellákkal, de nem tudták sem az axiómát, sem az ellenkezőjét bizonyítani.

A logikai elegancia követelményei:

- az axiómák legyenek egymástól függetlenek,
- az axiómák legyenek ellentmondásmentesek,
- az axiómarendszer teljes legyen:
  egy adott terület minden állítása levezethető legyen csupán csak az axiómákból. A levezetéshez használjuk a logika eszközeit, és a már levezetett állításokat.

Míg a relációs algebra alapja az adatbázislekérdező nyelveknek, addig a funkcionális függőség pedig az adatbázistervezés alapját képezi.

1) Funkcionális függőség (Functional Dependencies)

Adottak: A= {A1, ..., An}, R(A), X, Z⊆A

Ekkor az X→Y jelentése: X meghatározza Y-t, vagy Y függ X-től.

Az X→Y egy injektív függvény (nem szurjektív, és nem is bijektív !). A funkcionális függőségek meghatározása az adatbázistervező feladata, nincsenek rá szabályok, ez az ő választásától függ.

Megjegyzés:

a) Adottak X,Y halmazok. Ekkor X UY ≠ XY=YX

b) Adott r. Ekkor, ha r kielégíti → -t, azt 'r→' -vel jelöljük. Ha adott az X→Y funkcionális függőség, és r kielégíti X-et és Y-t, azt rx, ry -nal jelöljük.

Példa1:

Legyen A={ A1, A2..., A30 }, ahol {A1, A2, A3 }⊆X⊆R és {A10, A11, ...,A30 }⊆Y⊆R. Ekkor Y minden sora egyértelműen meghatározott az X sorai által.

Példa2:

Tekintsük az X→Y függvényt, ahol $Y=X^2$. Ekkor X={1,2,3} esetén Y={1,4,9}. Láthatjuk, hogy itt Y elemeit egyértelműen meghatározzák X elemei.

Példa3:

Az R(város,utca,ir-szám) relációban igazak a következő (egyértelmű) függőségek:

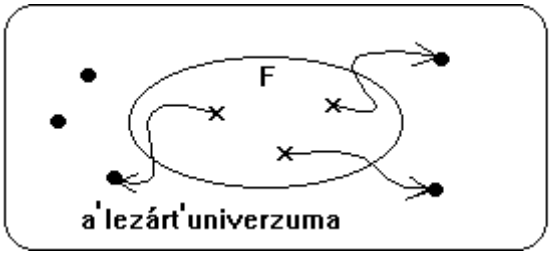
ir-szám→város, és város&utca→ir-szám (ez inkább Angliára jellemző):

2) A függőségek lezártja (Closure of Dependencies)

Legyen F={d1, ... ,dm} a függőségek halmaza, ~> pedig a logikai implikáció. Ekkor

F ~> d, ha ($\forall r \in R: r_F$) $\Rightarrow$ rd

(r_F : r kielégíti az F halmaz feltételeit; $\Rightarrow$ az implikáció jele)



Az F halmaz lezártja: F^+ ={ d | F~>d}

Ha F= F^+ , akkor F 'teljes családot' jelent, vagyis minden tagot tartalmaz.

3) Kulcs (Key)

Adott az R(A) reláció és az F függőség. Ekkor X$\subseteq$A egy kulcs, ha

1) (X$\rightarrow$A) $\in$ F^+

2) $\neg\exists$ Y $\subset$ X : (Y $\rightarrow$A) $\in$ F^+

4) A funkcionális függőség axiómái

1970: CODD

1974: Armstrong

1980: Ullmann

Legyen U_F az attribúrumok univerzális halmaza, X,Y,Z:attribútumok halmaza.

A1: REFLEXIVITÁS (REFLEXIVITY) : (Z $\subseteq$ X $\subseteq$ U) $\Rightarrow$ (F ~> (X$\rightarrow$Y))

A2: KITERJESZTÉS (AUGMENTATION) : X$\rightarrow$Y, Z$\subseteq$U $\Rightarrow$ XZ$\rightarrow$YZ

A3: TRANZITIVITÁS (TRANSITIVITY) : X$\rightarrow$Y, Y$\rightarrow$Z $\Rightarrow$ X$\rightarrow$Z

Minden más függőséget bizonyíthatunk e 3 axióma felhasználásával.

Példa:Adott: Ir-szám$\rightarrow$Város; Kiterjesztés: Ir-szám Utca$\rightarrow$Város Utca

5) Teljesség

Bizonyítsuk be, hogy a fenti axiómarendszer teljes !

a) helyesség (soundness)

b) megfelelőség (adequacy)

Helyesség ellenőrzése

minden függőség, mely az axiómákból levezethető, igaz: $F\ A\sim> (X\rightarrow Y) \Rightarrow X\rightarrow Y, \forall r_F$.

Az A1 axióma helyességének ellenőrzése ellentmondásra való visszavezetéssel:

Tegyük fel, hogy $\neg r_{X\rightarrow Y} \Leftrightarrow \exists\ t,u\in r: r_X \wedge \neg r_Y \Rightarrow$ ellentmond $Y\subseteq X$ -nek; $t_X = u_X \Rightarrow t_Y = u_Y$

Az A2 axióma helyességének ellenőrzése ellentmondásra való visszavezetéssel:
Tegyük fel, hogy $X \rightarrow Y$, $Z \subseteq U$ és $\neg XZ \rightarrow YZ$, vagyis $\exists\, t,u \in R$: $tu_{XZ} \wedge \neg tu_{YZ} =$
$(tu_{XZ} \Rightarrow tu_X \wedge tu_Z) \wedge (\neg tu_{YZ} \Rightarrow \neg tu_Y \wedge \neg tu_Z)$, ahol $\neg tu_Y$ ellentmond $X \rightarrow Y$ -nak.

Az A3 axióma helyességének ellenőrzése ellentmondásra való visszavezetéssel:
Legeyenk $X \rightarrow Y$, $Y \rightarrow Z$ funkcionális függőségek. Tegyük fel, hogy $\neg X \rightarrow Z$.
Ekkor $\exists\, t,u \in r \in R$: $tu_X \wedge \neg tu_Z$ esetén vagy tu_Y -t ($Y \rightarrow Z$), vagy $\neg tu_Y$ -t ($X \rightarrow Y$) sértettük meg.
Mivel mindhárom esetben ellentmondáshoz jutottunk, így a feltevéseink hamisak voltak, vagyis az axiómák helyesek.

6) Lemma1

Tulajdonságok:
I. Unio-szabály: $(X \rightarrow Y) \wedge (Y \rightarrow Z) \Rightarrow (X \rightarrow YZ)$
Bizonyítás: a kiterjesztés szabályát alkamazva kapjuk: $(X \rightarrow Y) \Rightarrow (X \rightarrow YX)$ és $(X \rightarrow Z) \Rightarrow (XY \rightarrow ZY)$. A tranzitivitás szabályából következik: $(X \rightarrow ZY) \Rightarrow (X \rightarrow YZ)$.

II. Pszeudotranzitivitási szabály: $(X \rightarrow Y) \wedge (WY \rightarrow Z) \Rightarrow (XW \rightarrow Z)$
Bizonyítás: : Az $(X \rightarrow Y) \Rightarrow (XW \rightarrow YW)$ és $(WY \rightarrow Z) \Rightarrow (XY \rightarrow ZY)$ szabályok tranzitivitását felhasználva kapjuk: $XW \rightarrow Z$

III. Dekompozíciós szabály: $(X \rightarrow Y) \wedge (Z \subseteq Y) \Rightarrow (X \rightarrow Z)$
Bizonyítás: : A $Z \subseteq Y$ reflexivitása miatt teljesül: $Y \rightarrow Z$. Mivel $X \rightarrow Y$ és $Y \rightarrow Z$, így a tranzitivitást felhasználva kapjuk $X \rightarrow Z$ -t. Pl.: $Y=\{a,b,c\}$ és $Z=\{a,b\}$.

7) Az attribútumok lezártja (closure of attributes)

$$F_{U_i},\ X \subseteq U:\quad X_F^+ = \{\, A \mid F \rightsquigarrow (X \rightarrow A)\}$$

8) Lemma2

F A~> $(X \rightarrow Y) \Leftrightarrow Y \subseteq X^+$ (F^+ és X^+ közötti kapcsolat meghatározása)

Bizonyítás:

a) $\Leftarrow$:

$Y \subseteq X^+$, Y=A1,A2,....An $\Rightarrow$ $X \rightarrow Ai, \forall i \Rightarrow X \rightarrow \cup i\ Ai,=Y$

b) $\Rightarrow$:

F A~> $(X \rightarrow Y) \Rightarrow X \rightarrow Y$=A1,A2,....An $\Rightarrow X \rightarrow Ai, \forall i \Rightarrow Ai \in X^+ \Rightarrow \cup i\ Ai \subseteq X^+ \Rightarrow Y \subseteq X^+$

9) Megfelelőség (adequacy) ellenőrzése

Amikor az F funkcionális függésből az axiómák felhasználásával nem vezethetünk le egy adott függőségi viszonyt (amely egyébként lehet igaz), akkor F -ből logikailag nem következik az.

Adott F, és $X \rightarrow Y$ -t nem kapjuk meg A1-A3 axiómák felhasználásával, vagyis

$$F \not\vdash (X \rightarrow Y),$$

mely szerint $\exists r$: rd $\forall d \in F$, de $\neg\ rX \rightarrow Y$.

Bizonyítás:

Adott F, $X \rightarrow Y$. Tekintsük X^+ -t , ahol r = $X^+ \cup${más, X^+ -tól és az üres halmaztól eltérő attribútumok halmaza}.

a) Mutassuk meg, hogy rd , $\forall d \in F$ (ellentmondásra való visszavezetéssel) !

$\exists\ V \rightarrow W \in F : \neg\ rv \rightarrow w \Rightarrow X \subseteq X^+$ (különben $\neg$ rv teljesülne) vagy rv $\wedge \neg$ rw, $W \not\subset X^+$ (különben rw teljesülne).

Tekintsük $A \in W$, $A \notin X^+$ -t. Ekkor

$V \subseteq X^+ \Rightarrow$/lemma2/$\Rightarrow X \rightarrow V$, továbbá tudjuk, hogy $V \rightarrow W$, a tranzitivitásból következően $X \rightarrow W$. $A \in W$ reflexivitásából következik $W \rightarrow A$. Az $X \rightarrow W$ és $W \rightarrow A$ szabályok tranzitivitása eredményezi: $X \rightarrow A$, vagyis (lemma2) $A \in X^+$, ami ellentmond a fenti $A \notin X^+$ feltevésnek.

b) Mutassuk meg, hogy $\neg$ rv$\rightarrow$ w (ellentmondásra való visszavezetéssel!)

Tegyük fel, hogy r$X \rightarrow Y \Rightarrow Y \subseteq X^+$ (különben $\neg$ rY) $\Rightarrow$/lemma2/$\Rightarrow X \rightarrow Y \Leftrightarrow$ ha $X \rightarrow Y$ származtatható az A1-A3 axiómákból, ami ellentmond a fenti feltevésünknek.

Mivel bebizonyítottuk, hogy az axiómák rendszere helyes (soundness) és megfelelő (adequacy), ezért az teljes.

Megjegyzés:

GÖDEL tétele: Minden rendszer tartalmaz olyan állítást, amelyről sem az állítás helyességét, sem annak ellenkezőjét nem tudjuk bebizonyítani az adott rendszer keretein belül. (Ez nem azt jelenti, hogy nem lehet azokat egyáltalán bebizonyítani!)

10) F^+ és X^+ kiszámítása

a) F^+

F={d1, ... , dn}={$A \rightarrow Bi$, $n \geq i \geq 1$}: az adatbázis-tervező által meghatározott függőségek

$F^+ = \{A \rightarrow Y \mid Y \in \wp(\{Bi\}\ n \geq i \geq 1)\} \Rightarrow |F^+| = 2^n$

Futási idő: $O(2^n)$ (NP probléma)

b) X^+ algoritmus

1) $X^{(0)} := X$

2) $X^{(i+1)} := X \cup A$, $\exists d = Y \rightarrow Z \in F : A \subseteq Z \wedge Y \subseteq X^{(i)}$

Mivel $X=X^{(0)} \subseteq X^{(1)} \subseteq ... \subseteq X^{(i)} \subseteq ... X^{(0)} \subseteq U$, $|U| <$ végtelen, ezért $\exists i_0 : X^{(i0)} = X^{(i0+1)}$.

Ezek szerint $X^+ = X^{(i0)}$

Példa:

F: AB→C, C→A, BC→D, ACD→B, D→EG, BE→C, CG→BD, CE→AG.

Legyen X=BD. Határozzuk meg X lezártját, X^+ -t !

1) $X^{(0)} = X = BD$

2) Olyan függőségeket keresünk, melyeknek a bal oldalát $X^{(0)}$ attribútumai alkotják.

Mivel D→EG, ezért $X^{(1)} = X^{(0)} \cup EG = BDEG$

3) Rekurzívan ismételjük meg a 2) lépést ! Mivel D→EG és BE→C, így $X^{(2)} = X^{(1)} \cup EG \cup C = BCDEG$

4) Mivel C→A, BC→D, D→EG, BE→C, CG→BD, CE→AG, így $X^{(3)} = ABCDEG = X^+$

NORMALIZÁLÁS

Tekintsük a következő példát:

SZÁLLÍTÓK reláció:

név	**cím**	**áru**	**ár**
n1	a1	i1	p1
n2	a2	i2	p2
n1	a1	i3	p3
n2	a2	i1	pk

Ebben a relációban előfordul:

1. redundancia: több egyező sor van az adatbázis valamely attribútumában (oszlopában)
2. szabálytalanság, anomália:
   - ha n1-et megváltoztatjuk az első sorban, akkor a 3. sorban levő n1 inkonzisztenciát fog eredményezni.
   - n2 törlése inkonzisztenciát fog eredményezni.

Ezen problémák elkerülése végett normalizálni kell az adatbázist!

A normalizáció egyes lépéseit a következő adatbázison nézzük meg.

ügyf_az	ügyf_név	ing_az	ing_cím	bérlés _kezd	bérlés _bef	bérl _dij	tul_az	tul_név
CR76	Peti	PG4 PG16	Hawai Veszpr	98szept 75márc	1999dec 2002jun	300 10000	C56 C40	HajniPeti PetiHajni
CR56	Hajni	PG4 PG36 PG26	Hawai Athén Nápoly	98szept 92szept 96jul	1999jul 92dec 97okt	300 25 100	C56 C76 C56	HajniPeti HajniHajni HajniPeti

Azt a relációt, melyben van olyan attribútum, melynek ismétlődő sorai vannak, normalizálatlan relációnak nevezzük. (**UNF=UnNormalized Form**)

Normalizálás: olyan folyamat, mely egy relációt (ált.) több relációba transzformál úgy, hogy azáltal megszünteti a redundanciát és a szabálytalanságokat (anomáliákat), és az új relációk megfeleltethetők az eredeti relációnak.

1NF: Első Normál Forma

Pontosan egy adat szerepelhet bármely sor és oszlop metszetében.

A fenti relációban 1.sor ∩ 3.oszlop = {PG4, PG16} .. tehát ez a reláció UNF alakú.

UNF→NF átalakítási szabály:

Meg kell szüntetni a többértékű mezőket úgy, hogy a többértékű mező minden adatához a táblázat egy sorát rendeljük:

ügyf_az	ügyf_név	*ing_az*	ing_cím	bérlés_ kezd	bérlés _bef	bérl_dij	tul_az	tul_név
CR76	Peti	PG4	Hawai	98szept	1999dec	300	C56	HajniPeti
CR76	Peti	PG16	Veszpr	75márc	2002jun	10000	C40	PetiHajni
CR56	Hajni	PG4	Hawai	98szept	1999jul	300	C56	HajniPeti
CR56	Hajni	PG36	Athén	92szept	92dec	25	C76	HajniHajni
CR56	Hajni	PG26	Nápoly	96jul	97okt	100	C56	HajniPeti

2NF: Második Normál Forma

Az 1NF relációból - meghatározott módon - több relációt hozunk létre:

R1 reláció			
ügyf_az	*ing_az*	bérlés_ kezd	bérlés_ bef
CR76	PG4	98szept	1999dec
CR76	PG16	75márc	2002jun
CR56	PG4	98szept	1999jul
CR56	PG36	92szept	92dec
CR56	PG26	96jul	97okt

R2 reláció				
ing_az	ing_cím	bérl_dij	tul_az	tul_név
PG4	Hawai	300	C56	HajniPeti
PG16	Veszpr	10000	C40	PetiHajni
PG36	Athén	25	C76	HajniHajni
PG26	Nápoly	100	C56	HajniPeti

R3 reláció	
ügyf_az	ügyf_név
CR76	Peti
CR56	Hajni

Kérdés: Az így kapott relációk mindig egyértelműen megfeleltethetők az eredeti relációnak ?
A 2NF a függőségeket és a kulcsattribútumokat veszi figyelembe.

Egy reláció 2NF formában van, ha

- 1NF formában van és
- minden nem kulcsattribútum finkcionálisan teljeseen függ az elsődleges kulcstól. (teljes bijekció, egyértelmű meghatározás)

1NF→2NF átalakítási szabály

Távolítsuk el a részleges függőségeket: ezen attribútumokat új relációba helyezzük az őket meghatározó attribútumokkal együtt.

Megjegyzés:

1.A létrejött új relációknál *mindig* felmerül a fent említett kérdés.

2. Az új relációk megőriznek *minden* függőséget?

Példa a függőségekre:

Elsődleges kulcs: ügyf_az, ing_az

1) ügyf_az,ing_az → bérlés_kezd, bérlés_bef (teljes függőség)

2) ügyf_az → ügyf_név (részleges függőség)

3) ing_az → ing_cím, bérl_dij, tul_az, tul_név (részleges függőség)

4) tul_az → tul_név (A tul_az nem elsődleges kulcs, ezért nem hozunk létre itt új relációt!) (tranzitivitás)

3NF Harmadik Nomál Forma

Egy reláció 3NF-ben van, ha:

- 1NF-ben és 2NF-ben van és
- funkcionális függés csak az elsődleges kulcsból indul ki, vagyis megszüntettük a közvetett (tranzitív) függéseket.

2NF→3NF átalakítási szabály

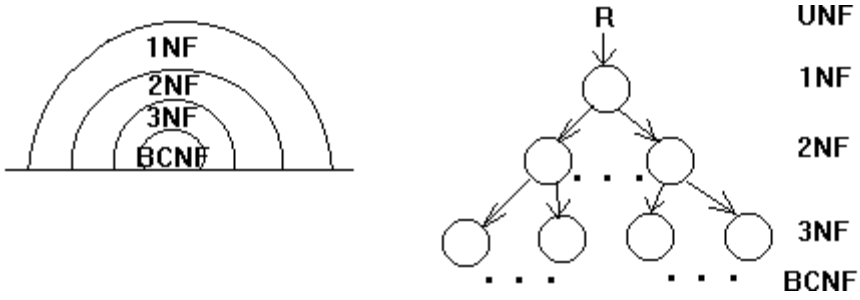
Távolítsunk el minden közvetett függést az adott relációból úgy, hogy az elsődleges kulcstól közvetetten függő attribútumokat új relációba helyezzük azok meghatározó (kulcs) attribútumaival együtt.

A fenti példában tranzitív függés van az R2 relációban, melyet az alábbi módon hozhatunk 3NF-re:

ing_az	**ing_cím**	**bérl_dij**	**tul_az**
PG4	Hawai	300	C56
PG16	Veszpr	10000	C40
PG36	Athén	25	C76
PG26	Nápoly	100	C56

tul_az	**tul_név**
C56	HajniPeti
C40	PetiHajni
C76	HajniHajni

A normálformák hierarchiája:



Az ábrából kiovasható, hogy minden 3NF-ban levő reláció 2NF-ben és 1NF-ben is van, de ez fordítva nem igaz.

Boyce-Codd NF (BCNF)

Egy reláció Boyce-Codd normál formában van, ha 3NF-ben van, és minden függőséget egy kandidát kulcs határoz meg. A BCNF forma létrehozásánál a kandidát kulcsokat vesszük figyelembe.

3NF→BCNF leképezési szabály

A 3NF-ben levő relációt további relációkra bontjuk úgy, hogy minden függőséget egy kandidát kulcs határozzon meg.

Példa: Tekintsük az R relációt, melynek attribútumai: ABCDE, és a következő függőségeket ismerjük:

AB→CDE, DBC→A, DB→E.

Kandidát kulcsok: AB, DBC (összetett kulcsok)

Elsődleges kulcs: AB, továbbá R 3NF-ben van!

A DB→E függőség miatt az R relációt az alábbi 2 relációra kell bontanunk:

R1: DBE

R2: ABCD

Ekkor DB elsődleges kulcs R1-ben, AB pedig elsődleges kulcs R2-ben. R2 attribútumai az R/R1 attribútumai , valamint R1 elsődleges kulcsa (DB). Az így kapott R1 és R2 relációk tehát már BCNF-ben vannak.

Vezessük végig az egyes normalizációs lépéseket a következő példán:

Albérletközvetítő Iroda ingatlannyilvántartási formanyomtatványa a következő adatokat tartalmazza:

dátum, ingatlanszám, ingatlan címe, megtekintés ideje, személyzet kódja és neve, autó rendszáma, megjegyzés.

1) Az R reláció (UNF forma):

ing_az	ing_cím	dátum	időpont	megjegyzés	szem_az	szem_név	rendszám
PG4	Valahol	98jun2	98jun28	vélemény1	S56	Jean	HVG113
		98máj14	98jun5	vélemény2	S42	Joe	XYZ999

2) UNF→1NF átalakítás:

Minden sor és oszlop metszetében csak egy adat szerepelhet.

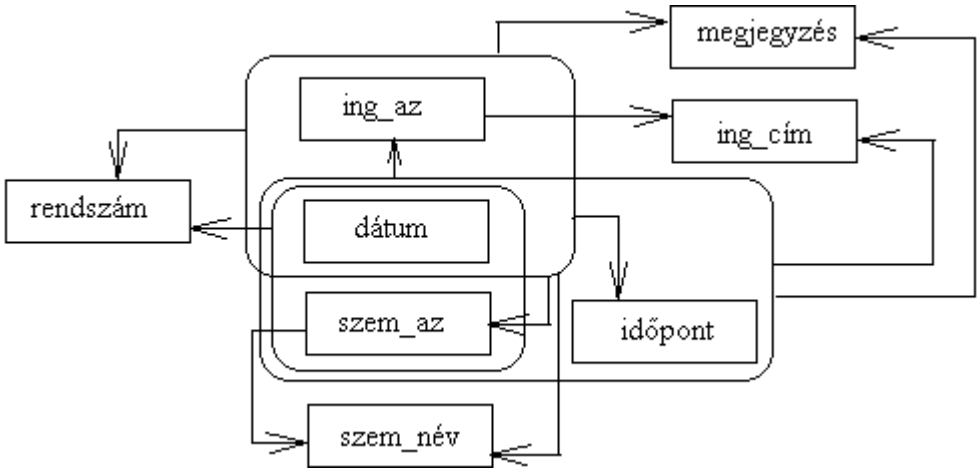
ing_az	ing_cím	dátum	időpont	megjegyzés	szem_az	szem_név	rendszám
PG4	Valahol	98jun2	98jun28	vélemény1	S56	Jean	HVG113
PG4	Valahol	98máj14	98jun5	vélemény2	S42	Joe	XYZ999

3) 1NF→2NF átalakítás

Függőségek meghatározása:

1. ing_az, dátum→időpont, megjegyzés, szem_az, név, rendszám
2. ing_az→cím
3. szem_az→név
4. szem_az, dátum→rendszám
5. szem_az, dátum, időpont→ing_az, cím, megjegyzés

Függőségi diagram:



Kandidát kulcsok (egyértelműen azonosítják a sorokat): (ing_az, dátum), (szem_az, dátum, időpont).

A két összetett kandidát kulcs közül a kevesebb attribútumot tartalmazót érdemes elsődleges kulcsnak megválasztani. Legyen az (ing_az, dátum) az elsődleges kulcs!

Az R relációt két részre bontjuk úgy, hogy az így kapott R1 és R2 relációk nem kulcs attribútumai az elsődleges kulcstól függjenek:

R1 reláció	
ing_az	ing_cím
PG4	Valahol

R2 reláció						
ing_az	*dátum*	időpont	megjegyzés	szem_az	szem_név	rendszám
PG4	98jun2	98jun28	vélemény1	S56	Jean	HVG113
PG4	98máj14	98jun5	vélemény2	S42	Joe	XYZ999

4) 2NF→3NF átalakítás. Szüntessük meg a tranzitív függéseket !

A szem_az→szem_név tranzitív függést tegyük új relációba:

R21 reláció	
szem_az	szem_név
S56	Jean
S42	Joe

R22 reláció					
ing_az	*dátum*	időpont	megjegyzés	szem_az	rendszám
PG4	98jun2	98jun28	vélemény1	S56	HVG113
PG4	98máj14	98jun5	vélemény2	S42	XYZ999

5) 3NF→BCNF

Ha megvizsgáljuk a fenti relációkat, láthatjuk, hogy R1 és R21 BCNF formában van, R22 viszont nem.

Mivel a szem_az,dátum→rendszám funkcionális függőségben a (szem_az,dátum) nem kandidát kulcs, ezért R22-t szét kell bontani R221 és R222-re az alábbi módon:

R221 reláció		
szem_az	*dátum*	rendszám
S56	98jun2	HVG113
S42	98máj14	XYZ999

R222 reláció				
ing_az	*dátum*	időpont	megjegyzés	szem_az
PG4	98jun2	98jun28	vélemény1	S56
PG4	98máj14	98jun5	vélemény2	S42

Megjegyzés:

A BCNF formába való átalakítás NEM MINDIG őrzi meg a függőségeket !

Projekt 5.

Feladat: az előző, áruházas projektekben használt relációk normalizálási lépéseinek bemutatása.

Az áruház egyes osztályain nyilvántartott adatok a következők:

- az alkalmazott azonosító száma
- az alkalmazott neve
- az alkalmazott címe
- az osztály azonosítója
- az osztály vezetője
- az osztály neve
- az osztályon értékesített áruk kódja
- az osztályon értékesített áruk neve
- az osztályon értékesített áruk ára
- a beszállító cégek kódja
- a beszállító cégek neve
- a beszállító cégek címe.

Ezen adatokat táblázatban ábrázolva UNF alakú relációhoz jutunk:

alk_az	alk_vez	alk_ker	alk_vár	alk_ut	alk_hsz	oszt_az	vez_az	o_név	áru_kód	áru_név
A19 A22	Piros Kis	Piroska Virág	Tab Lenti	Kis Fő	1 2	o2	A22	Virág	VA315	Amarilisz
A20	Fekete	Anna	Öskü	Nagy	3	o3	A21	Ital	IB238	Barackpálinka

A21	Zöld	Xénia	Márkó	Széles	2				IW114	Whisky
...	...	...	...	...	...	...	...	...	...	...

1NF

Ezt a táblázatot azonban nehéz karbantartani az ismétlődések (tárolási redundancia) és az ebből származó anomáliák miatt. 1NF alakra hozva azonban már nem tartalmazhat többértékű mezőket. Mivel a relációban azonos sorok nem fordulhatnak elő, a sorismétlés összetett kulcsot fog eredményezni (vastagított betűkkel jelöltem):

alk_az	alk_vez	alk_ker	alk_vár	alk_ut	alk_hsz	**oszt_az**	vez_az	o_név	**áru_kód**	áru_név	ár
A19	Piros	Piroska	Tab	Kis	1	o2	A22	Virág	VA315	Amarilisz	725
A22	Kis	Virág	Lenti	Fő	2	o2	A22	Virág	VA315	Amarilisz	725
A20	Fekete	Anna	Öskü	Nagy	3	o3	A21	Ital	IB238	Barackpálinka	672
A21	Zöld	Xénia	Márkó	Széles	2	o3	A21	Ital	IW114	Whisky	135
...	...	...	...	...	...	...	...	...	...	...	...

A négy érték (alk_az, oszt_az, áru_kód, gy_kód) együttesen már egyértelműen azonosítja a táblázat sorait.

2NF

A 2NF alak meghatározásánál olyan tábláztokat kell létrehoznunk, melyekben az összes nem kulcs attribútum teljesen függ az elsődleges kulcstól:

ALKALMAZOTT					
alk_az	alk_vez	alk_ker	alk_vár	alk_ut	alk_hsz
A19	Piros	Piroska	Tab	Fő	1

ÁRH_OSZT		
oszt_az	vez_az	o_nev
o1	A21	Élelmiszer

A20	Fekete	Anna	Öskü	Nagy	3
A21	Zöld	Xénia	Márkó	Széles	2
A22	Kis	Virág	Lenti	Kis	2

o2	A22	Virág
o3	A21	Ital

GYÁRTÓ				
gy_kód	gy_név	gy_vár	gy_ut	gy_hsz
G217	Fincsi Sütöde	Tab	Nagy	21
G332	Veszprémtej Rt.	Veszprém	Külső	7
G444	Bajor Pékség	Budapest	Petőfi	9
G523	Zwack	Budapest	Kossuth	41
G422	Kerekes Éva	Tab	Görbe	2
G231	Virágh József	Tab	Fő	9

ÁRU		
áru_kód	áru_név	ár
TT107	sajtos kifli	14
TT221	zsömle	8
TK329	rozskenyér	96
TF241	tejföl	49
IB187	Bólé	312
IB238	Barackpálinka	672
IW114	Whisky	356
VI105	Ibolya	218
VA315	Amarilisz	725
VP222	Páfrány	427

KAPCSOLATOK			
alk_az	**oszt_az**	**áru_kód**	**gy_kód**
A19	o2	VA315	G422
A19	o2	VA315	G231

A22	o2	VA315	G422
A22	o2	VA315	G231
...	...	...	...

3NF

A fenti relációkat megvizsgálga kapjuk, hogy azok nemcsak 2NF-ben, hanem 3NF-ben is vannak, hiszen nincsenek közvetett függések az egyes relációkon belül.
A *KAPCSOLATOK* relációban azonban megfigyelhetjük, hogy sok az ismétlődés! A BCNF (4NF-nek is nevezik) formára hozással azonban a felesleges tárolási igényt csökenthetjük.

BCNF

A *KAPCSOLATOK* relációt megvizsgálva láthatjuk, hogy többérétkű függéseket tartalmaz:

- egy osztályhoz több alkalmazott is tartozhat,
- egy osztály több árut is értékesíthet,
- egy osztály több cégtől is vásárolhat terméket,
- egy gyártó (beszállító) többféle árut is szállíthat,
- egy gyártó (beszállító) több osztály részére is szállíthat árut.

A *KAPCSOLATOK* relációt tehát tovább bontjuk:

DOLGOZIK	
alk_az	oszt_az
A19	o2
A20	o3
A21	o1
A21	o3
A22	o2

VÁSÁRLÁS	
oszt_az	áru_kód
o1	TT221
o2	VA315
o1	TK329
o3	IB238
o3	IW114

GYÁRTÁS	
gy_kód	áru_kód
G217	TT107
G217	TT221
G332	TF241
G444	TK329
G523	IB187
G523	IB238
G523	IW114
G422	VI105
G422	VA315
G422	VP222

A példából láthatjuk, hogy a normalizáció eredményeként egymással kapcsolatban álló, az eredetinél kisebb tárolási igényű relációkat kaptunk; megszűntek a törlési, módosítási és beszúrási anomáliák, és logikailag áttekinthetőbb lett az adatbázis.

LEFEDÉSEK (COVERS)

Definíció. Legyen R reláció, F és G pedig függőségi halmazok.
Azt mondjuk, hogy R ekvivalens G-vel, ha $F^+ = G^+$, és F≈G = FcoversG =GcoversF -vel jelöljük. (A lefedés és az ekvivalencia azonos fogalmak.)

Adott $d \in F$. Ellenőrizzük, hogy $d \in G^+$, ha $d = X \rightarrow Y$!
1. Határozzuk meg X^+ -t !
2. Ellenőrizzük, hogy $Y \subseteq X^+$!

Tulajdonságok
1) Ha $\forall d \in F \Rightarrow d \in G^+$, akkor $\forall\ V \rightarrow W \in F^+ \Rightarrow V \rightarrow W \in G^+$.
Bizonyítás.
$d = Y \rightarrow Z \in F \Rightarrow Y \rightarrow Z \in G^+ = \{ t \mid G \sim> t \} \Rightarrow F \subseteq G^+$.
Mivel $q \in F^+ \Leftrightarrow F \sim> q$, így $q \in G^+$.

2) Minden F függőségi halmaznak van olyan G lefedése, melyben a funkcionális függőség jobb oldalán csak egy attribútum szerepel.
Bizonyítás:
a) $F^+ \subseteq G^+$: $X \rightarrow Y \in F$, $Y = A1, \ldots An \Rightarrow X \rightarrow A1 \cup \ldots \cup An \in F$ ..
b) $G^+ \subseteq F^+$: (dekompozícióval)

Példa: $D \rightarrow EG \Rightarrow$(dekompozíció) $\Rightarrow D \rightarrow E,\ D \rightarrow G$

MINIMÁLIS LEFEDÉS

A minimális lefedés a függőségek minimális halmazát jelenti, ahol egyetlen függőség sem redundáns, valamint egyetlen funkcionális függőségi szabály bal oldala sem redundáns.

Definíció. Az F lefedés (függőségi halmaz) minimális, ha

1. a funkcionális függőségi szabályok jobb oldalán csak egy attribútum szerepel,

2. $\neg\exists d \in F : F\text{-}d \approx F$,

3. $\neg\exists (X \rightarrow A) \in F \wedge \neg\exists Z \subseteq X: (F\text{-}d) \cup (Z \rightarrow A) \approx F$.

Tétel. Minden F lefedésnek van egy vele ekvivalens minimális lefedése.

Bizonyítás.

1) $\exists$ F' : F'≈F", ahol F' olyan funkcionális függések halmaza, melyeknek a jobb oldalán csak egy attribútum szerepel.

2) Tekintsük $d \in F'$ -t. Ha (F'-d) ≈F, akkor töröljük d-t.

3) Legyen F" olyan funkcionális függőségi halmaz, melyre teljesül: $F'' \subseteq F'$, és F"≈F. Vizsgáljunk meg minden $d \in F''$-t, és töröljük a szabály bal oldaláról az attribútumokat mindaddig, amíg az ekvivalenciát nem veszítjük el.

Megjegyzés. A minimális lefedés nem egyértelmű!

Példa:

a) Tekintsük az alábbi függőségi halmazt: A→B, B→A, B→C, A→C, C→A !

Ekkor egy minimális lefedése a halmaznak: B→A, B→C, C→A, A→C.

Másik lehetséges minimális lefedés: A→B, B→A, A→C, C→A.

b) Tekintsük az alábbi funkcionális függéseket: AB→C, A→B, B→A. Az AB→C függőség bal oldaláról az A-t eltávolítva kapjuk a fenti halmaz egy minimális lefedését: B→C, A→B, B→A.

Ha a szabály bal oldaláról a B-t töröljük, akkor pedig az A→C, A→B, B→A minimális lefedést kapjuk. Tehát ebben a példában 2 minimális lefedése van az adott halmaznak.

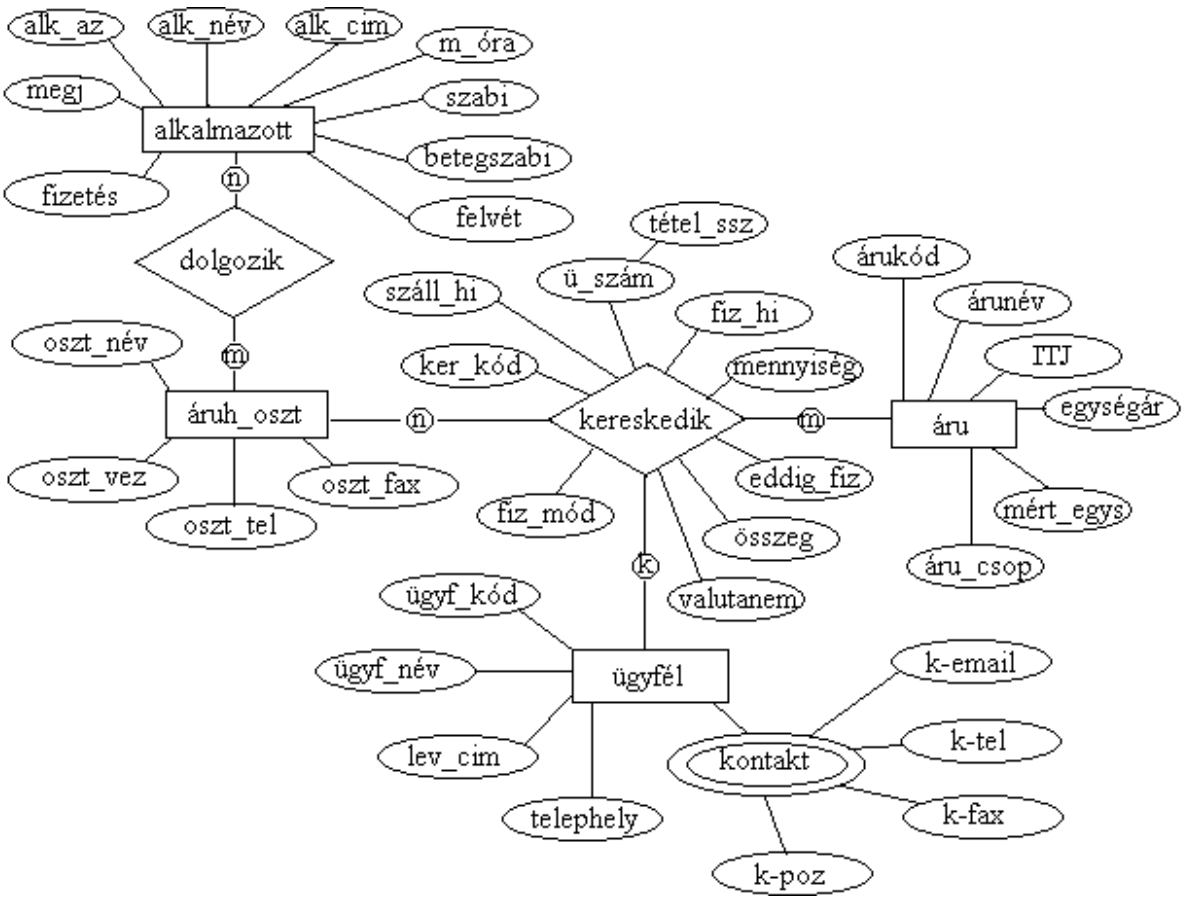
A korábban már tárgyalt áruházas példát bővítsük ki az alábbi módon:

Az áruházról a következő adatokat tartsuk nyilván:

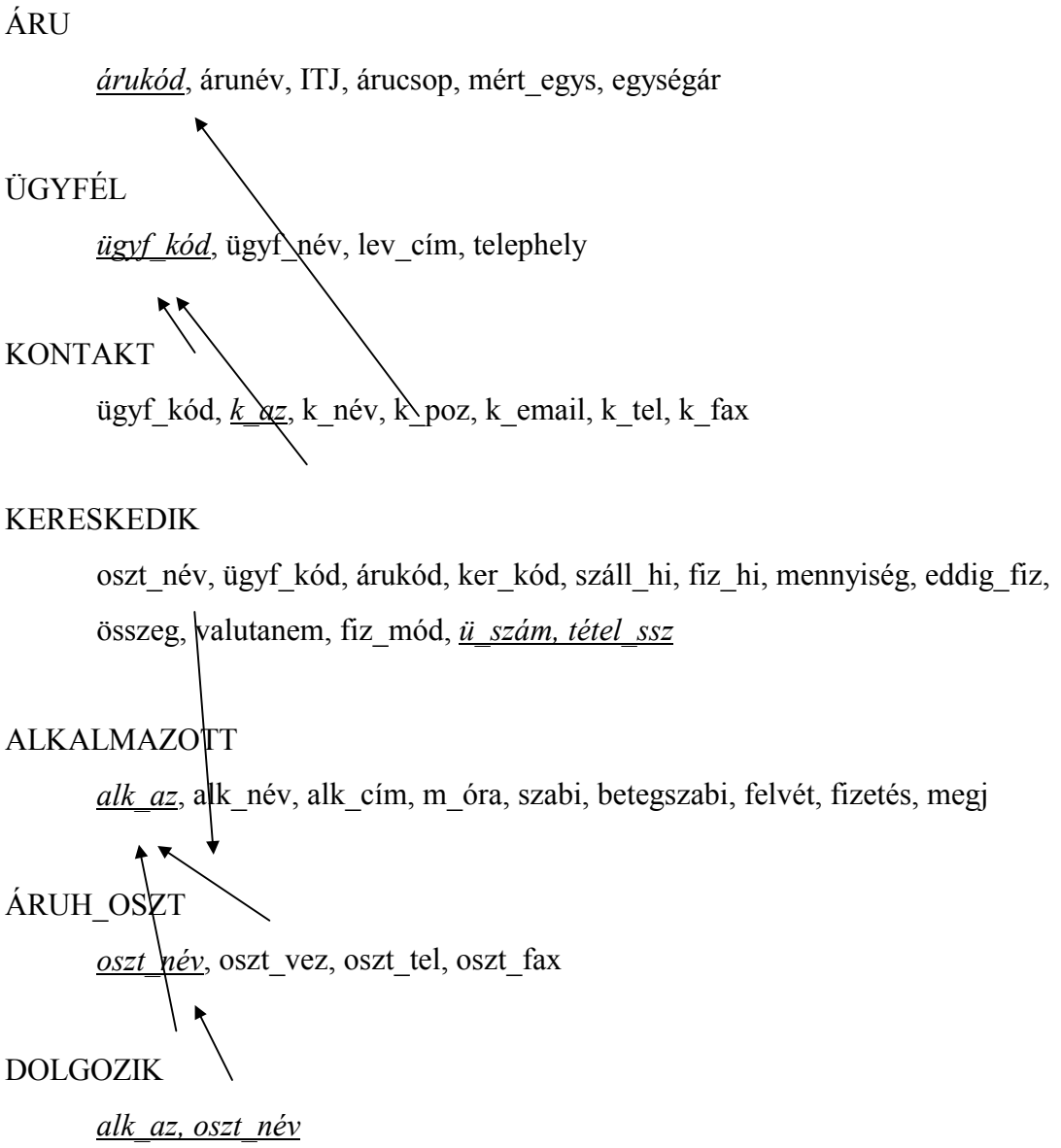
- Minden alkalmazott azonosító száma, neve, felvételének időpontja, mely áruházosztályon dolgozik és milyen pozícióban, lakcíme, fizetése, a havonként ledolgozott munkaórája, kivett szabadsága, kivett betegszabadsága, és egyéb megjegyzéseket is nyilvántarthassunk.
- Minden áruházosztály neve, vezetője, telefonszáma, faxa, alkalmazottai és azok beosztása.
- Minden áru kódja, neve, ITJ száma, mely árucsoportba tartozik, mértékegysége, egységára, mely cégtól vásárolta az áruház, milyen mennyiségben, milyen szállítási határidővel szállítja, milyen fizetési módot használ, fizetési határidő, egyéb megjegyzések.
- Minden ügyfél kódja, neve, levélcíme, telephelye, a kontaktszemély neve a vállalatnál, annak telefonja, faxa és e-mail címe, milyen ügyletet bonyolított le az áruházzal

(vásárolt vagy eladott), milyen tételben szállított/vásárolt, milyen fizetési és szállítási határidővel, milyen fiztési módban, valutanemben vásárolt /adott el, és mennyiért.

Ezek alapján megadhatjuk az áruház entitás-reláció (ER) modelljét:



Az ER modell relációs modellbe való átalakítás szabályait alkalmazva kapjuk az alábbi relációkat (a nyilakkal az idegen kulcsokat jelöltem) :



A fenti relációkat vizsgáljuk meg normalizálási szempontból !

Ha adatokkal töltjük fel a táblázatokat, akkor láthatjuk, hogy a mezők egyértékűek, tehát azok 1NF alakban vannak.

A relációk nem kulcs attribútumai funkcionálisan teljesen függenek az elsődleges kulcstól, tehát 2NF alakban vannak a relációk.

Mivel funkcionális függés csak az elsődleges kulcsból indul ki, vagyis nincsenek tranzitív függések, ezért a relációk 3NF alakban vannak.