Dr. Kotsis Domokos

Adatbázisok

Segédanyag az előadáshoz

Cél

Cél: információ tárolás, feldolgozás, továbbítás.

Az információ feldolgozás alapvető módszerei

- **Szóbeli**
- **Írásbeli**
  Nyomtatott
- **Elektronikus**

A szóbeli információ feldolgozás

Az információ tárgya akkor, ott nem kell, hogy jelen legyen.

Szóbeli leírás: magas absztrakciós szint.
Interaktív.

Mit kell tudni?

Forrás: **értelmes beszéd.**
Nyelő: **a beszéd értése.**

Kicsi különbség.

Az írásbeli információ feldolgozás

Az információ szolgáltatás tárgyán kívül annak forrása (az információ „adója”) sem kell, hogy akkor, ott jelen legyen.

Nincs metakommunkáció, nincs interakció : még magasabb absztrakciós szint.
Hang nincs, de képek, utalások lehetnek.
Nyomtatás: sokakhoz eljut.

Mit kell tudni?

Forrás: **írás tudás, eszközök használata (stilus, lúdtoll, töltőtoll, írógép).**

Nyelő: **olvasás tudás.**

Nagyobb különbség.

Az elektronikus információ feldolgozás

Az eddigieken kívül:
A korábbi módszerek határai kitágulnak (telefon, Email).
Az információ szolgáltatásnak nem egy forrása van (rádió televízió, internet).

Változatos hangok, képek (mozgók is), interaktív.
Alacsonyabb absztrakciós szint.
Mindenkihez eljuthat.

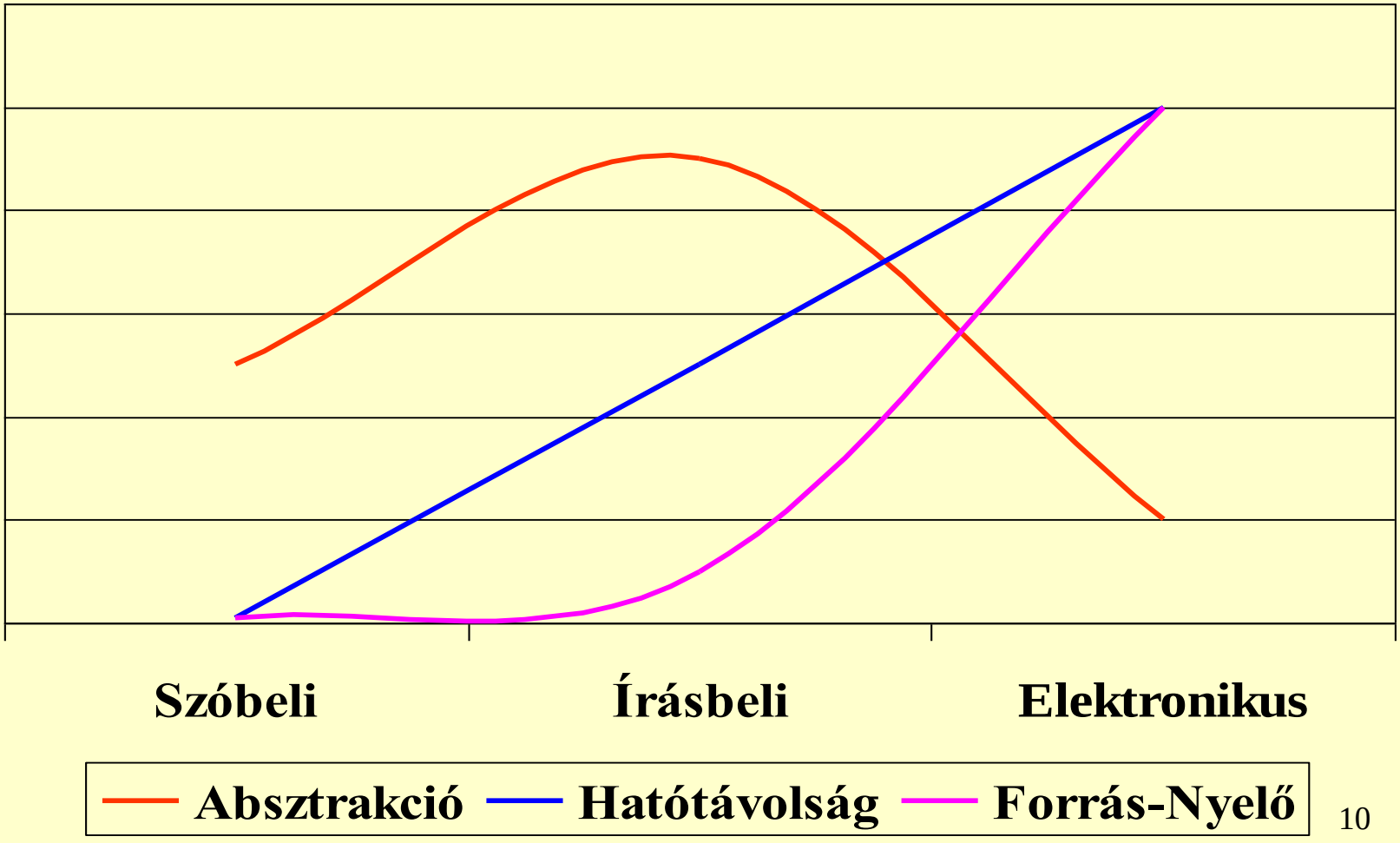
Mit kell tudni?

Forrás: **tv, rádió műsorok készítése;**
távközlési hálózat tervezése, üzemeltetése;
számítógépek hálózatok programozása.

Nyelő: **tv, rádió, telefon kezelése; számítógépek,**
programok használata (ECDL).

Óriási különbség

Összefoglalás



Szakmai bevezetés

Az információfeldolgozás kezdetei: a Hollerith gépek
A számítógépek kezdeti felhasználási területei. Számítástechnika-Informatika.
A mai helyzet: információ = energia?

Az információ feldolgozás alapvető módszerei

- **Folyamat szemléletű információ feld.**
- **Adatbázis szemléletű információ feld.**
- **Döntés-támogató rendszerek.**
- **Tudásbázisok (szakértői rendszerek).**

A folyamatszemléletű információ feldolgozás

cél → folyamat → állományok

Előnyök:

- **csak a szükséges adatokkal dolgozik**
- **optimális adatstruktúra használható**

Hátrányok:

- **többszörös tárolás (inkonzisztencia)**
- **csak egy cél**

Optimális struktúra

Opt. szemp.

- **tárolóhely**
- **feldolgozási idő**
  - **létrehozás**
  - **keresés**
  - **változtatás**
- **felhasználás egyszerűsége**

Keresés 1.

Egy N elemű file-ban egy rekord keresésének lépései:

$$S_N\ (A+B+C)$$

A keresési idő:

$$T_N = \sum_{i=1}^{N} p_i \sum_{j=1}^{S_i} (A_{i,j}+B_{i,j}+C_{i,j})$$

ahol p_i az i-edik elem keresésének valószínűsége ($\sum_{i=1}^{N} p_i = 1$),

és S_i az i-edik elem megtalálásához szükséges lépésszám.

Keresés 2.

Fontos a hardware, a tároló szerkezete!

RAM: a címek sorrendje közömbös.

Mozgófejes lemez: pozícionálás, forgás, beolvasás.

Keresés 3.

átlagos t időt véve:

$$T_N = \sum_{i=1}^{N} p_i \, t$$

egyenlő gyakorisággal számolva:

$$T_N = S_N t$$

A keresést meghatározó tényezők

Struktúra

Algoritmus

Pl. rendezetlen esetben lineáris keresés:

$S_N \approx N/2 \quad S'_N \approx N$;

ugyanez rendezett esetben:

$S_N \approx N/2 \quad S'_N \approx N/2$;

de itt nyilván van jobb módszer is.

A legfontosabb állomány struktúrák

- **Szekvenciális állomány struktúrák**
- **Hierarchikus állomány struktúrák;**
- **Hálós állomány struktúrák;**
- **Nem konzekutív állomány struktúrák**

Fizikai szekvenciális állomány struktúrák

A rekordok logikai és fizikai sorrendje megegyezik.

Keresési módok:

- **Lineáris keresés**
- **Bináris, logaritmikus keresés**
- **Peterson-féle keresés**

Bináris, logaritmikus keresés

Minden lépésben hosszra nézve felezzük a vizsgálandó állományt.

$$S_N \approx S'_N \approx \log_2(N)$$

Heurisztikus bizonyítás:

legyen K olyan, hogy $N=2^K$.

Minden felezésnél $K_{új} = K_{régi} - 1$.

$K_{új}=0$ éppen innen K lépés után lesz, és

$$K=\log_2(N)$$

Peterson-féle keresés

Minden lépésben az előfordulás valószínűségére nézve felezzük a vizsgálandó állományt.

Egyenletes eloszlás mellett:

$$A = A_E + \frac{A_U - A_E}{K_U - K_E} * K$$

ekkor:

$$S_N \approx S'_N \approx \tfrac{1}{2} \log_2(N)$$

Bináris vs. Peterson keresés

S_N a Petersonnál kisebb, de

1. az eloszlás nem feltétlenül egyenletes,

2. Petersonnál valódi osztás kell, a binárisnál a léptetés alkalmazható.

A leggyorsabb, ha $N = 2^k - 1$ alakú.

Ha nem ilyen:

1. két részre osztás

2. kiegészítés álrekordokkal

A fizikai szekvencialitás előnye, hátránya

A keresés lépésszáma kicsi, de a rendezettség biztosítása is időigényes!

1. bonyolult beszúrási algoritmus

2. rendszeres fizikai rendezés

Logikai szekvenciális állomány struktúrák

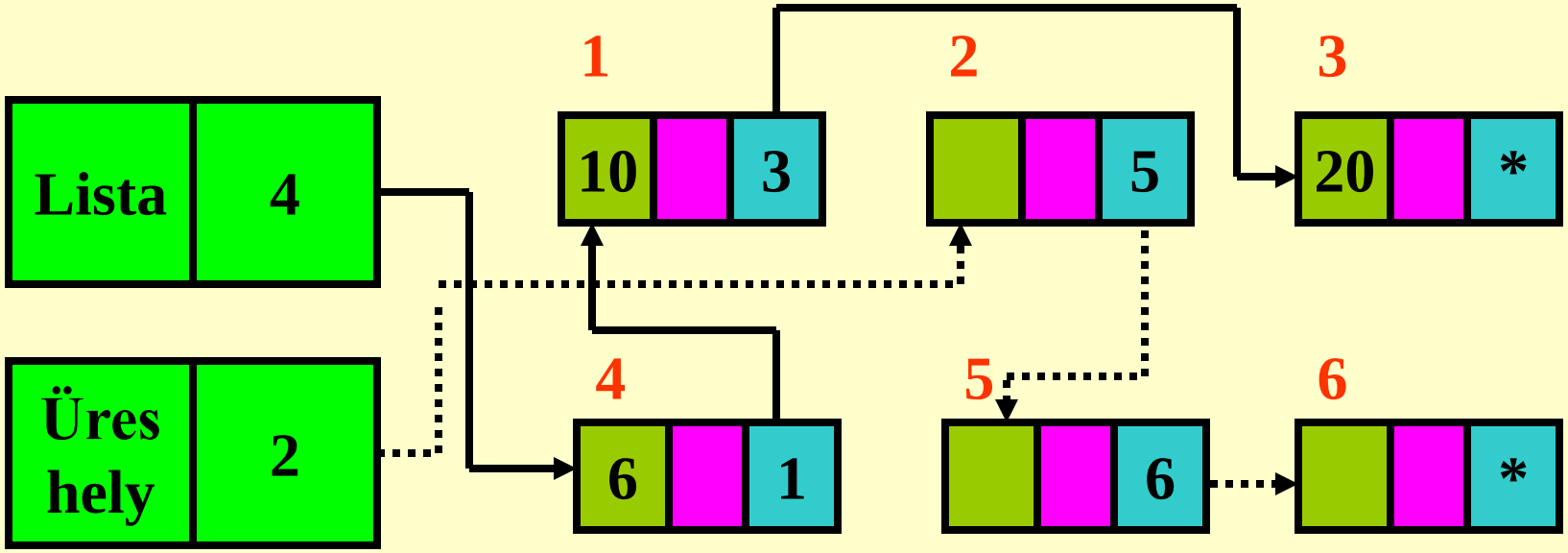
A rekordok logikai és fizikai sorrendje nem egyezik meg.

Mutatórendszerek (egy- kétirányú, gyűrűs).

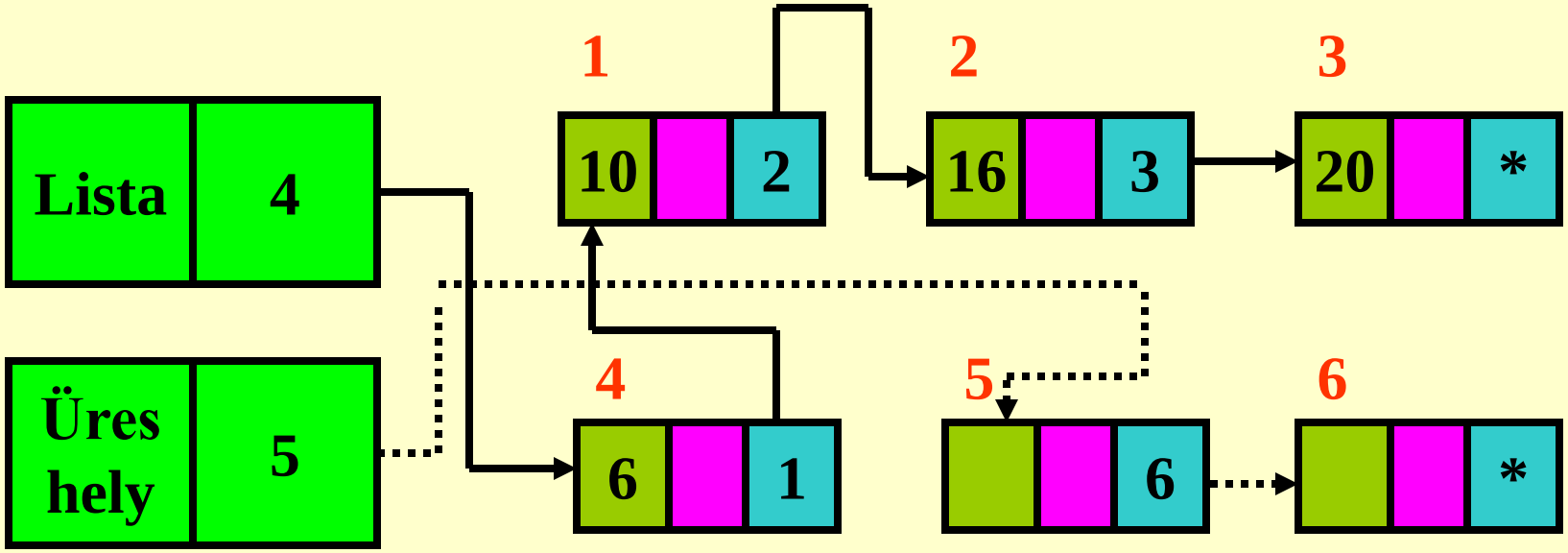
Indító tábla.

Dinamikus file-oknál jó: csak mutatókkal kell dolgozni.

Példa

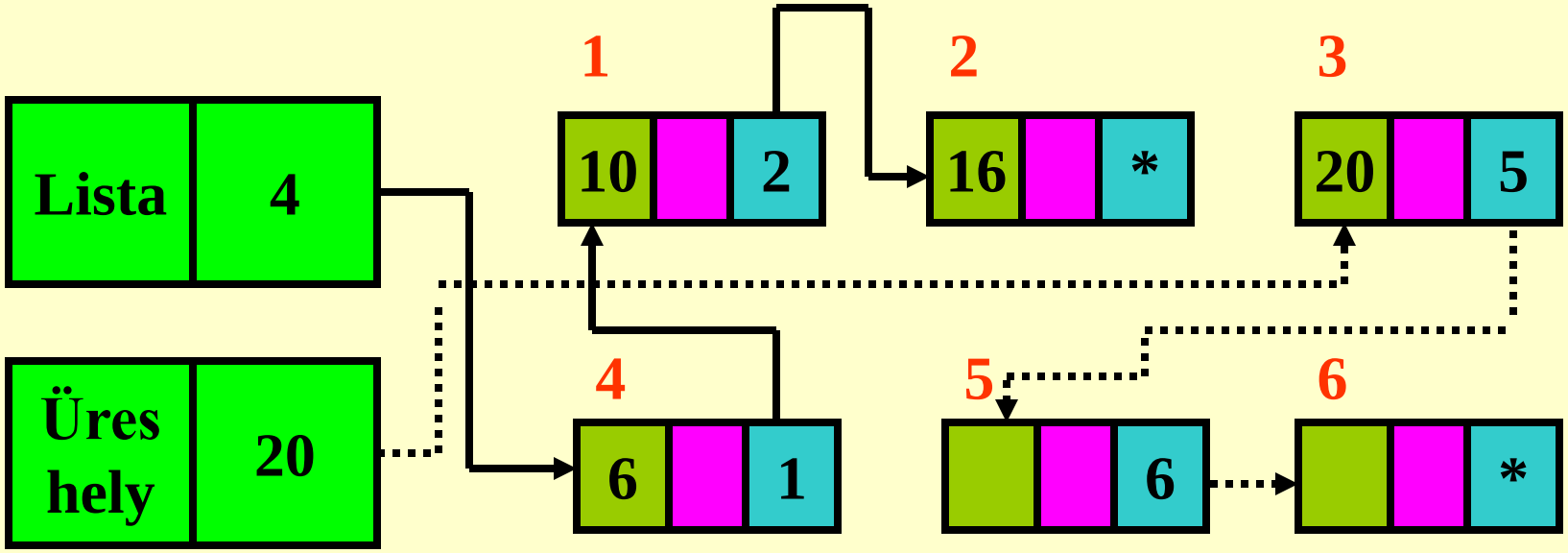


Beszúrás



Új elem: 16

Törlés



Törlendő: 20

Hátrányok

A fizikainál ismert keresési algoritmusok nem használhatók.

Mozgófejes lemeznél:

Pozícionálás, forgási idő kell az eléréshez.

Véletlenszerű elhelyezkedésnél C cilindernyi anyagnál átlagosan C/3 cilindernyi pozícionálás kell.

Csaknem fizikai szekvenciális file

Létrehozáskor fizikai szekvenciálisan elhelyezett logikai szekvenciális file.

Feldolgozáskor logikai szekvenciális.

Túlcsordulási terület a cilinderen.

Kupacos keresés 1.

N rekord, G rekord/ csoport, N/G csoport.

$$\sum_{k=1}^{\frac{N}{G}} k\left(\frac{1}{\frac{N}{G}}\right) = \frac{\frac{N}{G}+1}{2}$$

Kupacos keresés 2.

$$\overline{S}_N \approx \frac{\frac{N}{G}+1}{2} + \frac{G-1}{2} = \frac{1}{2}\frac{N}{G} + \frac{G}{2}$$

Kupacos keresés 3.

A minimum kereséshez G szerint deriválva:

$$\frac{N}{G^2} = \frac{1}{2}$$

Ennek minimum helye: $G = \sqrt{N}$

Ebből következően: $\overline{S}_N \approx \sqrt{N}$

Gyakoriság szerint rendezett file

Statikus, dinamikus megoldás.

Önrendező algoritmusok.

Hierarchikus struktúrák 1.

Egy megelőző, több rákövetkező (fa).

Nyelvi leírás: COBOL, LISP

Hierarchikus struktúrák 2.

Ábrázolás:

A.) Belső mutatós módszerek

1. Left-list

2. Több pointer

3. Segédrekordok

4. Gyűrűk

Hierarchikus struktúrák 3.

B.) Külső mutatós módszerek

1. Táblázatok

2. Bináris mátrixok

Hálós struktúrák 1.

1. Visszavezetés hierarchikusra.
2. Az ott leírt módszerek?
    A.) Belső mutatós módszerek
        1. Left-list **nem**
        2. Több pointer **igen**
        3. Segédrekordok **nem**
        4. Gyűrűk **nem**

Hálós struktúrák 2.

B.) Külső mutatós módszerek

1. Táblázatok igen

2. Bináris mátrixok igen

Nem konzekutív struktúrák

Indexelt szervezés

Direkt szervezés

Indexelt szervezés

1. Sűrű indexelés

2. Ritka indexelés

Sűrű indexelés

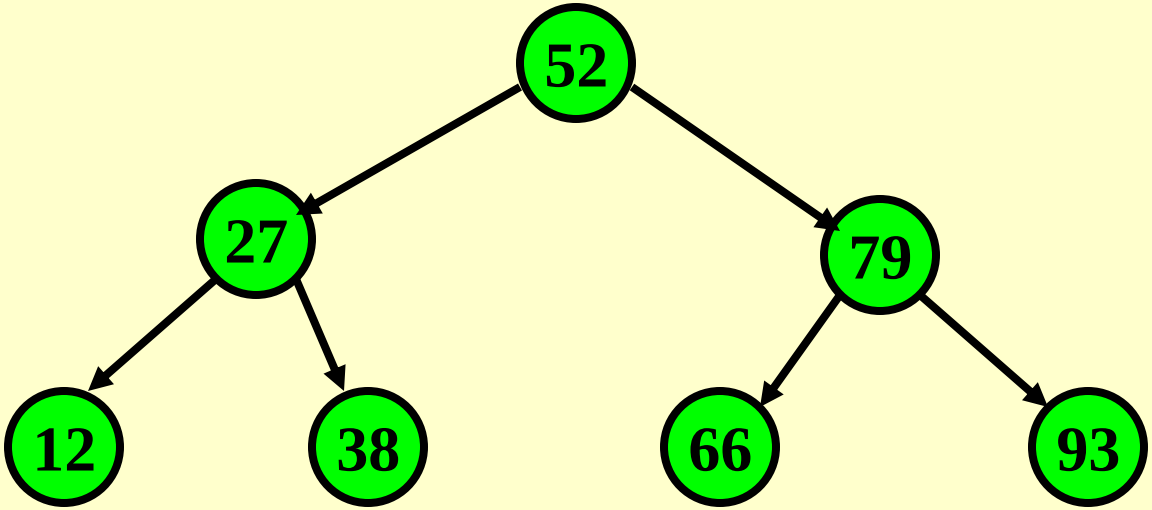
Minden rekordhoz tartozik index bejegyzés

Előnyök: több index szervezhető

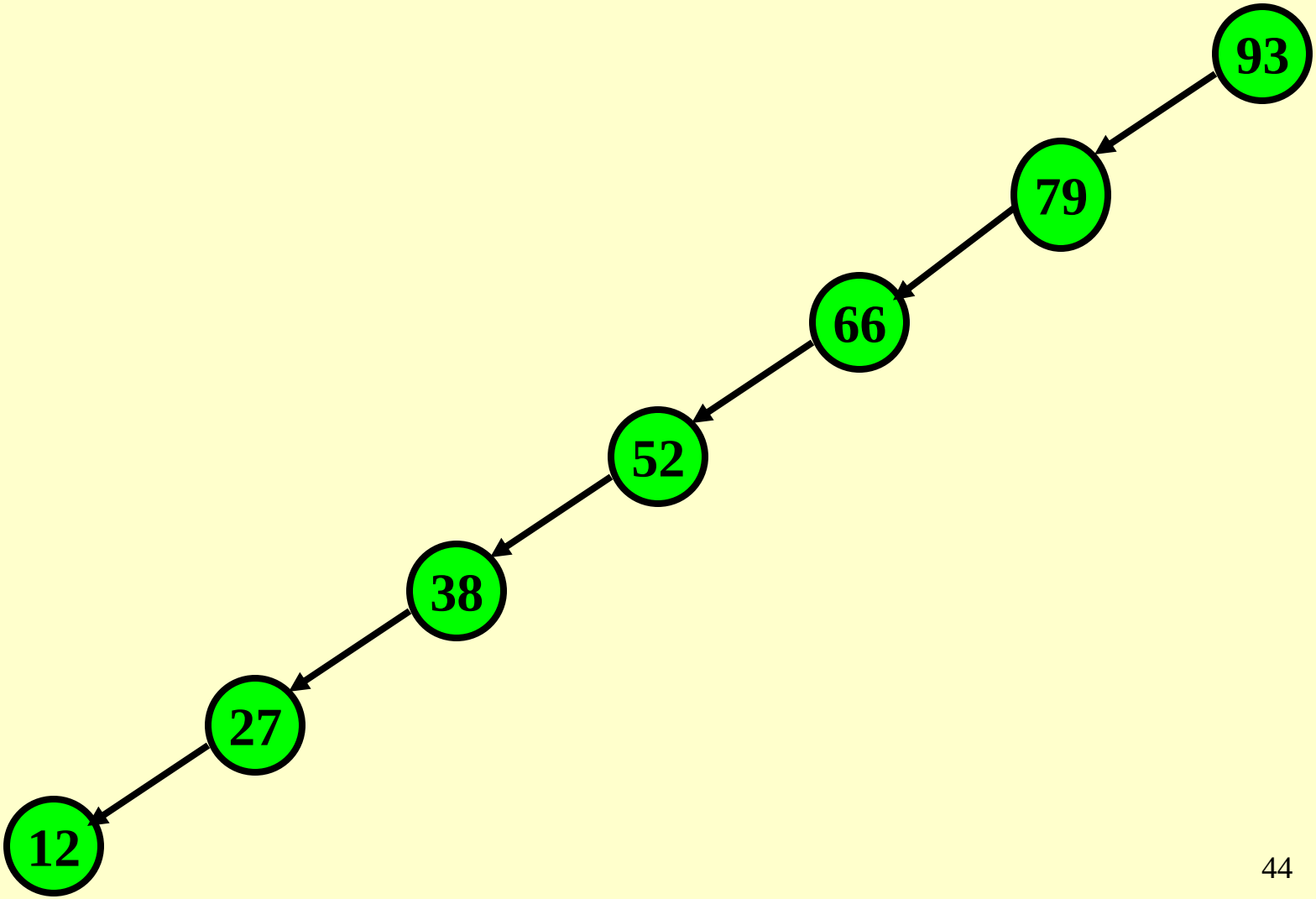
Hátrányok: nagyok az index file-ok

Mikor lehet gyors a keresés?

Bináris fa I.



Bináris fa II.



B fák

R. Bayer 1970

n a B fa rendje

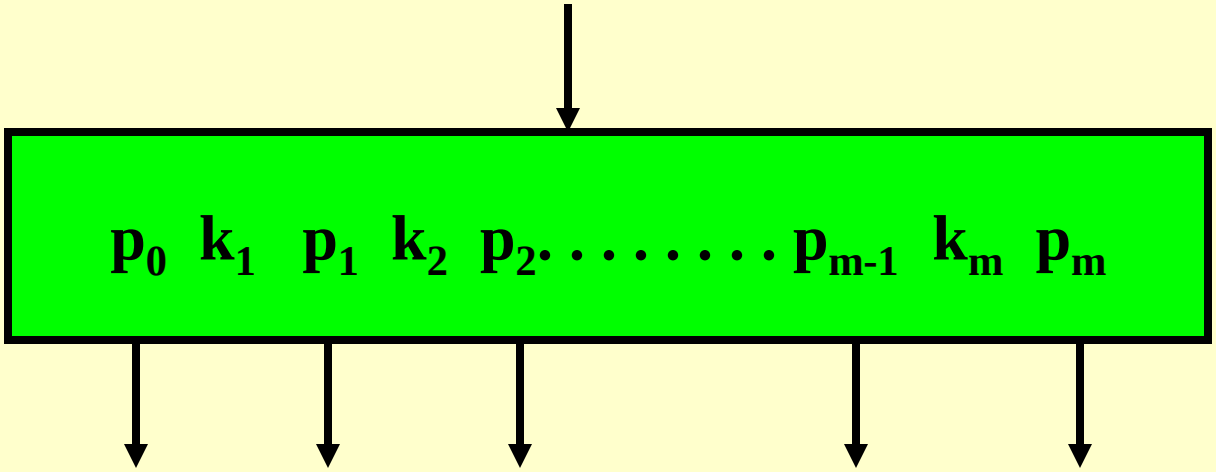
1. **Minden lap legfeljebb 2n tételt tartalmaz.**
2. **Minden lapon – a gyökér kivételével - legalább n tétel van .**
3. **Ha egy lapon m tétel van, vagy levéllap, vagy m+1 utódja van.**
4. **Minden levéllap ugyanazon a szinten van.**

B fák elérése

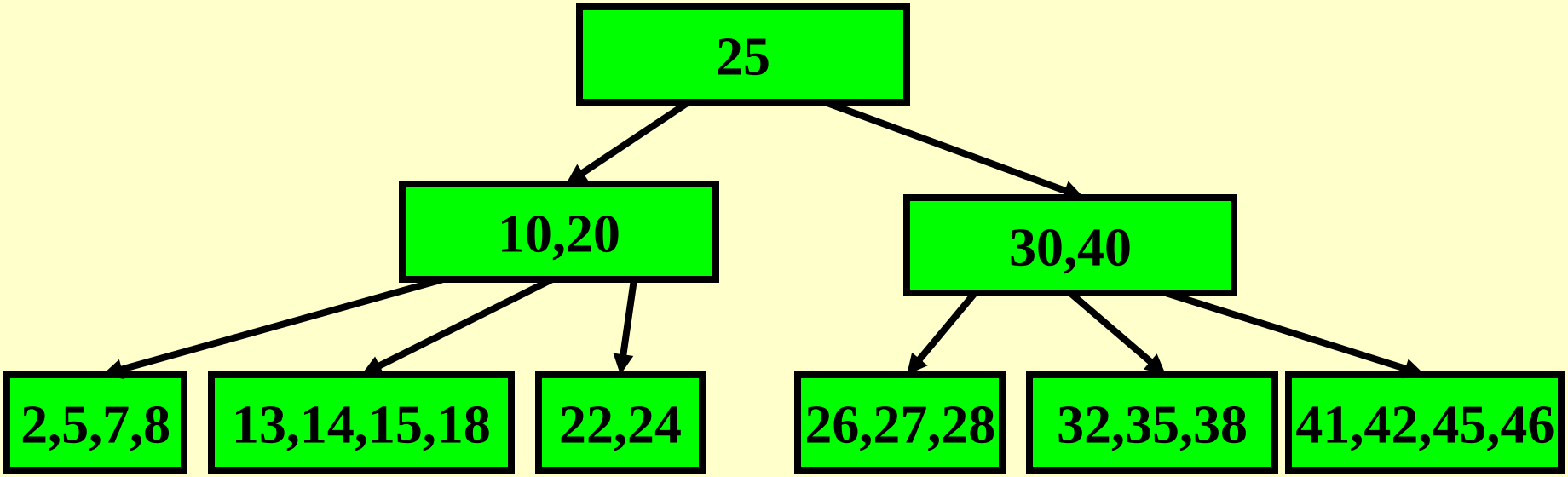
A kereséshez szükséges átlagos lépésszám a laphivatkozásoktól függ, ezek száma N elemnél n rendű fában:

$$\log_n N$$

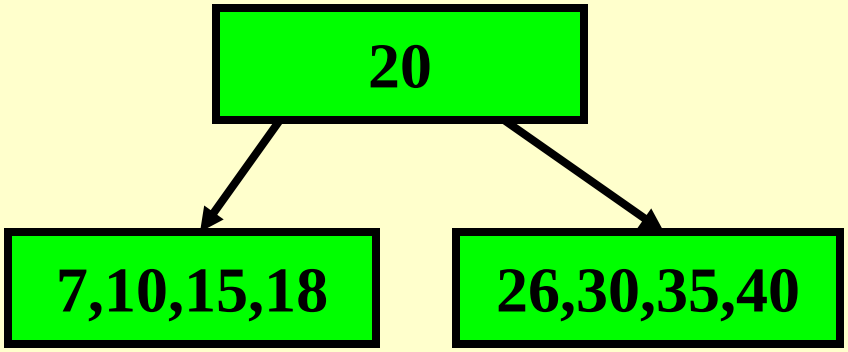
B fa egy lapja

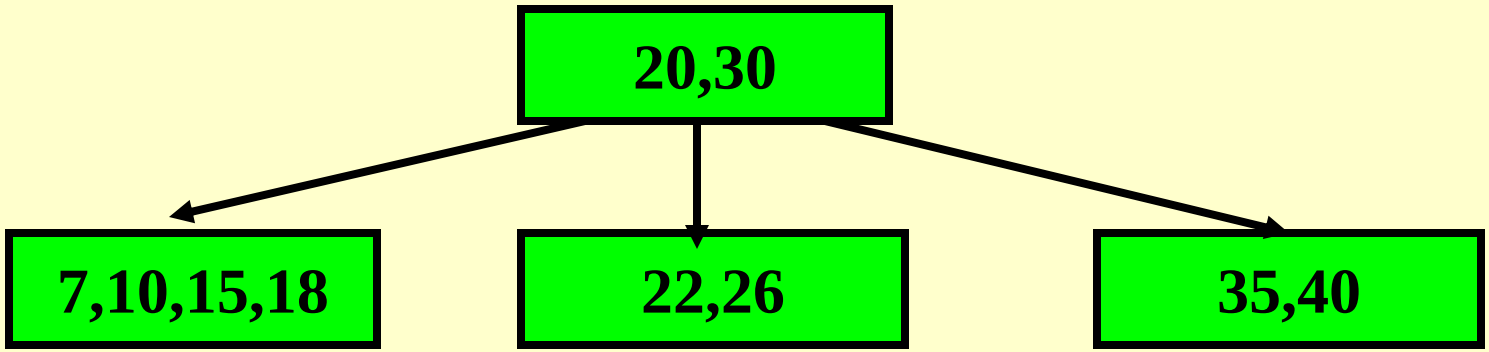


Másodrendű B fa

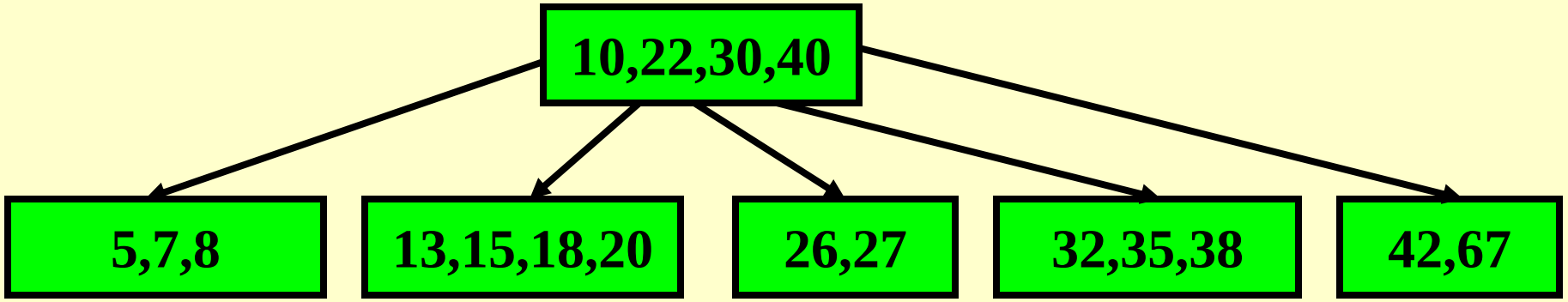


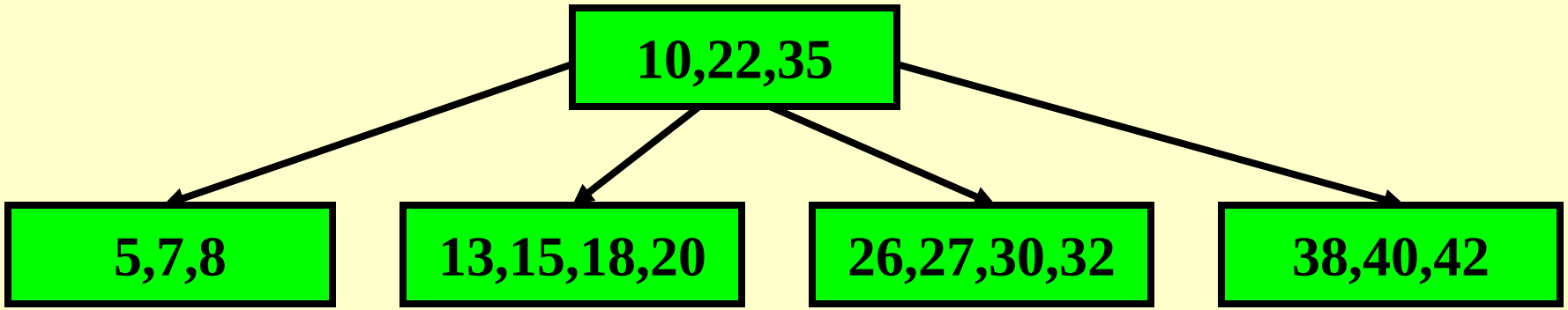
Egy új (22) kulcs beszúrása





Egy kulcs (67) törlése





B+ fák

A gyakorlatban index állományok készítésére az u.n. B+ fákat használják. (Ilyen struktúrát alkalmaz – sok más rendszerrel együtt – az Oracle is.)

A B+ fa egy olyan B fa, melyben azok a pointerek, melyek adatokra (adatok címére) mutatnak, s amelyeket keresünk, valamennyien a levéllapokon helyezkednek el, így a levéllapok és a többi lap struktúrája különbözik.

Ritka indexelés

„Jelző cölöpök” mutatják a rekord helyét.

Rendezettség kell!

Több szintű index rendszer is készíthető.

Gyors elérés, de csak egy szempont szerint.

Index-szekvenciális szervezés 1.

A ritka indexelés egy megvalósítása.

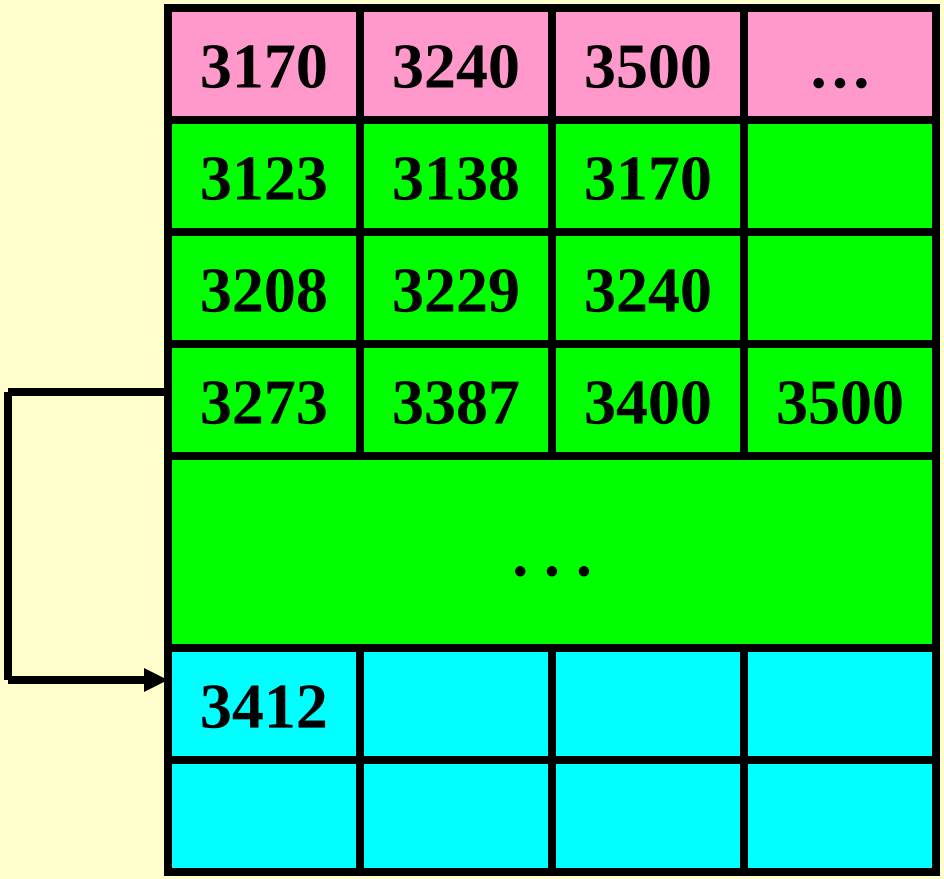
A fizikai architektúrának megfelelően:

Sáv; cilinder; fő index.

Index; elsődleges; túlcsordulási terület.

Szervezés: az elsődleges területen fizikai, a túlcsordulási területen logikai szekvenciális szervezés.

A tároló felosztása

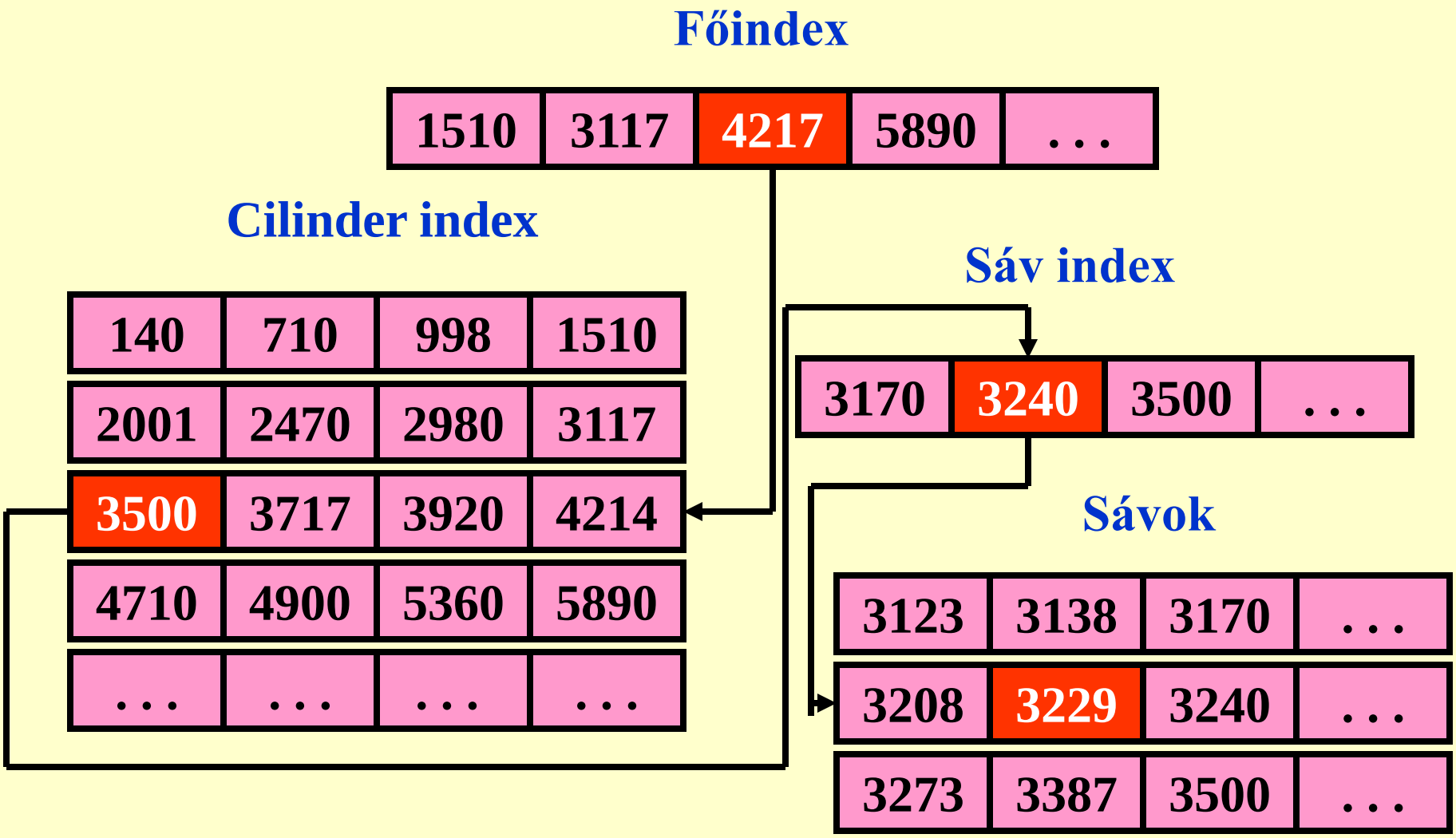


Sávindex

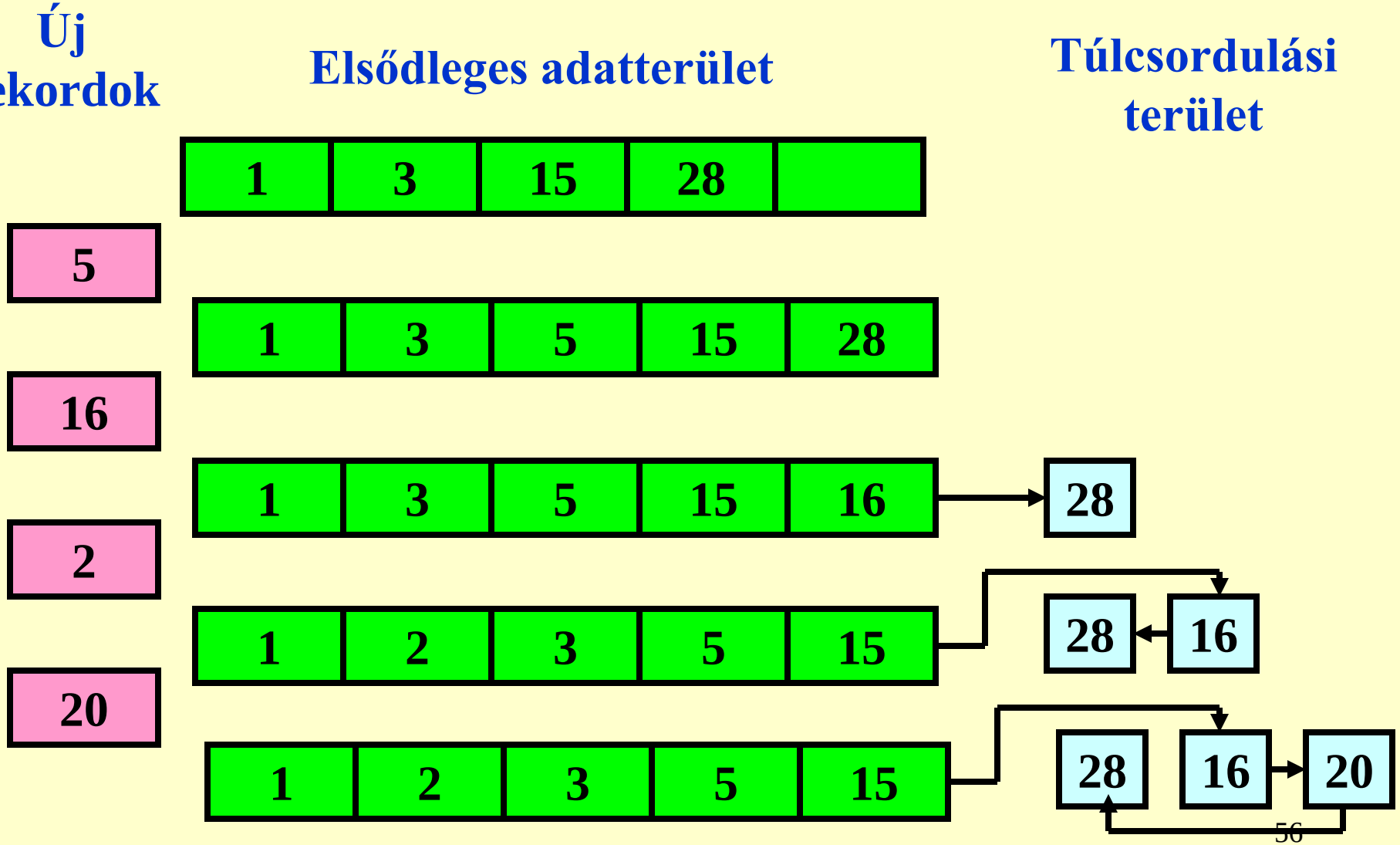
Elsődleges terület

Túlcsordulási terület

Egy rekord (3229) megkeresése



Rekordok beszúrása



Direkt szervezés 1.

Egy leképezést adunk meg a rekord kulcsa (K) és a címe (A) között.

$$A = f(K)$$

Kívánatos lenne az egy-egy értelműség:

$$K_1 \neq K_2 \Rightarrow f(K_1) \neq f(K_2)$$

Direkt szervezés 2.

Közvetlen leképezés: igen rossz a tárkihasználás.

Hashing algoritmusok: megengedjük, hogy

$$K_1 \neq K_2$$

esetén is – lehetőleg ritkán - előfordulhasson

$$f(K_1) = f(K_2)$$

Két hashing algoritmus

Maradék módszer: M modulus mellett

$$A = R\left(\frac{K}{M}\right)$$

Csonkítás: a kevés információt hordozó jegyek elhagyása.

Szinonímok

A hashing algoritmusok nem egy-egy értelműek, így szinonimok keletkezhetnek.

Szinonim kezelő módszerek:

Külső láncolás

Belső láncolás

Nyílt (Peterson) módszer

Többszörös hashing

Bucket-ek használata

A külső láncolás

A szinonimokat mutatók kötik össze.

Külön túlcsordulási terület.

Problémák:

- **Tárkihasználás**
- **Törlés: az elsőt nem lehet törölni.**

A belső láncolás

A szinonimokat itt is mutatók kötik össze, de azok az (egyetlen) elsődleges terület még nem használt helyeire kerülnek, az eredeti címhez a lehető legközelebb.

A tárkihasználás jó, de másodlagos szinonimok keletkezhetnek.

A nyílt módszer

A rekordok ugyanúgy helyezkednek el, mint a belső láncolásnál, de nincsenek mutatók.

A keresésnél szükséges lépésszám nagyobb, mint a belső láncolásnál.

A többszörös hashing

Ha a leképezés szinonimra vezet, egy másikkal próbálkozunk.

(A gyakorlatban két hashing algoritmust, majd a korábban ismertetett módszerek valamelyikét használják.)

Bucket-ek használata

E módszernél nem egy-egy rekordnak, hanem egy-egy rekord csoportnak van külön címe.

Kevesebb szinonimkezelésre van szükség, de a csoportokon belül is kell keresnünk.

Hasznos akkor lehet, ha hardware szempontok (szektor) indokolják.

Példa szinonímkezelésre

35, 46, 18, 14, 63, 06,75, 85,91, 52.

Leképezés: a 11-el osztás maradéka.

Második: a 7-el osztás maradéka.

Példa bucket használatra

21, 32, 71, 14, 83, 72, 43, 51, 02, 11, 93, 44.

Leképezés: 4 bucket-be a második jegy szerint.

Összehasonlítás 1.

Az egyes módszerekhez átlagosan szükséges lépésszámot a tényleges (N), és az egyáltalán elhelyezhető (M) rekordszámmal meghatározott A=N/M u.n. kitöltési tényező függvényében vizsgáljuk.

Összehasonlítás 2.

Belső láncolásnál sikeres esetben:

$$S_N = 1 + \frac{1}{8A}\left(e^{2A} - 1 - 2A\right) + \frac{1}{4}A$$

Sikertelen esetben:

$$S_N^{'} = 1 + \frac{1}{4}\left(e^{2A} - 1 - 2A\right)$$

Összehasonlítás 3.

Nyílt módszernél sikeres esetben:

$$S_N = \frac{1}{2}\left(1 + \frac{1}{1-A}\right)$$

Sikertelen esetben:

$$S_N' = \frac{1}{2}\left[1 + \left(\frac{1}{1-A}\right)^2\right]$$

Összehasonlítás 4.

Többszörös hashing-nél sikeres esetben:

$$S_N = -A^{-1} \ln(1 - A)$$

Sikertelen esetben:

$$S_N' = (1 - A)^{-1}$$

Összehasonlítás 5.

Példaképpen sikeres esetben:

$$1+\frac{1}{8A}\left(e^{2A}-1-2A\right)+\frac{1}{4}A=1+\frac{1}{2}A+\sum_{i=2}^{\infty}\frac{2^{i-2}A^{i}}{(i+1)!}$$

$$\frac{1}{2}\left(1+\frac{1}{1-A}\right)=1+\frac{1}{2}A+\frac{1}{2}A^{2}+\ldots=1+\frac{1}{2}\sum_{i=1}^{\infty}A^{i}$$

$$-A^{-1}\ln(1-A)=1+\frac{A}{2}+\frac{A^{2}}{3}+\ldots=\sum_{i=0}^{\infty}\frac{A^{i}}{i+1}$$

Összehasonlítás 6.

Sikertelen esetben:

$$1+\frac{1}{4}\left(e^{2A}-1-2A\right)=1+\frac{1}{4}\sum_{i=2}^{\infty}\frac{(2A)^i}{i!}$$

$$\frac{1}{2}\left[1+\left(\frac{1}{1-A}\right)^2\right]=1+A+\frac{3}{2}A^2+\ldots=1+\sum_{i=1}^{\infty}\frac{i+1}{2}A^i$$

$$(1-A)^{-1}=1+A+A^2+\ldots=\sum_{i=0}^{\infty}A^i$$

Összehasonlítás 7.

Nem elég a lépésszámot vizsgálni, hisz az egyes lépések időigénye módszerenként más.

Figyelembe kell venni a hardware adottságokat is.

Összehasonlítás 8.

Gyors véletlen elérésű (pl. RAM) tárolónál:

TH: nagy algoritmusidő.

NyM: több lépés mint BL-nél.

BL: másodlagos szinonimok miatt több lépés, mint KL-nél.

KL: leggyorsabb, de rossz tárkihasználás.

Összehasonlítás 9.

Mozgófejes tárolónál:

TH, KL: sok fejmozgás.

NyM: több lépés mint BL-nél.

De NyM-hez nem kell CPU, így a leggyorsabb lehet.

Alapelv

A rendelkezésre álló adatokból indulunk ki, az „összes” adatot (entity) és a közöttük fennálló „összes” kapcsolatot (relationship) egy integrált adatbázisba gyűjtjük, és valamennyi felhasználó valamennyi kérdéséhez ezt az adatbázist (vagy ennek egy részét) használhatja.

Adatreprezentáció

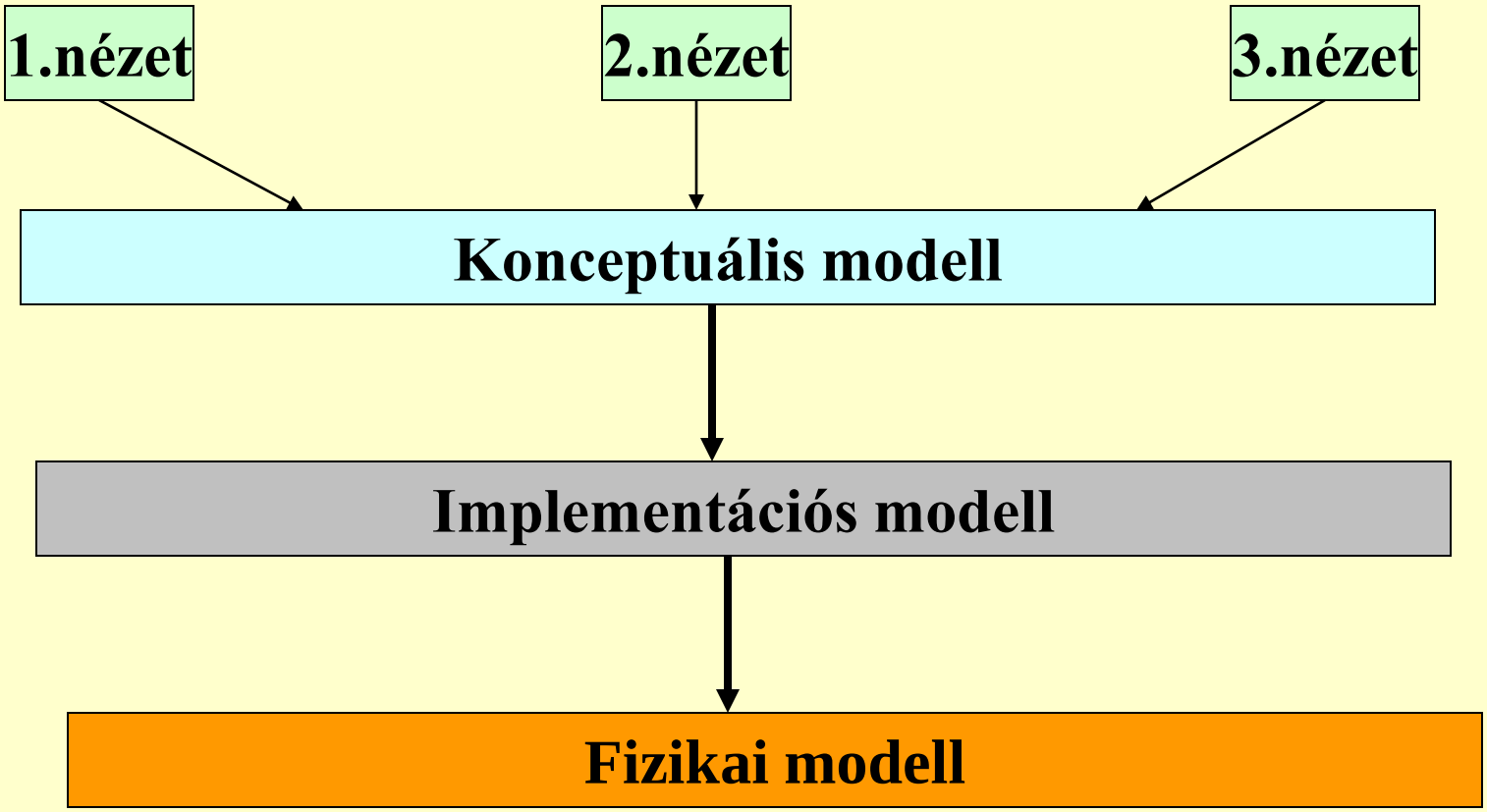
Az „összes adat” egy „mini világra” vonatkozik.

Ennek leírása több szintű:

- ***konceptuális*** **(magas szintű),**
- ***implementációs*** **(reprezentációs),**
- ***fizikai*** **modell.**

Az egyes felhasználói *nézetek* (view-k) az adatbázisnak csak egy részét látják.

A leképezések (mapping-ek)



Egyed-kapcsolat leírások

Az alapvetően használt modell az *egyed-kapcsolat* (entity-relationship), azaz ER modell.
Az egyedeknek *tulajdonságaik* (attribute) vannak, ezek egyike, vagy ezekből egy készlet az egyedet egyértelműen azonosító *kulcs*.
A kapcsolatok lehetnek 1:1, 1:n, m:n jellegűek.
A kapcsolatokat sokszor egyedhalmazzá alakítjuk (kapcsoló rekord: a kapcsolat mint egyed).

Adatbázis felügyelő

Több felhasználó: védeni kell tőlük a rendszert, és őket egymástól.
Ez az *adatbázis felügyelő* (data base administrator) egyik feladata.

Tervezés: *séma, alséma* készítés.
Eszköz: DDL.

Adatfüggetlenség

***Logikai*: az adatok logikai struktúrájában (konceptuális, implementációs modell) történő változtatás ne érintse az ezen változtatást nem igénylő felhasználók munkáját.**

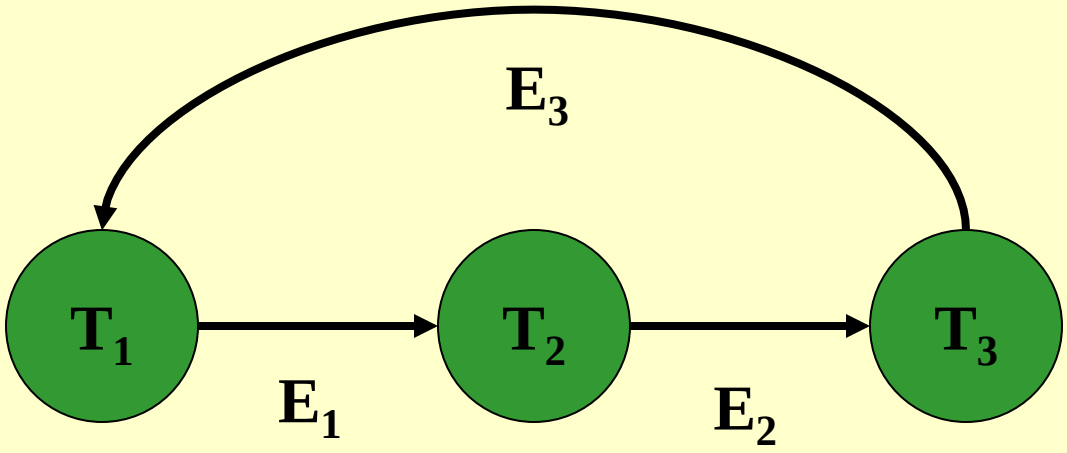
***Fizikai*: a fizikai modell változtatása ne kényszerítse ki az implementációs modell változtatását.**

A kettő együtt az *adatfüggetlenség*.

Tervezési szempontok

Kapcsolat múlttal, jövővel.
Biztonság, titkosság, pontosság.
Konkurrens folyamatok kezelése.
Válaszidő.
Kényelmes használat.
Költségek.

Várakozási gráf



Az ABF üzemeltetési feladatai

Mentések, karbantartás.

Kapcsolattartás a felhasználóval.

A DML

A felhasználó eszköze.

Többféle felosztás:

1. **Alkalmazott nyelv szerint:**
   - **Host language**
   - **Self Contained Language**
2. **A felhasználás jellege szerint:**
   - **Procedurális**
   - **Deklaratív**

Host Language (Beépített, beágyazott nyelvű) rendszerek

Egy korábban ismert magas szintű nyelvet használnak.

Megoldások:

Eljárás gyűjtemények könyvtárban.

Eljárás gyűjtemény előfordítóval.

Eljárás gyűjtemény interpreterrel.

Self Contained (Önálló) nyelvű rendszerek

Lehet külön programozható nyelv.

Lehet, csak paraméterezendő rendszer.

Lehet 4GL fejlesztő rendszer.

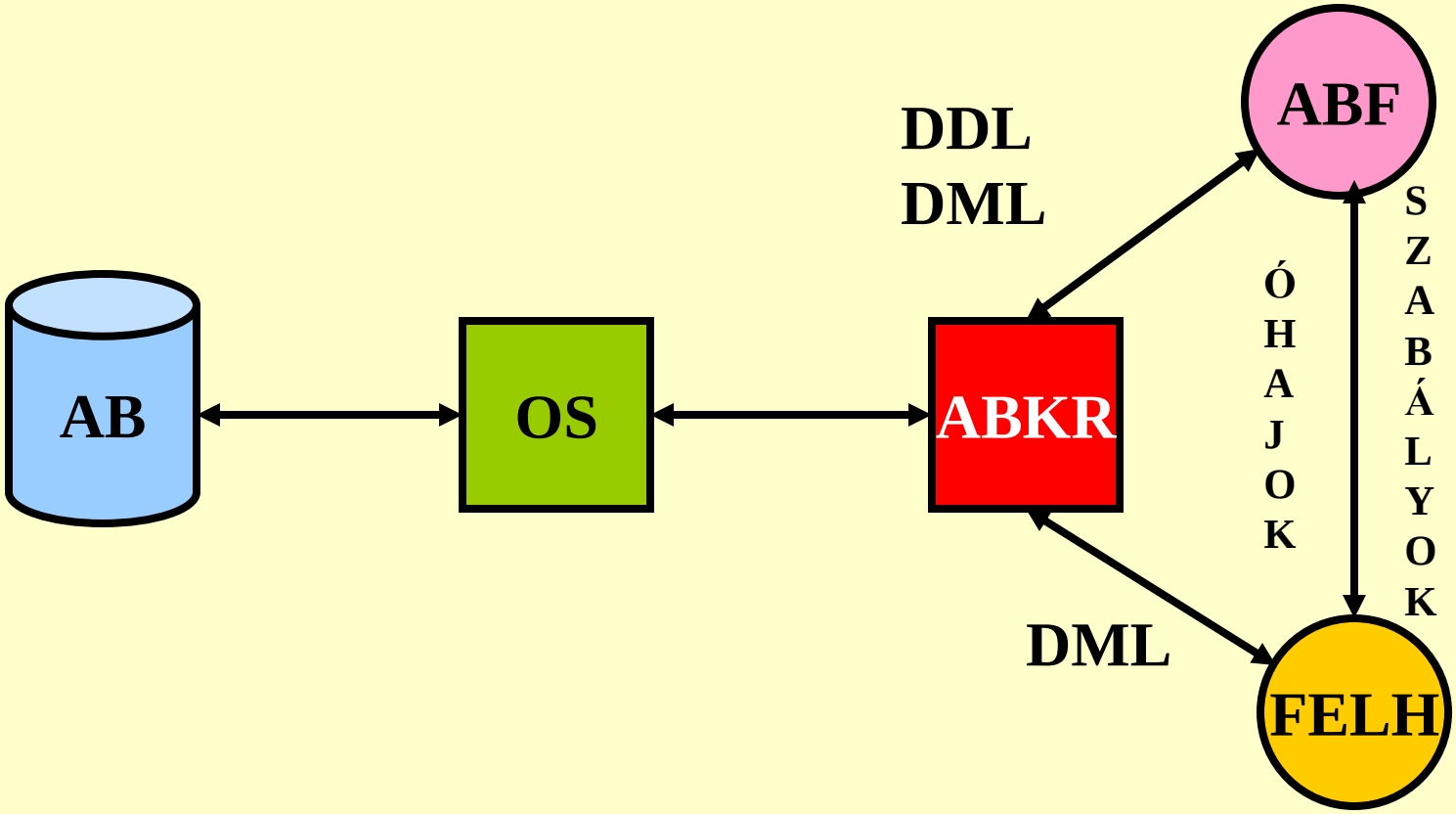
Procedurális rendszerek

Egy lépésben egy rekordot dolgoznak fel (record-in-time feldolgozás).

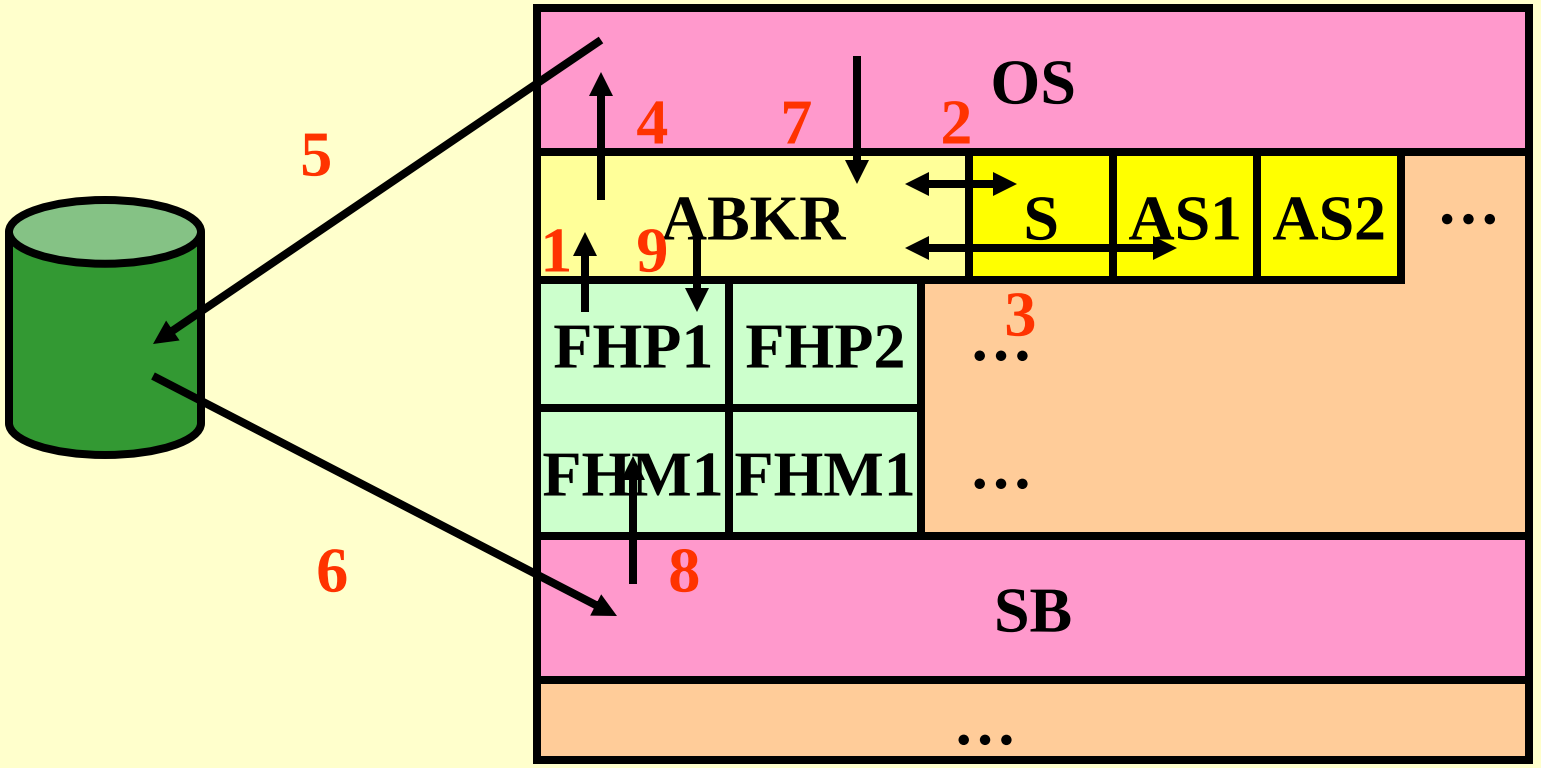
Deklaratív rendszerek

Egy lépésben egy meghatározott tulajdonságú halmazt (pl. egy táblázatot) dolgoznak fel(set-in-time rendszerek: ilyen pl. az SQL).

Az OS és az ABKR



Egy lekérés folyamata



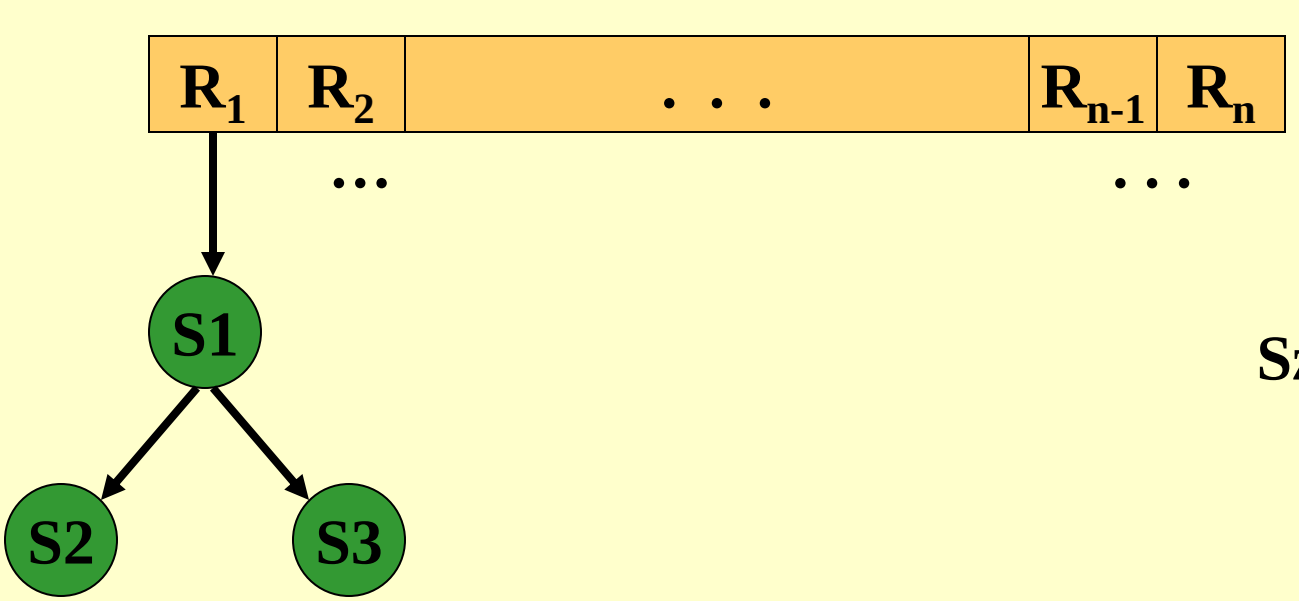
Alapvető ABKR modellek (approach)

1. **Hierarchikus**
2. **Hálós (CODASYL, DBTG)**
3. **Relációs**

A hierarchikus modell I.

A legelső modell, ma már nem használják.
Az adatokat fákban tároljuk, ahol a fák egy-egy szögpontja a *szegmens* (segment) adatokat és a további szegmensekre utaló mutatókat tartalmaz. A fák gyökérelemei hagyományos állományokba vannak szervezve. Az egyes nézetek a számukra *érzékeny szegmenseket* (sensitive segment) látják.
Jellegzetes képviselője az IMS (IBM).

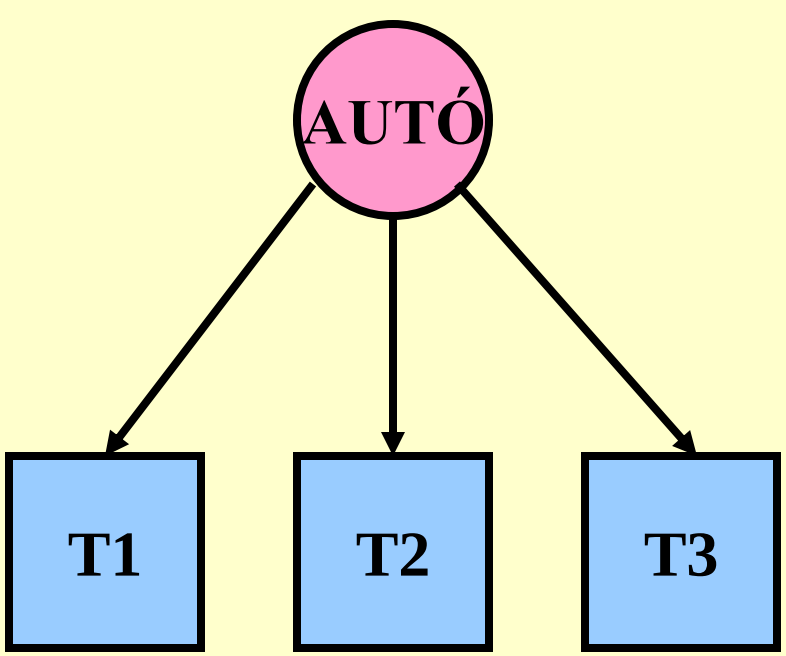
A hierarchikus modell II. (IMS)

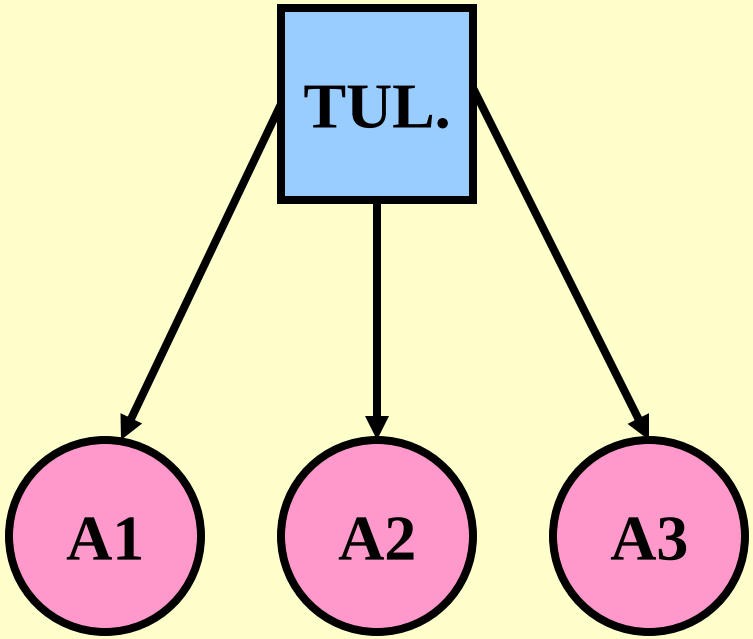


A hierarchikus modell tulajdonságai

Igazi ABKR, hisz többfelhasználós, az egyes felhasználók nem ugyanazt látják.
Nem igazi ABKR, mert a lekérdezés hatékonysága erősen függ az adatstruktúrától.

Lehetséges struktúrák egy hierarchikus modellben





A hálós modell 1.

A CODASYL bizottság által létrehozott DBTG (Data Base Task Group) jelentései (1969-1971) hozták létre.
Két évtizedig szinte kizárólag ezt használták.
Terminológiai és metodológiai javaslatokat tartalmaz.

A hálós modell 2.

Már bevezetett fogalmak:

DDL

DML

séma

alséma.

A hálós modell 3.

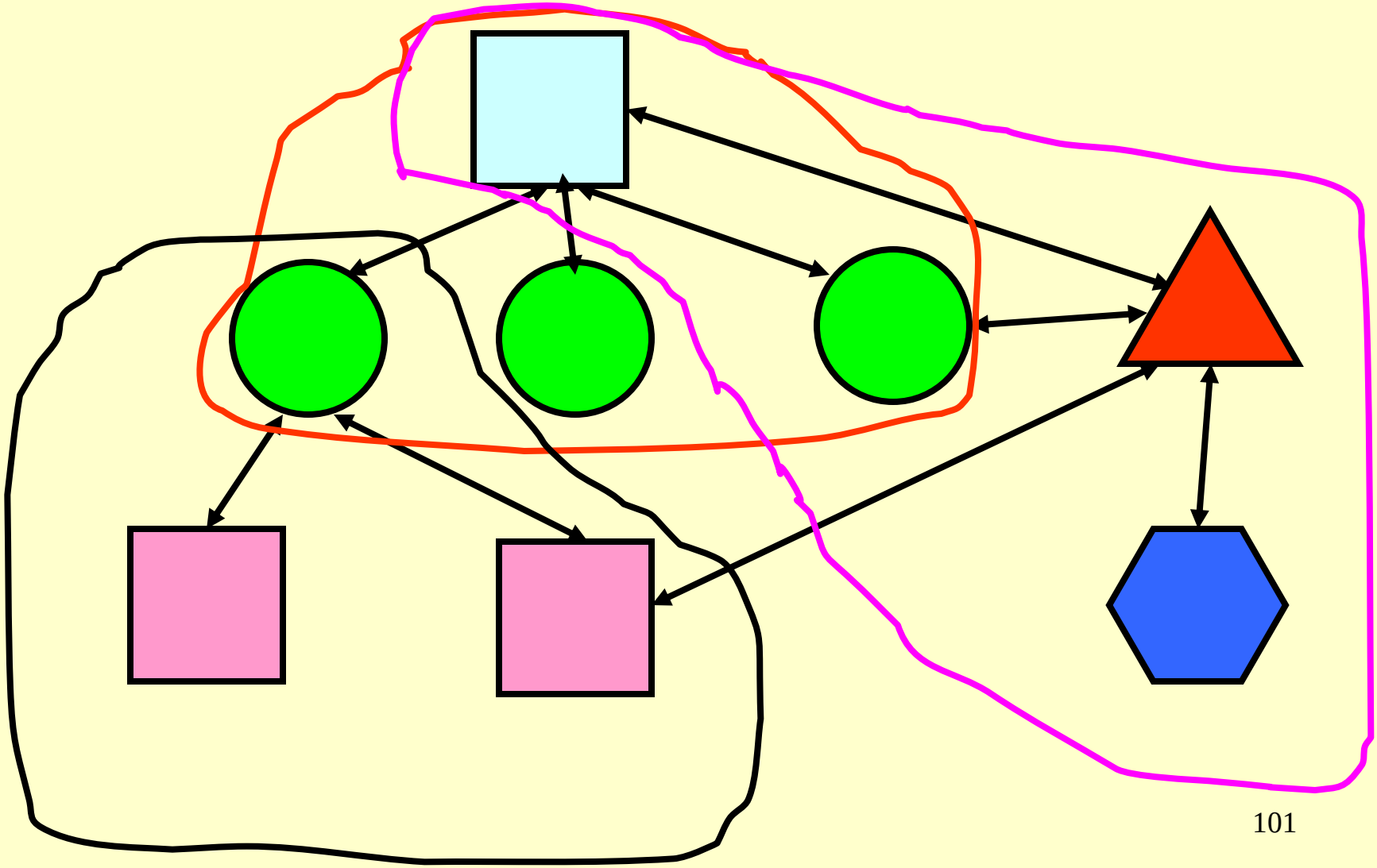
Új fogalmak:

Area: **valamilyen szempontból egységesen kezelendő adathalmaz.**

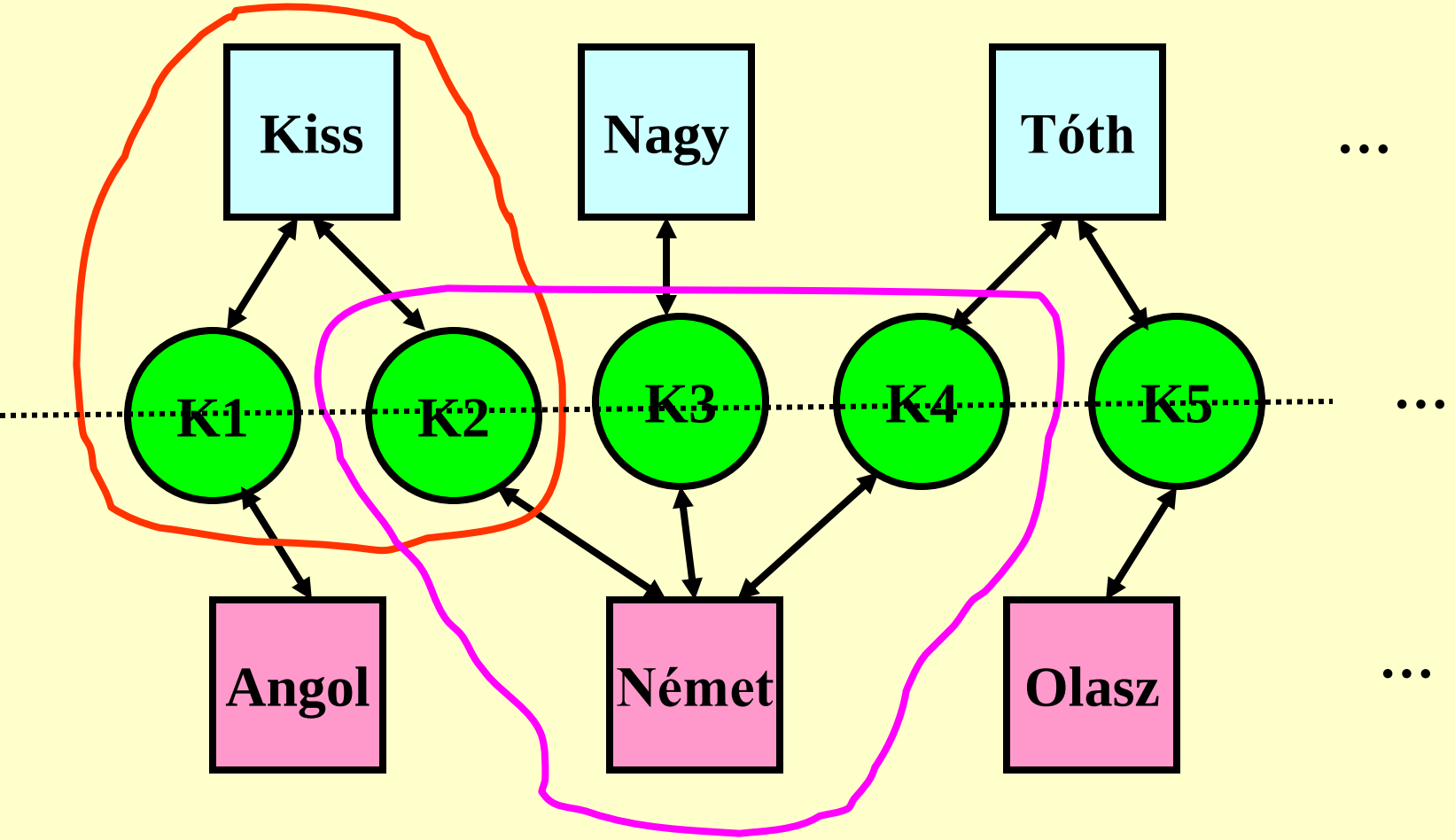
Set: **kétszintű fa, melynek gyökéreleme a *tulajdonos* (owner), levelei a *tagok* (members).**

A set-ek segítségével a legbonyolultabb hálós kapcsolatok is leírhatók.

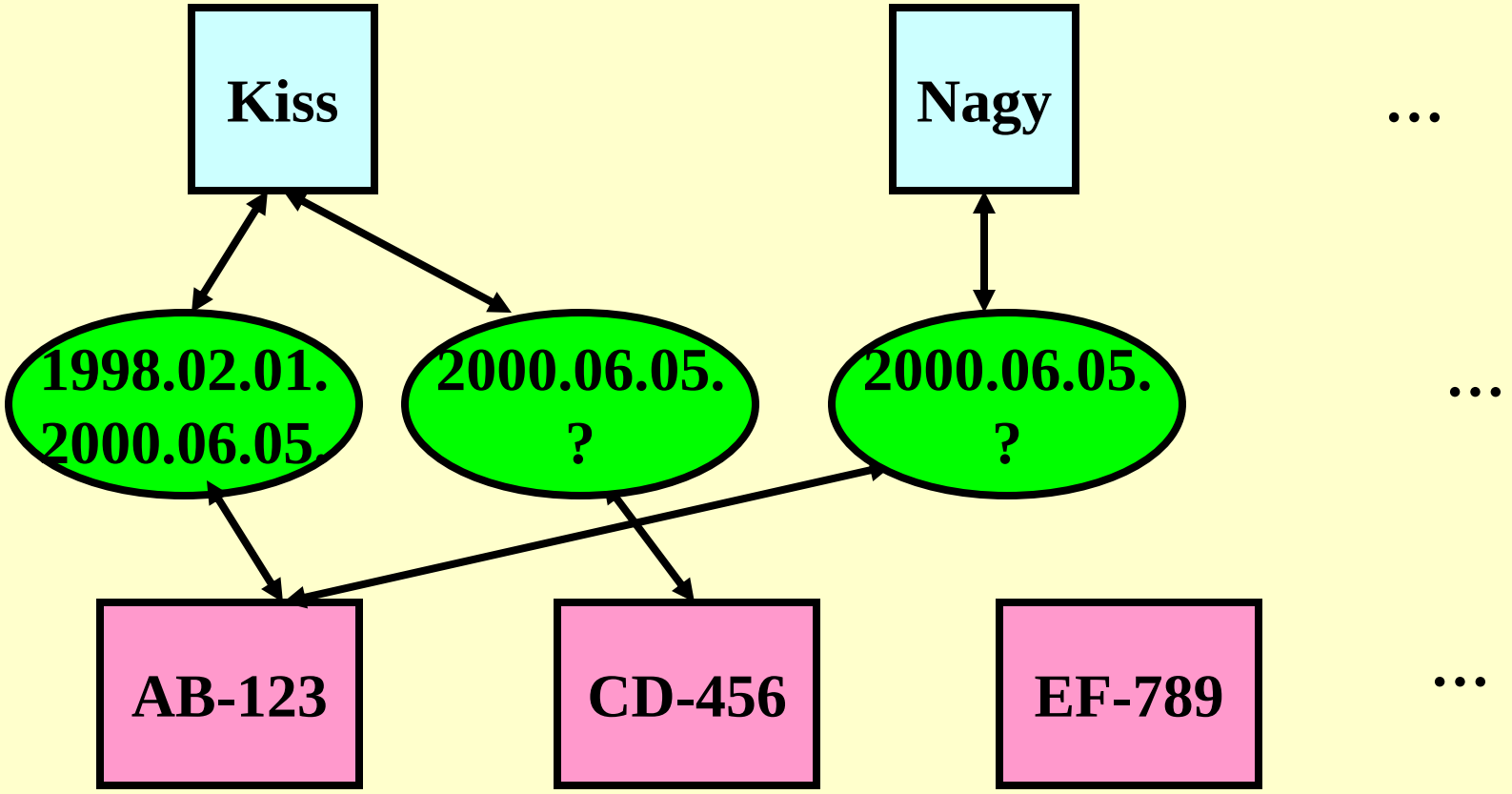
Példa SET-ekre



Kapcsolatok hálós modellben 1.



Kapcsolatok hálós modellben 2.



Séma hálós modellben

Séma név.
Zárak, kulcsok.
Rekord leírások (kalkulált mezők, kódolt mezők).
Kapcsolat leírások: SET-ek.
AREA leírások.

Alséma hálós modellben

A séma leírás valódi része.
Újdonság nem szerepelhet, de mindent át lehet nevezni.

Lekérdezés hálós modellben

Az eredeti javaslat a COBOL nyelvet javasolta (BATCH feldolgozás!), ezt váltotta fel a PL1, majd az interaktív felhasználás.

Fontos reprezentáns: IDMS (IBM 360 és 370: ESzR 1. és 2.).

Az IDMS specialitásai

Member-ek rendezése.
Indexek készítése.
Hashing algoritmus haználata.

A relációs modell 1.

A relációs modell ötlete Codd 1970-es cikkéből származik, így lényegében egyidős a DBTG jelentéssel.
Két évtized után átvette a vezető szerepet, ma szint kizárólag ezt használják, ezért ezt vizsgáljuk részletesen.

A relációs modell 2.

A *reláció* ebben az értelemben tulajdonképpen egy *táblázat* (table).
A táblázat oszlopai a *tulajdonság*ok (domains), a sorai az *n-es*ek (tuples).
Gyakran nevezik azonban a sorokat rekordnak, az egyes oszlopokhoz tartozó értékeket mezőnek (field).
Az R reláció sorainak hosszát r-el jelöljük.

Alapfeltételek relációkkal kapcsolatban

1. **Ne legyenek teljesen megegyező tartalmú sorok vagy oszlopok,**
2. **a sorok és az oszlopok sorrendje ne hordozzon információt.**

Azt a mezőt, vagy mezőkészletet, mely a sort (a sor többi elemét) egyértelműen meghatározza, *szuperkulcs*nak nevezzük. A nem szűkíthető szuperkulcs neve *kulcs*. (1. miatt van ilyen.)

Anomáliák

1. **Módosítási anomália**
   egy attribútum értéke több helyen szerepel, így több helyen kell módosítani (inkonzisztencia).
2. **Beírási anomália**
   hiányos adatok miatt valamit nem lehet bevinni (információvesztés).
3. **Törlési anomália**
   egy sor törlésével olyan információ is elvész, amire még szükség lenne (információvesztés).

Példa anomáliákra

Egy reláció oszlopai:

Tantárgyszám, tantárgynév, követelmény, oktatónév, oktatócím

- **Változtassuk meg az oktató címét.**
- **Mi történik, ha egy új tárgyhoz nincs oktató kijelölve?**
- **Mi történik, ha egy oktató kilép?**

Az anomáliák kiküszöbölése

A felsorolt anomáliák kiküszöbölhetők, a relációk olyan felbontásával (dekompozíció), melyek eredményei normalizált relációk.

1. Normálforma

A reláció 1NF értelemben normalizált, ha minden sorának minden mező értéke egy elemi érték.

(Újabb rendszerek lehetővé teszik a figyelmen kívül hagyását.)

2. Normálforma

A reláció 2NF értelemben normalizált, ha 1NF értelemben normalizált, és ha egy mezőérték meghatározásához összetett kulcs szükséges, egy másik mezőérték meghatározásához nem elegendő ennek egy része.

Példa 2. normálformára

cikkszám, ár, szállítási sorszám, cikkleírás

Anomália: ha az utolsó sort töröljük, elvész a cikkszám, cikkleírás közötti összefüggés.

Ok: az ár azonosításához szükséges a cikkszám, szállítási sorszám összetett kulcs, a cikkle1rás meghatározásához elég a cikkszám.

Megoldás: dekompozíció.

cikkszám, cikkleírás

cikkszám, ár, szállítási sorszám

3. Normálforma

A reláció 3NF értelemben normalizált, ha 2NF értelemben normalizált, és ha egy mezőérték sem függ olyan attribútum értéktől, vagy ezek olyan halmazától, mely nem kulcs.

Példa 3. normálformára

rendszám, gyártmány, típus, gyártási idő, szín

Ha egy időben csak egy színt használnak, elvész a gyártási idő és a szín közötti összefüggés.

Boyce-Codd (BCNF) normálforma

A 3NF tovább gondolása.

A reláció BCNF értelemben normalizált, ha 3NF értelemben normalizált, és ha egy összetett kulcs egy részét sem határozza meg más attribútum érték (nincs kulcstörés).

Ha egy reláció BNCF, akkor 3NF is.

Példa BCNF normálformára 1.

tantárgy, tanár, diák

Tegyük fel, hogy egy tanár csak egy tárgyat oktat.
Mi legyen a kulcs?

Egyszerű kulcs nem egyértelmű.

tantárgy, tanár nem határozza meg a diákot

tanár, diák nem 2NF (a tanár meghatározza a tantárgyat).

tantárgy, diák nem BCNF (a tanár meghatározza a tantárgyat).

Példa BCNF normálformára 2.

Anomália: törléskor eltűnik, hogy egy tanár milyen tárgyat tanít.

Megoldás: dekompozíció

tantárgy, <u>tanár</u>
tanár, <u>diák</u>

Többértékű függőség

Az $A_1A_2...A_n \rightarrow\rightarrow B_1B_2B_m$ többértékű függőség fennáll, ha

az R reláció soraiból tekintve azokat, melyek minden A_i értéken megegyeznek, ezek Bj-ken felvett értékei függetlenek az A-któl és B-ktől különböző attribútumok értékeitől.

A többértékű függőség nem triviális, ha

1. **egyik B sincs az A-k között,**
2. **az A-k és a B-k együttesen sem tartalmazzák R összes atribútumát.**

4. Normálforma

Egy reláció 4NF-ben van, ha valahányszor az $A_1A_2...A_n \rightarrow\rightarrow B_1B_2B_m$ nem triviális többértékű függőség fennáll, akkor $\{A_1A_2...A_n\}$ szuperkulcs.

Ha a reláció 4NF, akkor BNCF is.

Példa többértékű függőségre I.

Pl. legyenek az R reláció oszlopai:

Tanár Tantárgy Kar

Egy tanár több tárgyat taníthat, egy tárgyat többen taníthatnak, egy tárgyat több karon tanítanak, egy karon több tárgyat tanítanak.

Tanár$\rightarrow\rightarrow$Tantárgy és Tantárgy$\rightarrow\rightarrow$Kar fennáll, és nem triviális.

Példa többértékű függőségre II.

Tanár	Tantárgy	Kar
...		
Nagy	AB	KGK
Nagy	AB	NIK
Kiss	PPT	NIK
...		

BCF, de nem redundancia mentes:

pl. egy Tanár → Tantárgy bejegyzés többször is előfordulhat.

Tanár Tantárgy nem szuperkulcs, azaz nem 4F.

Példa többértékű függőségre III.

Megoldás itt is a dekompozíció:

Tanár Tanfolyam

Tanár Kar

Tárgy Kar

Vigyázat:

Tanár Tárgy

Tanár Kar

nem jó, mert elvész, hogy melyik tanár melyik karon mit tanít.

További normálformák

Létezik 5NF, de kisebb jelentősége miatt nem foglalkozunk ezzel.

A lekérdezés elve relációs rendszerekben

1. relációs algebra
2. relációs kalkulus

A relációs algebra 1.

Alapműveletek:

1. **Unió:** $R \cup S$
2. **Különbség:** $R - S$
3. **Direkt szorzat:** $R \otimes S$
4. **Projekció:** $\Pi_{i1,i2...ik}(R)$
5. **Szelekció:** $\sigma_F(R)$

ahol: Az F feltétel az [i] Θ [j], és az [i] Θ c elemi kifejezések logikai műveletekkel ($\wedge$, $\vee$, $\neg$) való összekötéséből állhat. Θ tetszőleges összehasonlító operátort (=, ≠ , < , > , ≤ , ≥) jelenthet.

A relációs algebra 2.

Következmény műveletek:

6. Metszet: $R \cap S$

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

7. Hányados: $R \div S$

Feltesszük, hogy $r > s$, és $s \neq 0$.)

Legyen

$T = \Pi_{1,2,\ldots r-s}(R)$, **továbbá**

$V = \Pi_{1,2,\ldots r-s}((T \otimes S)-R)$.

Ekkor belátható, hogy

$R \div S = T - V$.

A relációs algebra 3.

8. Feltételes kapcsolat (Θ join):

$$R \underset{[i]\,\Theta\,[j]}{*} S = \sigma_{[i]\,\Theta[r+j]}(R \otimes S)$$

9. Természetes kapcsolat (natural join): R * S

Hasonló a feltételes kapcsolathoz de itt a szelekció feltétele az, hogy az azonos nevű oszlopokban azonos érték szerepeljen.

Példa relációs algebrai műveletekre 1.

Számoljuk ki az R ÷ S műveletet eredményét!

$R =$

a	b	c	d
a	b	e	f
b	c	e	f
e	d	c	d
e	d	e	f
a	b	d	e

$S =$

c	d
e	f

Példa relációs algebrai műveletekre 2.

Számoljuk ki az $R *_{B<D} S$ művelet eredményét!

R =

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

S =

D	E
3	1
6	2

Példa relációs algebrai műveletekre 3.

Számoljuk ki az R * S művelet eredményét!

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
d	d	b

Példa relációs algebra alkalmazására 1.

Tekintsük az alábbi három relációt.

a.) Autók: jele A

Mezői: rendszám, gyártmány, típus, szín, ...

b.) Emberek: jele E

Mezői: szemszám, név, foglalkozás, cím, ...

c.) Kapcsolat: jele K

Mezői: forgrsz, szemszám.

Feladat: **keressük ki a sárga opelek tulajdonosainak foglalkozását.**

Példa relációs algebra alkalmazására 2.

A megoldás:

$$R1 = \sigma_{\text{szín} = \text{sárga} \wedge \text{típus} = \text{opel}} (A)$$

$$R2 = \Pi_{\text{rendszám}} (R1)$$

$$R3 = R2 * K_{\text{rendszám}=\text{forgrsz}}$$

$$R4 = \Pi_{\text{szemszám}} (R3)$$

$$R5 = R4 * E$$

$$R6 = \Pi_{\text{foglalkozás}} (R5)$$

Példa relációs algebra alkalmazására 3.

A megoldás egy lépésben:

$$\Pi_f(\Pi_{szsz}(K *_{fsz=rsz} (\Pi_{rsz} (\sigma_{sz = sárga \wedge t = opel} (A)))) * E)$$

(foglalkozás→f, szemszám →szsz, forgrsz →fsz, rendszám →rsz, szín →sz, típus →t helyettesítéssel).

A relációs kalkulus 1.

Létezik sor- és oszlopkalkulus, mi csak az előbbivel foglalkozunk.
A kalkulus az alábbi alakú kifejezésekből áll:

$$\{t \mid \psi(t)\}$$

melynek jelentése: azok a t sorok, melyek kielégítik a ψ(t) függvényt, azaz amelyekre a ψ(t) függvény értéke igaz.

A relációs kalkulus 2.

Atomi formulának, vagy atomnak nevezzük, az alábbi kifejezéseket:

1.) R(s) **(azaz az s sor R relációbeli),**

2.) s[i] Θ u[j] **(ahol s[i] az s-edik sor i-edik elemét, u[j] az u-adik sor j-edik elemét jelenti, Θ a szokásos összehasonlító operátor),**

3.) s[i] Θ c **(ahol c egy konstans).**

A relációs kalkulus 3.

A ψ(t) függvény, melyet *formulának* nevezünk, az alábbiak szerint épülhet fel:

1.) Minden atom formula.

2.) A formulákból a ∧, ∨ , ¬ logikai műveletekkel képezett kifejezések is formulák.

3.) A (∃s)(ψ) alakú (van ilyan s, hogy ψ igaz) kifejezések is formulák.

A relációs kalkulus 4.

4.) A $(\forall s)(\psi)$ alakú (minden s olyan, hogy ψ igaz) kifejezések is formulák.

5.) A kifejezések a precedencia (Θ, $\exists$, $\forall$, $\neg$, $\wedge$, $\vee$) megváltoztatása érdekében zárójelezhetők.

6.) Más formula nincs.

A relációs algebra és kalkulus összehasonlítása 1.

A relációs algebra alapműveletei kifejezhetők a relációs kalkulus eszközeivel.

Unió: $\{t \mid R(t) \vee S(t)\}$

Különbség: $\{t \mid R(t) \wedge \neg S(t)\}$

Direkt szorzat: $\{t^{(r+s)} \mid (\exists u^{(r)})\ (\exists v^{(s)})(R(u) \wedge S(v) \wedge t[1]=u[1] \wedge \ldots \wedge t[r]=u[r] \wedge t[r+1]=v[1] \wedge \ldots \wedge t[r+s]=v[s]\ \}$

Projekció: $\{t^{(k)} \mid (\exists u)(R(u) \wedge t[1]=u[i_1] \wedge \ldots \wedge t[k]=u[i_k]\ \}$

Szelekció: $\{t \mid (R(t) \wedge F\ \}$

A relációs algebra és kalkulus összehasonlítása 2.

$\{t \mid \neg R(t)\}$
értelmetlen, de nem írható le a rel. algebrában.

DOM fv.:
A DOM(ψ) azon elemek halmaza, melyek közvetlen, vagy közvetett módon ψ-t alkotják.

A relációs algebra és kalkulus összehasonlítása 3.

Biztos kifejezések:

1. **Ha t kielégíti ψ(t)-t, t minden komponense DOM(ψ)-beli.**
2. **ψ minden (∃u)(ω(u)) alakú részkifejezésére, ha u kielégíti ω(u)-tu minden komponense DOM(ω)-beli.**

A rel. kalkulus biztos kifejezései ekvivalensek a rel. algebra kifejezéseivel.

Lekérdezés relációs rendszerekben 1. (ISBL)

Korai IBM termék.

Lényegében a relációs algebra kifejezéseinek linearizálása.

Nem felhasználóbarát.

Lekérdezés relációs rendszerekben 2. (QBE)

IBM termék.

Tábla vázakat (skeleton) használ, ezeket vektorváltozók segítségével lehet összekapcsolni.

Példa QBE-re 1.

AUTÓK

rendsz.	gym.	szín	. . .
	OPEL	SÁRGA	

Példa QBE-re 2.

AUTÓK

rendsz.	gym.	szín	. . .
X	OPEL	SÁRGA	

Példa QBE-re 3.

AUTÓK

rendsz.	gym.	szín	. . .
X	OPEL	SÁRGA	

KAPCSOLAT

frsz.	sz.sz.	. . .
X	Y	

Példa QBE-re 4.

AUTÓK

rendsz.	gym.	szín	. . .
X	OPEL	SÁRGA	

KAPCSOLAT

frsz.	sz.sz.	. . .
X	Y	

EMBEREK

sz.sz.	név	fogl.	. . .
Y	Kiss	tanár	
	…	…	

Lekérdezés relációs rendszerekben 3. (SQL)

Structured Query Language

Több részből áll:

1. DDL

2. DCL

3. DML

4. Query

DDL

CREATE TABLE
ALTER TABLE
DROP TABLE

CREATE VIEW
DROP VIEW

CREATE [UNIQUE] INDEX
DROP INDEX

DCL

Tranzakció kezelés:
COMMIT
ROLLBACK
LOCK
UNLOCK

Jogosítvány kezelés:
GRANT …. WITH GRANT OPTION
REVOKE

DML

INSERT
UPDATE
DELETE

QUERY

A SELECT utasítás valósítja meg.
Legegyszerűbb formája:

select [distinct] ***oszlopnevek*** **from** ***táblanevek***

[where ***feltétel*****]**

[order by ***rendezési feltétel*****]**

[group by ***feltétel*** **[having** ***having- feltétel*****]]**

Keresés egymásba ágyazott SELECT-ek esetén

A WHERE részben alkalmazható újabb SELECT klauzula:

1. [NOT] IN (selectkifejezés)

2. Θ[ANY|ALL] (selectkifejezés)

3. [NOT] EXISTS

Több SELECT esetén a kiszámítás belülről kifelé történik.

Beágyazott (embedded) SQL

1. EXEC SQL <beágyazott SQL utasítás>

2. Változók használata V→:V

3. Vezérlés:

$$\text{WHENEVER} \left\{ \begin{matrix} \text{NOT FOUND} \\ \text{SQL WARNING} \\ \text{SQL ERROR} \end{matrix} \right\} \left\{ \begin{matrix} \text{GO TO cimke} \\ \text{CONTINUE} \\ \text{STOP} \end{matrix} \right\}$$

4GL program generátorok

1. **FORM (űrlap)**
2. **REPORT (jelentés)**
3. **MENU**

Továbbfejlesztett modellek

1. **Az EER modell.**
2. **Nested Relational Model.**
3. **Structural Data Model.**
4. **Objektum-orientált adatbázisok.**
5. **Deduktív adatbázisok.**

Az EER modell

Subclass, superclass bevezetése.
(Közös tulajdonságok, kapcsolatok leírása.)

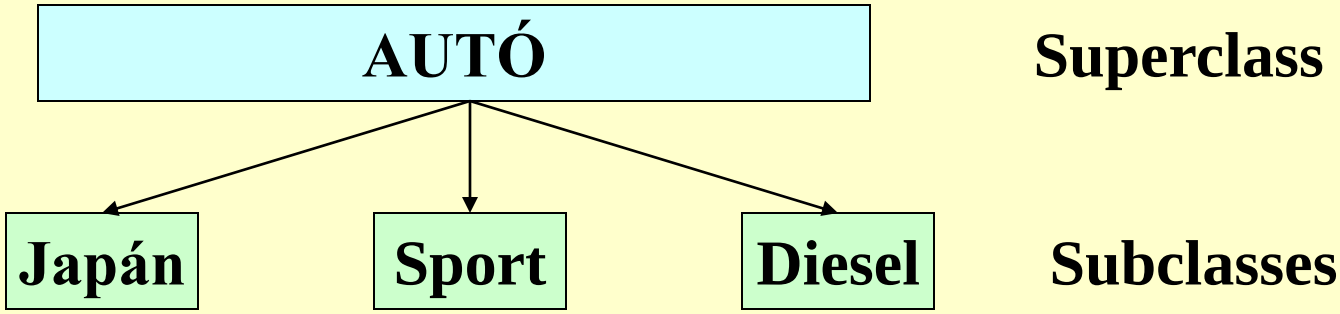
A hierarchiában a superclass tulajdonságai öröklődnek (inheritance).

Relációkkal megvalósítható.

Az EER modell felosztása

1. A subclass lehet attribútum által meghatározott, vagy felhasználó által definiált.
2. A felosztás lehet diszjunkt (disjoint), vagy átfedő (overlap).
3.) Ugyancsak lehet a felosztás teljes (complett) vagy részleges (partial).

Példa EER modellre

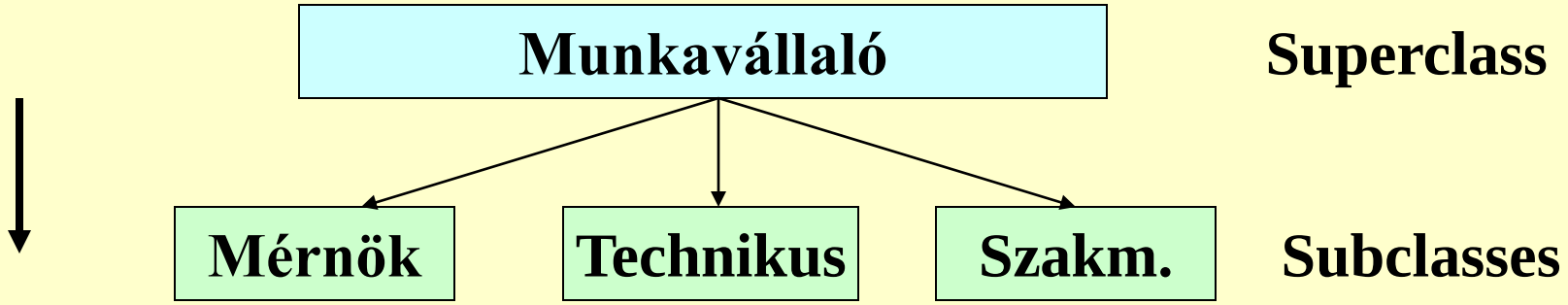


A subclass-ok nem diszjunktak,
a felosztás részleges!

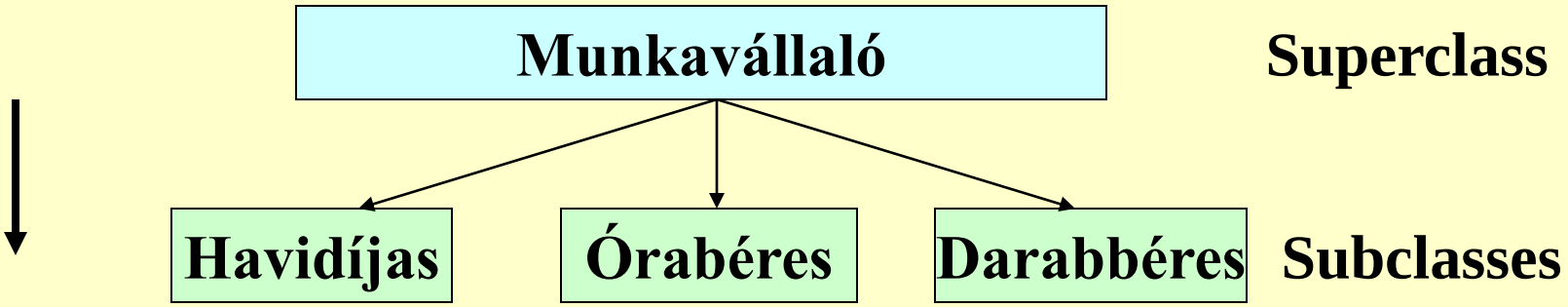
Az EER modell használata

1.) Specializáció
Megkeressük az egy-egy csoportra jellemző tulajdonságokat.
2.) Generalizáció.
Megkeressük a közös tulajdonságokat.

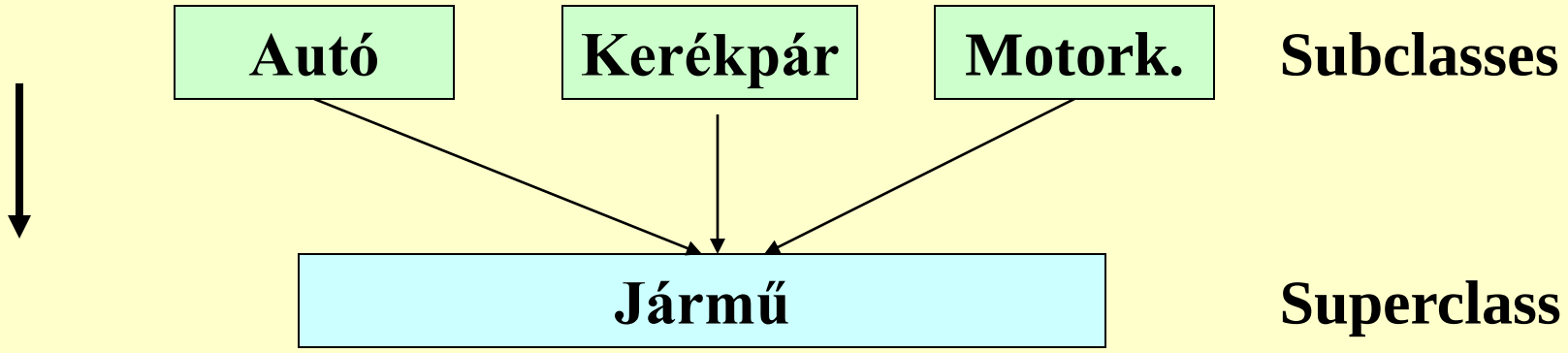
Példa Specializációra



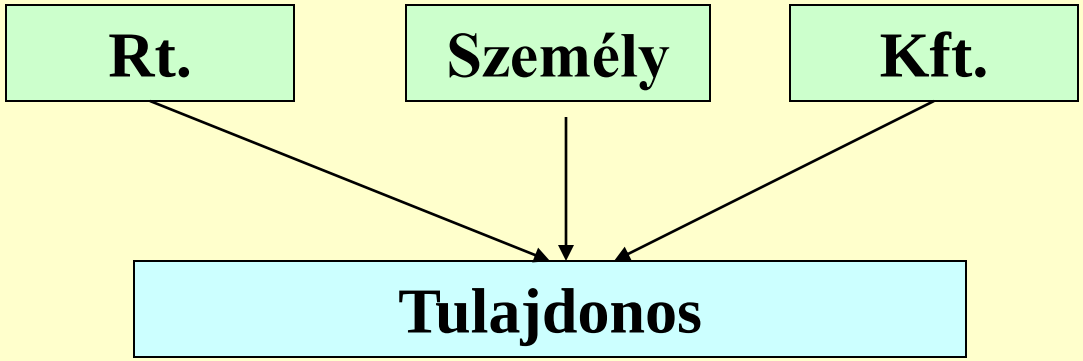
Lehet több szempont szerint is:



Példa Generalizációra



Kategória



Kategória az a subclass, melynek több superclass-a van.
Itt az öröklés szelektív lehet.

Nested Relational Model

Nem 1NF, ezért nevezik N1NF modellnek is.

Egy osztály sémája

<table>
<tr><td rowspan="3">Oszám</td><td rowspan="3">Onév</td><td rowspan="3">Ovez</td><td colspan="3">Dolgozó</td><td rowspan="3">Ocím</td></tr>
<tr><td rowspan="2">Dnév</td><td colspan="2">Gyerekek</td></tr>
<tr><td>Gynév</td><td>Gykor</td></tr>
</table>

Structural Data Model 1.

Az eredeti relációs modell továbbfejlesztése, az SQL2 alapja.

Két típus:
1.) Relations
2.) Connections

Structural Data Model 2.

Relations

1.) Primary: nem kapcsolódik hozzá semmi.

2.) Referenced: kapcsolódik hozzá reláció.

3.) Nest: többértékű attribútumot tartalmaz.

4.) Association: M:N kapcsolatot reprezentál.

5.) Lexicon: egy az egyhez kapcsolat az attribútumok között.

6.) Subrelation: csak egy másikkal együtt van értelme.

Structural Data Model 3.

Connections

1.) Ownership: tulajdonos és a hozzátartozó nest, vagy association reláció között.

2.) Reference: referenced relations-ok között.

3.) Identity: reláció és subrelation között, azaz ahol a primary key és a foreign key azonos.

OODB 1.

Az ötlet az OOP alapján keletkezett.

Jellegzetességek:

1.) Az objektumosztályok perzisztensek.

2.) Az objektumosztályok osztottak.

3.) Minden objektumnak külön azonosítója (OID) van (nem azonos a kulccsal, az különbözhet relációnként).

4.) Megengedettek a bonyolult objektumok (pl. egy objektumon belüli több reláció, stb.).

OODB 2.

Encapsulation

Az értékeket u.n. *instance variable*-ok tartalmazzák. ez hasonlít az attributum –hoz, de kívülről rendszerint nem látható, csak az előre definiált *operációk* férhetnek hozzá (metódus).

Egy operációnak két része van:

***1.)Signature*, vagy *interface*: a név és a paraméterek.**

2.)*Method*, vagy *body*: az implementáció.

OODB 3.

Inheritance, polymorphism

A szokásos. Érdekes az operátorok polimorfizmusa, azaz az a tulajdonság, hogy egy operátor névhez többféle implementáció tartozhat az objektum típu-sától függően (másnéven *operator overloading*. Kell hozzá a late binding!).

Kapcsolatok

Az encapsulation miatt nehézségek adódhatnak.

1. **Kapcsolatot kereső metódusok.**
2. **References:az objektum tartalmazza a kapcsolt objektumok OID-jét.**

Az O2 adatdefiniálás I.

A séma objektum típusokat és osztályokat definiál.
Az objektum típusokat az u.n. atomi típusok használatával és az O2 típus konstruktorok alkalmazásával definiálhatjuk.
Atomi típusok: Boolean, character, integer,real, string és bits.
A típus konstruktorok: tuple, list, set, unique set.
Az O2 metódusok nem részei a típus definíciónak.
Az osztály definíció két részből áll:típus és metódus megadásból.

O2 adatdefiniálás II.

A Különbség van az O2-ben értékek (value) és objektumok között.

Az értéknek csak típusa van, és önmagát reprezentálja.

Az objektum egy osztályhoz tartozik és így van egy típusa és vannak metódusai, melyeket az osztály határoz meg.

Mind az értékek, mind az objektumok komplex típusúak, és ezek lehetnek értékek, vagy lehetnek hivatkozások más objektumokra az OID használatával.

O2 adatdefiniálás III.

Az O2-nek van egy saját nyelve az O2C, ezen lehet definiálni osztályokat, metódusokat, típusokat, továbbá létrehozni (create) objektumokat és értékeket.

Az objektumok lehetnek perzisztensek (állandóan tárolva vannak az adatbázisban) és tranziensek (csak egy program végrehajtása alatt léteznek), az értékek tranziensek, (hacsak nem részei egy perzisztens objektumnak).

O2 adatdefiniálás példa

```
type telefon: tuple     (körzetszám: integer,
                         telefonszám: integer);

class személy:
        type tuple(      név: tuple (vezetéknév:string,
                                     keresztnév:string),
                         szülinap: Date,
...
        method           kor:integer
end
```

Az O2 adatmanipulálás

Adatmanipulálási lehetőségek:

- **az O2SQL lekérdező- és az O2C program nyelv,**
- **beágyazott pl. C++-ba.**

Fejlesztő környezetek:

- **O2Look (interface O2C-hez),**
- **O2Tools (grafikus fejlesztői környezet).**

Az Objectstore rendszer

A C++ nyelvhez készült, annak objektum deklarációs utasításait használja.

Adatmanipuláláshoz is a C++-t használhatjuk, de van grafikus interface is.

Deduktív adatbázisok

A logikai programozás eszközeivel dolgozik. Deklaratív nyelvet (pl. PROLOG) használ. ezen írja le a *tényeket* és a *szabályok*at.

Adatbázis kezelő architektúrák

Három fő rész:

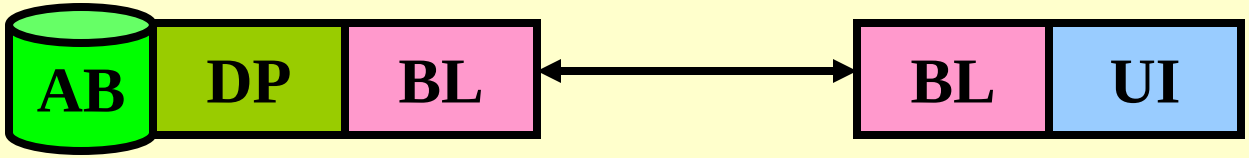
1.) Data processing (fizikai adatkezelés).

2.) Business logic (adatvédelem és –integritás, tranzakció kezelés, stb.).

3.) User interface

Hol helyezkednek el?

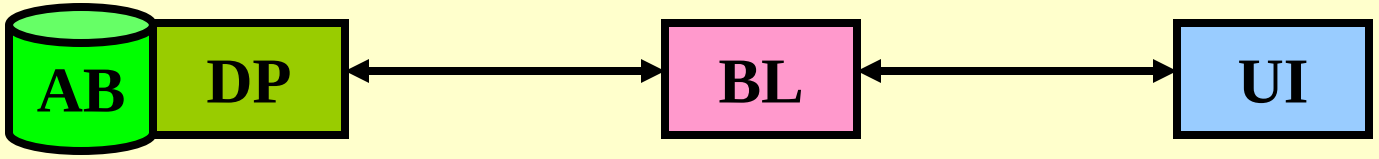
Kliens-szerver architektúra



Adatbázis szerver

Kliens alkalmazás

Többrétegű architektúra



Adatbázis szerver

Középső réteg (middle-tier)

„Sovány” („Thin”) kliens alkalmazás

Osztott adatbázisok

Megnövekedett a kommunikációs költségek részaránya. Javaslat: helyezzük az adatokat a felhasználóhoz közel.

Osztott (distributed) adatbázis:

fizikailag megosztott, de logikailag egységes adatbázis.

Adatbázis felügyelet: központi, csomóponti.

Osztott adatbázisok előnyei

1.) A kommunikációs költségek csökkenése.
2.) Mindenki a számára ismerős adatokat gondozza.
3.) Egy-egy csomópont kiesése esetén a többi adatai továbbra is elérhetőek.
4.) Lehetséges a moduláris tervezés, a rugalmas konfigurálás.
5.) Hosszabb idő alatt a rendszer gépei akár ki is cserélhetők.

Osztott adatbázisok hátrányai

1.) A rendszer bonyolultabb és sebezhetőbb lesz,
2.) Nem könnyű minden csomópontra egyformán jó személyzetet találni, másrészt, ha találunk fenyeget a szuboptimalizáció veszélye.
3.) Mindig valamennyi gépnek működnie kell.
4.) Többféle hardvert és szoftvert kell a rendszernek kezelnie és "összehoznia".
5.) Bonyolult a jogosultságok ellenőrzése.

Osztott adatbázisok konzisztenciája

Külön problémát jelent, ha feladjuk a redundancia-mentesség elvét. Erre okot szolgáltathat az is, ha nem eldönthető egy-egy adatról, hogy hol használják legtöbbet, de biztonsági okokból is dönthetünk egy-egy adat többszörözése mellett. Ilyen estekben biztosítanunk kell, hogy az egyes példányok tartalma azonos legyen, azaz a rendszer *konzisztens* maradjon.

Elemzések

1. Forrás nyelő elemzés

   A használat gyakorisága, módja.

2. ABC elemzés

   A „nélkülözhetetlenség” foka.

3. Érzékenység elemzés

   A költség/teljesítmény arány.

Konzisztencia, konvergencia

Létezik

- **Erős konzisztencia**
- **Gyenge konzisztencia**

Koherencia: a konzisztencia mérő száma, mely erős konzisztenciánál azonosan 1 értékű, gyenge konzisztenciánál pedig 1-hez tart.

Szinkronizációs protokollok

A. Központosított protokollok

a.) A központi zárellenőrzés

b.) A zseton módszer

c.) Az elsődleges példány módszer

B.) Osztott protokollok

Az időbélyeg módszer

Adatvédelem

a.) Fizikai védelem

b.) Ügyviteli védelem

c.) Algoritmikus védelem

Algoritmikus védelem

a.) felhasználó azonosítás, partner azonosítás

b.) hozzáférés védelem

c.) rejtjelezés

d.) üzenethitelesítés

e.) digitális kézjegy

Rejtjelezés

$y = E(x)$ → Védetlen közeg → $x = D(y)$

Legyen „megfejthetetlen”.

Mivel sok kell, legyen:

$y = E(k_r,x)$ → Védetlen közeg → $x = D(k_f, y)$

A rejtjelezés felosztása

1. **Konvencionális a rejtjelezés, ha a rejtő kulcsból a fejtő kulcs meghatározható.**

2. **Nyílt (nyilvános) kulcsú a rejtjelezés, ha rejtő kulcsból a fejtő kulcs nem határozható meg.**

A konvencionális kódolás

a.) Helyettesítés

b.) Periodikus helyettesítés

c.) Kulcsfolyam(at)os rejtés

d.) Rejtjelötvözés vagy keverő transzformációk

(Shannon, 1949)

Lucifer (128 bites blokk, 128 bites kulcs)

DES (64 bites blokk, 56 bites kulcs)

A nyilvános (rejtő) kulcsú kódolás

a.) MIT módszer (prímfelbontás).

b.) Merkle-Hellmann módszer (hátizsák problé-ma).

Kulcsgondozás

Ha a kulcsok megfelelő védelme nem biztosított, az egész kódolási rendszer értelmetlen. A kulcsok gondozása három feladatból áll.

a.) kulcsgenerálás

b.) kulcskiosztás

c.) kulcstárolás

Kulcsgenerálás

Az a művelet, melynek eredményeképpen a kulcsok előállnak. Egy valódi véletlenszám generátor segítségével végrehajtható.

Kulcskiosztás

Példák:

a.) Alapkulcsok

b.) Merkle „rejtvény” módszere

c.) A "hatványozós" módszer

Alapkulcsok

A kulcsok kiosztása történhet egy *alapkulcs készleten* keresztül, melyet rendszeren kívüli eszközökkel juttatnak el a résztvevőkhöz. Az alapkulcsokat, melyek gyakran nyilvános kulcsú rendszerhez tartoznak, csak a kulcsok cseréjéhez használják.

Merkle „rejtvény” módszere

A hívó fél n db (K_i, I_i) párt küld partnerének gyengén kódolva. Ez egyet kiválaszt, feltöri, és I_i-t visszaküldi. Ezzel a kommunikáció kulcsa meghatározott. A behatolónak átlagosan a párok felét kell ahhoz feltörnie, hogy megtudja, I_i-hez melyik K_i tartozik.

A "hatványozós" módszer 1.

A módszer alapja, hogy az i és a j felhasználó kitalál egy-egy x_i ill. x_j számot.

Egymás között kicserélik az

$$Y_i = \alpha^{x_i} \pmod p,$$

ill. az

$$Y_j = \alpha^{x_j} \pmod p$$

számokat (p prímszám, a p elemű véges test egy primitív eleme).

A "hatványozós" módszer 2.

A a kommunikáció kulcsa a

$$K = \alpha^{x_i * x_j}$$

szám (vagy annak valamilyen függvénye) lehet. Ennek előállítása α mellett Y_j és x_i vagy Y_i és x_j ismeretében egy egyszerű hatványozást igényel, a behatolónak viszont a diszkrét logaritmusképzés a feladata.

Kulcstárolás

Nehézségeket okozhat, ha a kulcsokat akár túl kevés, akár túl sok ember ismeri.

Megoldást jelentenek az u.n. *(n,k) küszöbrendszerek*. E rendszerek lényege, hogy a kulcsot n db. (nem feltétlenül diszjunkt) részre osztva, bármelyik k kulcsrészletből a kulcs előállítható, de nincs olyan k-1 kulcs-részlet, amiből ez megtehető lenne. Ilyen küszöbrendszerek készítésére több matematikai módszer is rendelkezésünkre áll.

Felhasználó azonosítás

Személy azonosítás:

a.) jelszóvédelem

b.) fizikai azonosító használata

c.) személyi jellemzők

Partner azonosítás 1.

A számítógép-számítógép kapcsolatokban is szükség lehet azonosításra. E célra fenntarthat minden számítógép pár egy-egy kulcsot, ez azonban egy n elemű hálózatnál n^2 kulcsot jelent. Folyhatnak az információcserék valamilyen hitelesítő központon keresztül, ez azonban a kommunikáció költségét növeli jelentősen.

Partner azonosítás 2.

Megoldás:ha minden csomópont egy, csak a hitelesítő központ által ismert kulccsal tud e központhoz bejelentkezni, s üzenet továbbítási igényét bejelente-ni. A központ jelöli ki a kommunikáció kulcsát, melyet a fenti kulccsal kódolva a hívónak visszaküld. Emellett küld egy csomagot, mely a hívandó fél kulcsával kódolva tartalmazza a kommunikáció kulcsát, s a hívó megjelölését. Ha a hívó e csomagot a hívott félhez továbbítja a párbeszéd kettejük között folytatódhat, s az azonosítás is kielégítő biztonsággal történt meg.

Hozzáférésvédelem

Nem elegendő kiszűrni a jogosulatlan behatolókat, a rendszernek azt is számon kell tartania, hogy a jogos felhasználók hatásköre mire terjed ki. A felhasználó által működtetett *ügynök folyamatok* hatáskörét a *hozzáférési mátrix* szabja meg. Ennek elemeit akár ügynökökhöz, akár adatokhoz kötötten tárolhatjuk.

Üzenethitelesítés

Az üzenetek hitelesítéséhez két feltétele van, egyrészt ellenőrizni kell, hogy egy-egy blokkban az és csak az érkezett-e a címzetthez amit a feladó feladott, másrészt tudnunk kell azt is, hogy az egyes blokkok abban a sorrendben érkeztek e meg amilyenben feladták őket (ideértve azt is, hogy nem hiányzik-e közülük). Az első célhoz ellenőrző összegeket, a másodikhoz sorszámot célszerű a blokkban elhelyezni még a kódolás előtt.

A digitális kézjegy

arra szolgál, hogy segítségével a címzett megbizonyosodhassék egy üzenet feladójáról és bizonyíthassa, hogy az illetőtől kapott ilyen üzenetet. Olyasmire van szükség tehát mint az aláírás, ami könnyen azonosítható, de nehezen hamisítható.

A cél elérhető úgy is, hogy a kényes kommunikációt egy hitelesítő központ közbeiktatásával végezzük (mintha tanú előtt beszélnénk) ez az u.n. *nem valódi digitális kézkegy*.

A valódi digitális kézjegy

Ehhez egy nyilvános kulcsú kódolási rendszer lehet felhasználni, mégpedig olyant, mely "megfordítható", azaz E(D(x)) = x (az általunk említett módszerek ilyenek). Rejtsünk a titkos fejtő kulccsal (és fejtsünk a nyílt rejtő kulccsal). A címzett, ha a nyílt kulccsal fejthető üzenetet kap, biztos lehet abban, hogy azt csak a titkos kulcsot ismerő feladó küldhette. Mivel a címzett nem ismeri a titkos kulcsot, ha rendelkezik az üzenetnek a nyílt kulccsal megfejthető kódolt változatával, nyilvánvaló, hogy azt ő nem készíthette.

Hagyományos igények

OLTP (On Line Transaction Processing):

- az ábrázolt mini világ minden adatát tartalmazza
- az utolsó állapotot mutatja
- sok adatmódosítás
- egy-egy tranzakció kevés adatot érint
- viszonylag egyszerű, de ad hoc kérdésekre is tud válaszolni
- a válaszidő kicsi
- jellemző több, párhuzamosan működő felhasználó

Új igények

1. **Adatfolyamok feldolgozása.**

2. **Előre elkészített, aggregált adatok a vezetőknek:**
   DSS (Decision Support System).

3. **Nem ismert összefüggések kiderítése:**
   Tudásfeltárás (Data Mining, adatbányászat)

Adatfolyamok feldolgozása I.

Pl. szenzorok állandóan tömegével generál adatokat, ezeket nem tároljuk mind le, de kérdéseink lehetnek.

Forgalom figyelés: hálózati, banki, közleke stb.

Naplózás: meteorológiai, egészségügyi, ipar

Adatfolyamok feldolgozása II.

Új eszközök (részben az SQL kiterjesztésé

Stanford University:
CQL (Continuus Query Language)

Cornell University:
COUGAR

Adatfolyamok feldolgozása III.

1. Az ilyen kérdések hosszú ideig működne
2. Sokszor a megválaszoláshoz hagyomány adatbázis is kell.

Probléma: hogyan kezeljük az adatbázis adatainak változását?

(Pl. kereskedelmio forgalom vizsgálata, de árak és az ügyfelek címei hagyományos adatbázisban.)

DSS (Decision Support System)

OLAP (On Line Analitical Processing)

Megoldásai:

ROLAP
MOLAP

OLAP

- nem feltétlenül egészen up-to-date
- csak az elemzéshez szükséges adatokat tartalmazza, ezek azonban több mini világból származnak
- tartalmazza a régi adatokat (trendek)
- jellemzően olvas, de bonyolult elemzéseket végez
- a válaszidő nem kritikus
- látványos riportok, ezek könnyen elérhetőek (intra- internet, wap, sms, stb.)

ROLAP

Relational On Line Analitical Processing:

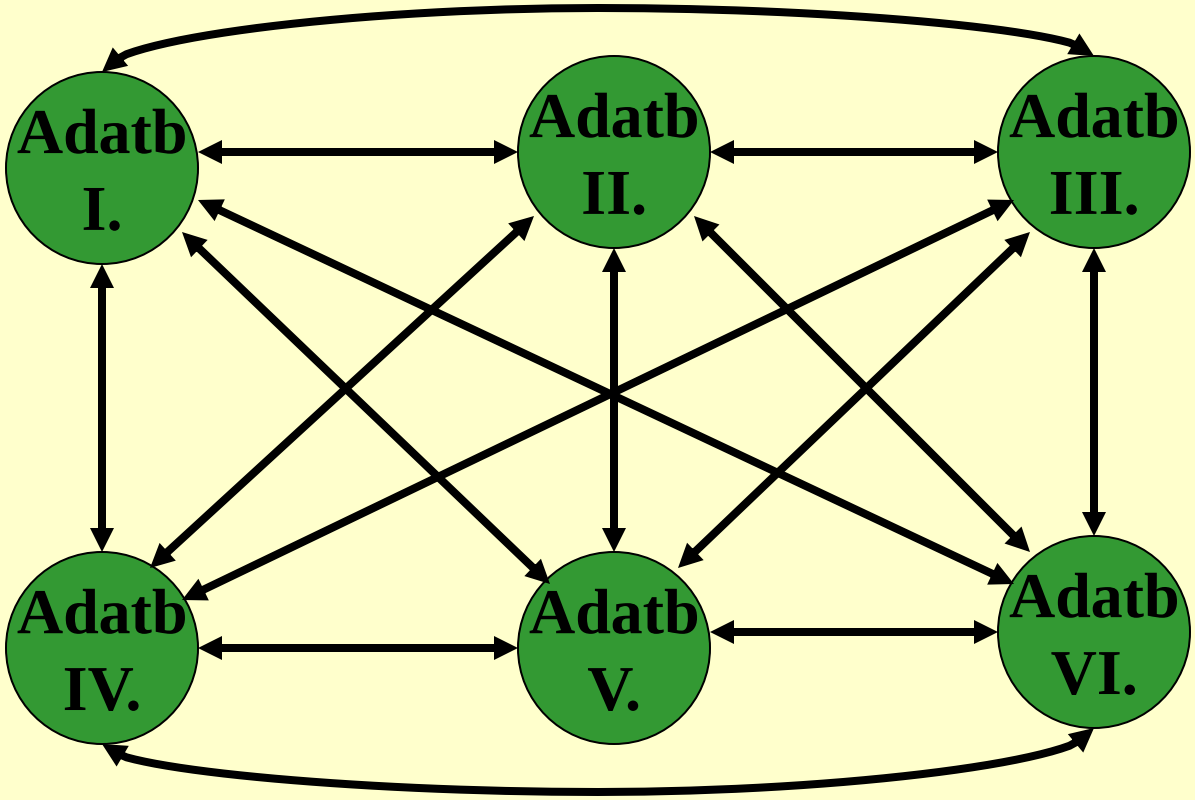
a jól ismert és bevált relációs eszközöket használja, ezek azonban nem erre a célra készültek.

MOLAP

Multidimensional On Line Analitical Processing:

Az adatokat egy többdimenziós kockában tárolja.
Könnyű megvizsgálni egy kiválasztott élnek a többitől való függését.
- lassan kiépíthető, hardware igényes rendszer
- gyorsan ad választ a várt kérdésekre.

Adatbázisok összekapcsolása

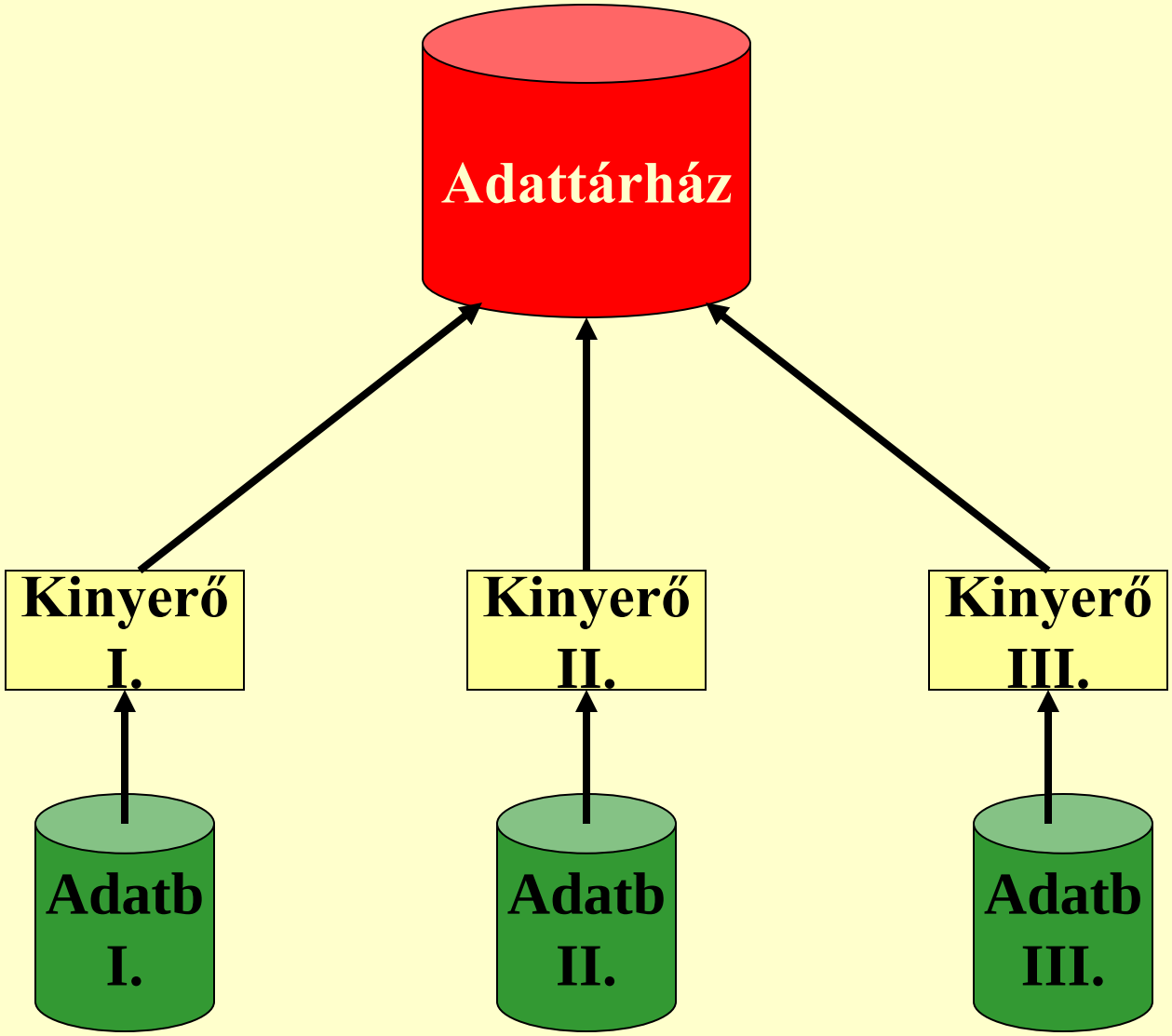


Data Warehouse: Adattárház

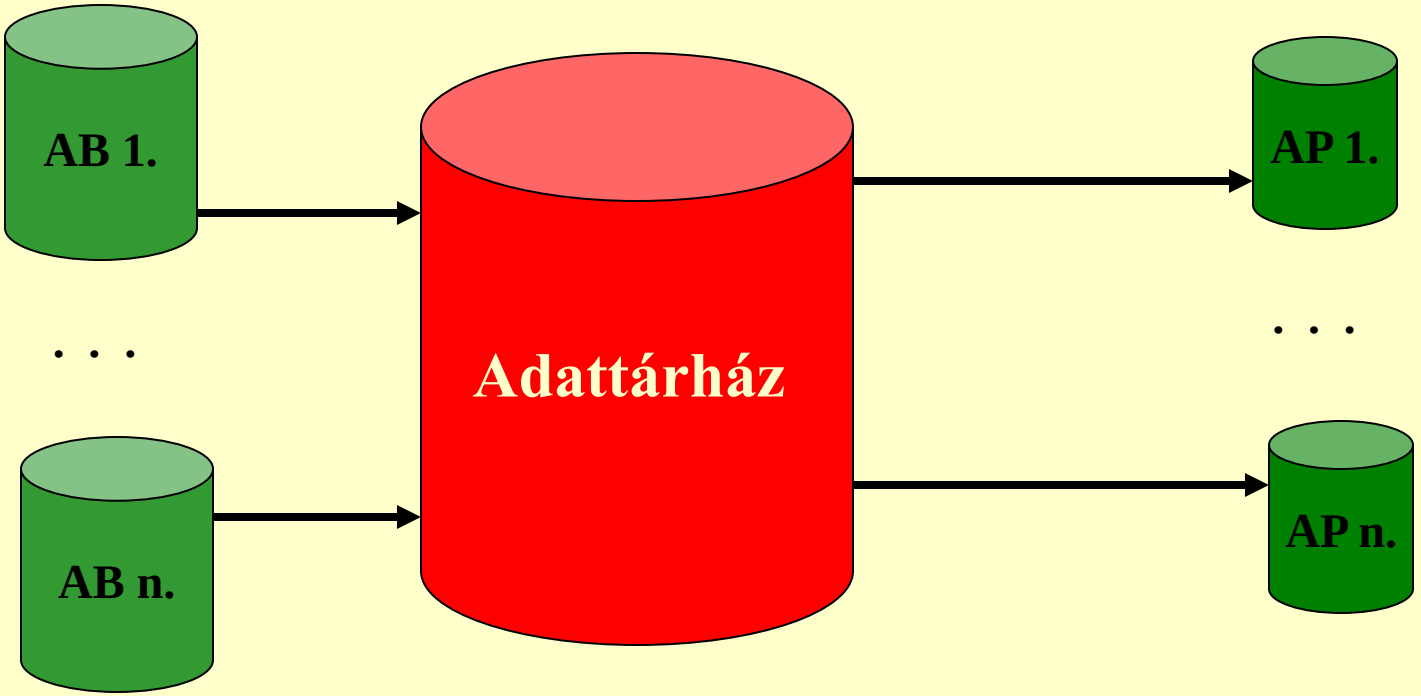
Bill Inmon:

„Az adattárház rendszer egy témaorientált, integrált, időben változó, nem átmeneti adat-rendszernek tekinthető, melynek elsődleges célja a stratégiai döntések támogatása.”

Adattárház


Adattárház
Kinyerő I.
Kinyerő II.
Kinyerő III.
Adatb I.
Adatb II.
Adatb III.

Data Marts: Adatpiacok



Operatív adatok

Adatpiacok

Adatpiacok: definíciók

1. Az adatpiac az adattárház egyik fontos komponense, egy, kiválasztott tárgyaknak osztályhoz kötött részhalmaza.

2. Az adatpiac egy alkalmazás-központú adattárház, a teljes adattárház egy része.

3. Az adattárház nem más, mint adatpiacok összessége.

Adatpiacok: előnyök

1. Adatkezelés: minden osztály maga állapíthatja meg az általa használt adatok struktúráját.
2. Relevancia: egy-egy osztály eldöntheti, hogy a historikus adatokból menynyire van szükség.
3.Önálló felhasználás: minden osztály maga döntheti el, mikor , milyen folyamatot futtat.
4. Hatékonyság: a kisebb egységek kezelése olcsóbb.

Adatpiacok: egyéb tulajdonságok

Különféle célokra sok adatpiac építhető ki egy adattárház ból, ezek akár átfedőek is lehetnek.

Egymástól függetlenül kiépült adatpiacok egy hálózatban egy virtuális adattárházat alkothatnak.

Tudásfeltárás

Definíció:
rejtett, ismeretlen, potenciálisan hasznos tudás kinyerése az adatokból nem triviális módon.

Több tudományágat átfogó kutatási terület.

Adatbányászat: az adatok összefüggéseinek feltárása.

Lépések

- **adatkiválasztás**
- **adattisztítás**
- **bővítés**
- **szűkítés**
- **kódolás**
- **adatbányászat**
- **jelentéskészítés**

Minta feladat

Egy kiadó magazinokat árul.
Kapcsolatokat keresünk az előfizetők tulajdonságai és előfizetési szokásai között.

Adatkiválasztás

Kiválasztjuk a szükséges adatokat az operatív adatbázisokból.

Pl.
ügyfélszám
név
cím
előfizetési dátum
magazin neve

Tisztítás

- **Véletlen kettőződések**
- **Név elírások (Kotsis, Kocsis, Kotsits)**
- **Kitöltés hiánya (alapértelmezés)**

Bővítés

Hozzáveszünk újabb adatokat:

- **születési idő**
- **jövedelem**
- **van-e autója**
- **van-e háza**
- **stb.**

Szűkítés

Kihagyhatunk sorokat:
pl. nincs kitöltve (?)

Kihagyhatunk oszlopokat:
pl. a név nem kell már (a tisztításhoz kellett!)

Jól meggondolni, mert elveszíthetünk fontos információkat!

Kódolás

Főként, ha az adat „túl részletes.

Pl.
Születési dátum helyett: korkategória
Cím helyett: régió kód
Előfizetés kezdete helyett: időtartam hónapokban
Jövedelem helyett: ezerrel(?) osztva
stb.

Alapvetően befolyásolhatja az eredményt!

Adatbányászat

Sokféle technika lehet, néhány ezek közül:

hagyományos lekérdező eszközök
statisztikai technikák
vizuális technikák
hasonlóság, távolság, szomszédság
döntési fák
társító szabályok
stb.

Hagyományos lekérdező eszközök

Egyszerű lekérdezések: pl. átlagszámítások.

Ezek szabhatják meg a további lépéseket!

(Pl. ha egy magazint 10 % vásárol, egy „90%-osan jó” módszer azt mondhatja, hogy ez nem kell senkinek!)

Statisztikai technikák

Összefüggéseket keresünk:
kor és vásárolt magazin
két magazint vásárlók tulajdonságai
stb.

Vigyázat! I.

Csak az olyan összefüggés nyereség, ami nem triviális:
Pl. nem meglepő, ha egy konkrét újságnál is ugyanazt az eloszlást tapasztaljuk a vásárlók életkora szerint, ami általában igaz.

Vigyázat! II.

Estleg a statisztikában mutatkozik olyan halmaz, aki semmilyen magazinra sem fizet elő. Ez adatszennyeződés, érdemes megvizsgálni, mi lehet az oka.

Vizuális technikák

Mivel nem tudjuk mit keresünk, segíthet, ha a számítógép ábrázoljuk a különféle eloszlásokat, esetleg időben változva. Érdekes összefüggéseket vehetünk észre.

Hasonlóság és távolság

A rekordokat tekinthetjük egy n dimenziós tér pontjainak. Közöttük lehet távolságot definiálni. (Legegyszerűbb az Euklidesi távolság:

$$\sqrt{(x_1-y_1)^2+\ldots(x_n-y_n)^2}$$

Érdekes csoportokat találhatunk. Kérdés: hány dimenziót vizsgáljunk? (És melyek legyenek azok?) Projekció!

Megjegyzés

Az OLAP eszközök is ilyenek, használhatóak is az adatbányászatban, de azok rögzített kérdésekre adnak választ.

Vigyázat!

A távolság numerikus értékét meghatározza a kódolás!
Pl.: a jövedelem nagyságrendje más, mint a koré: a fontossága is más lesz!

Szomszédság

A k db. legközelebbi szomszéd viselkedése adhat előrejelzést a vizsgált objektum viselkedésére.

Vigyázat!

A túl egyenletes eloszlásnál nem mond semmit, nincsenek jellegzetes osztályok.

Túl sok dimenzió esetén nem lesznek jellegzetes osztályok.

Új problémák

Nagyon sok a rendelkezésre álló adat:
a hálózat maga egy adatbázis.
Válogatás kell (felesleges másolat, elektronikus szemét).
Keresés (általában túl primitív, sok találat).
Ügynök (agent) programok (barátságos, barátságtalan).
Kiszolgáltatottság!

Vége