szept. 7.

Adatb.

EA1

Gajdos Sándor | 20093650773
gajdos @ db.bme.hu (tegez)

adat -> információ -> tudás -> HATALOM

- adat: valami értelmezhető, de nem feltétlen értelmezett
- információ: értelmezett adat
- tudás: kontextusba helyezett információ

38 -> cipőméret -> Kovács Ilona, XY felesége, itt & ott dolgozik, 38-as cipő

k	10^3
M	10^6
G	10^9
T	10^{12}
P	10^{15}
E	10^{18}
⋮	

/* vizsga: tanult tudás alkalmazása */

EA2

szept. 9. rekord: 80 byte, 1 kártya

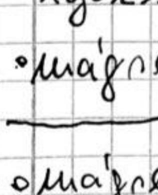
- lyukkártya - soros hozzáférés
- lyukszalag -> rekord: valami, ami egy szalagra ráfér
- mágnesszalag - még mindig soros

---

klasszikus adatbázisok ősainak eleje

- mágnesdob - direkt hozzáférés

↓



adatbázis szimbólum

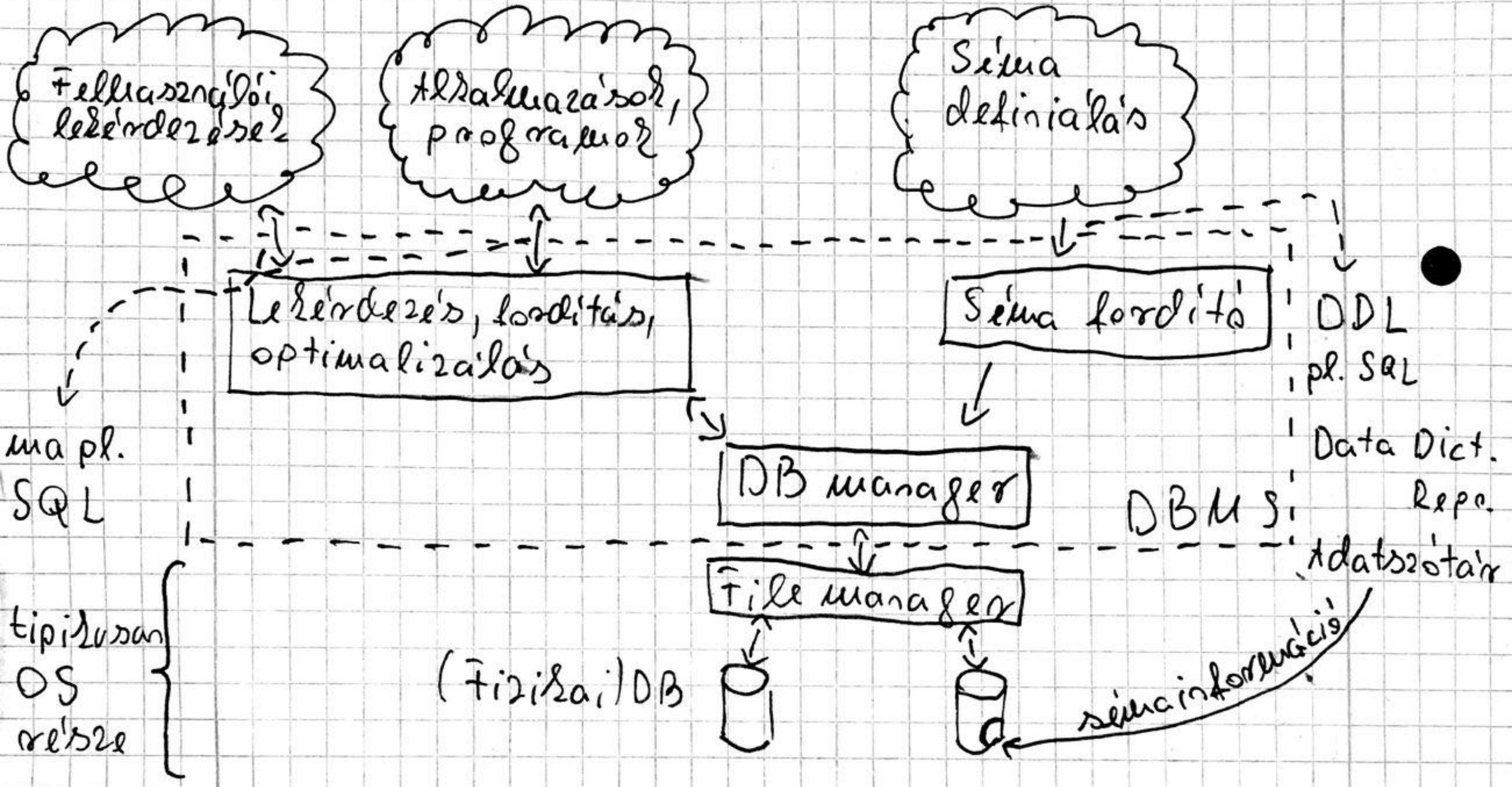
közvetlen hozzáférésű eszköz. hogy kezeljük az adatokat?

Adatbáziskezelő (DBMS):

hardver - szoftver rendszer adatok magas szintű kezelése

- gazdag adatstruktúra (meta-adatok, etc.)
- hosszú életciklus (visszafelé kompatibilis)
- nagy adatmennyiség

(igen, ez kibaszott pongyola)



Séma: adatszerkezet leírás - milyen adatok, milyen szerkezetben

Strukturált adat:

a sémát leíró adatok mennyisége jóval kisebb, mint maga az adat.

Szemi-strukturált:

a sémainformáció mérete összemérhető magával az adattal (XML, JSON)

Meta-adat: adat az adatról, pl. sémainfó

DB manager: - integritás (típus-kényszer)
- adatbiztonság (mirroring, etc.)
- szinkronitás (többszörös hozzáférés)

File manager: a rajz egy tipikus konfigurációt mutat, ilyenkor ez már az OS dolga, így a DBMS ezt csak használja.
megesik, hogy ez nem elég jó és megoldja a DBMS magának az egészet, közvetlen firkálva a merevlemezt

primitív 3 réteg:

3. logikai adatfüggetlenség $\boxed{\text{Nézet 1}}$ $\boxed{\text{Nézet 2}}$ ... $\boxed{\text{Nézet N}}$ szűrések

2. fizikai adatfüggetlenség $\boxed{\text{logikai vagy fogalmi}}$ táblázat egészben

1. $\boxed{\text{Fizikai réteg}}$ cilinder, fej, etc.

egy alsó réteg cseréje ne változtasson semmin

EA 3

szept. 16

adatbázis adminisztrátor:
- fizikai cuccok
- adatbiztonság
- jogosultságok

nyelvek:
- adatlekérdező
- sémaleíró
- gazda(g) (házi módra)

DML
data manipulation language
pl. SQL

DDL

adatszerkezeteket formálisan
leíró nyelv, sémák
pl. SQL

gazda

mindenféle dologra, ami túl
komplex az eddigieknek

C kód → gazda
SQL kód → beágyazott
C kód

pc $\xrightarrow{\text{előfordítás}}$ c

Logikai adatszerkezetek létrehozása & tervezése

↓

adatmodellezés

$\underbrace{\text{adatmodellek}}_{}$ létrehozása

formális jelölés-
rendszer adatok
& kapcsolataik
leírása, ill.
műveletek halmaza

egy adatdiagram-ot fogunk rajzolni,
ami formális módon, a szemantikától
elvonatkoztatva gyakorlatilag egy adatmodell

ER - entity relationship model

modell adatokról, de nem adatmodell
adatok & kapcsolataik leírására

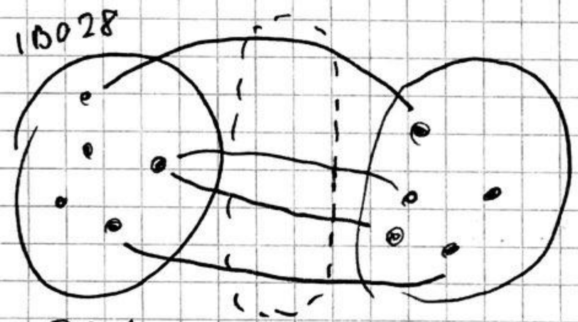
ER részei: • egyedhalmazok / entity-set
• tulajdonsághalmazok / attribute-set
• kapcsolathalmazok / relationship-set

bármi lehet, aminek saját léte van. megkülönböztethető, "egyed";

valahány darab egyed egymáshoz rendelése

egy egyedhez kapcsolódva jellemez / egyedivé teszi

ezekből halmaz úgy lesz, hogy valamilyen relevancia alapján csoportosítom őket
a halmaz elemeit a szemantika kapcsolja össze



Hogy lehet ezt formalizálni?

egyedhalmaz:

ÉPÜLET NAPA

TERMÉK(ID, ~~[illegible]~~, ~~[illegible]~~)

halmaz neve

tulajdonság-halmazok

sok formalizmus

TÁBLÁK(ID, SZÍN, PEZ)

kapcsolathalmaz:

OTT_VAN: TERMEK, TABLAK

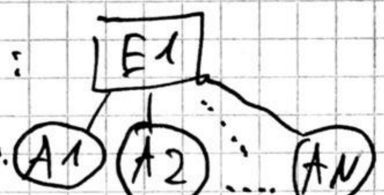
ez így tök jó, de a gyakorlatban használhatatlan

ER-diagram

ER modell grafikus ábrázolása

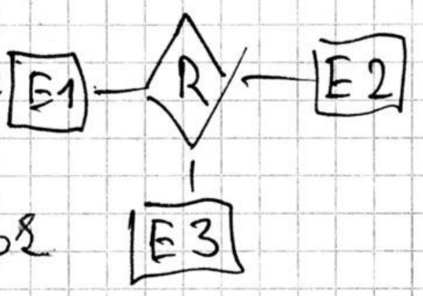
egyedhalmaz:

tulajdonsághalmaz:



kapcsolathalmaz:

aritás: egyedhalmazok száma



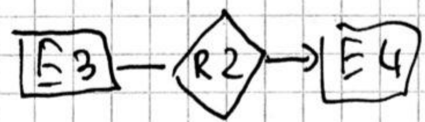
Kapcsolatok funkcionalitása:

Bináris kapcsolathalmaz esetén

1:1

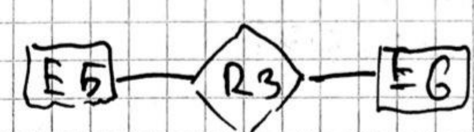
FŐNÖK: OSZTÁLY, EMBER

N:1



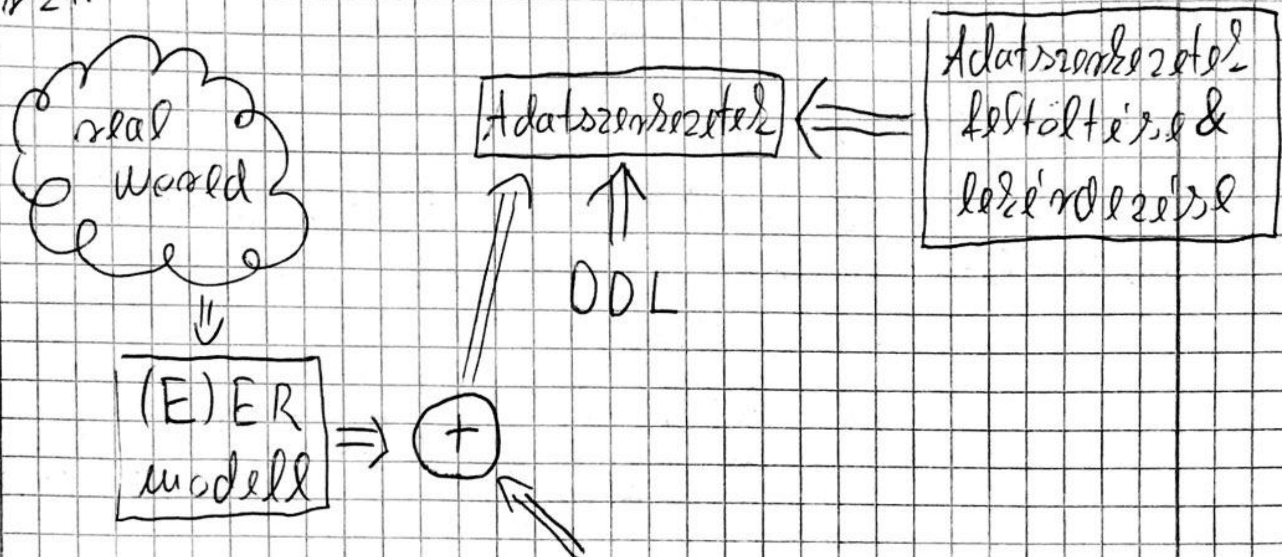
DOLGOZÓ: OSZTÁLY, EMBER

N:M



TAN: DIÁK, TANÁR

szeptember 21.

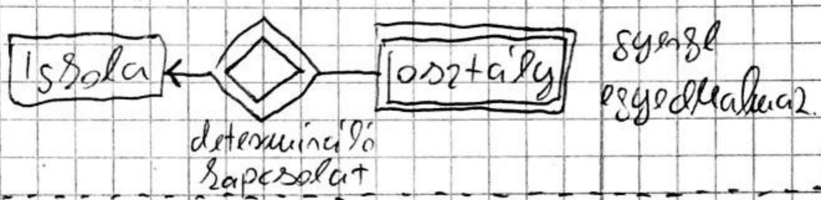


Adatmodellek:

- hierarchikus
- hálós
- relációs
- objektum orientált
- deduktív
- fuzzy

Egyedhalmazok & determináló kapcsolatok

lehetséges olyan egyedhalmazok megadása, ahol a példányok egyediségét egy másik egyed biztosítja

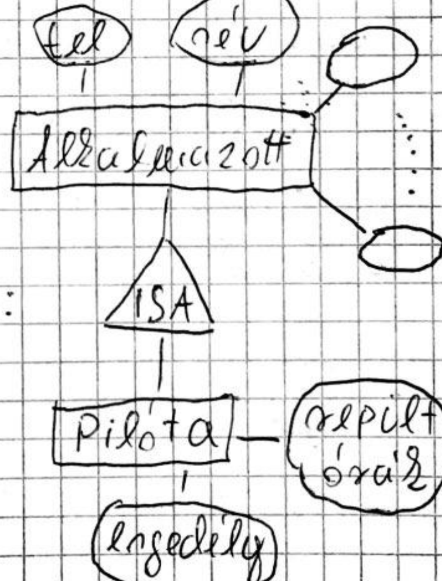


Kulcs

olyan halmaza a tulajdonsághalmazoknak, melyek segítségével az egyedhalmaz példányai egyedivé tehetők.

ISA kapcsolatok

specializáció, egy kép többet mond ezer szónál:



Relációs adatmodell

- Hogyan lehet az adatokat leírni?

A relációs adatmodellben a kártéz relációs elmélete: halmazok desc. szorzatainak részhalmaza.

pl.: $R = \{1, 2, 3\}$, $S = \{a, b\}$ $\Big\}$ $R \times S = \{(1,a); (2,a); (3,a); (1,b); (2,b); (3,b)\}$

R & S legyen inkább r & s, de tollat nem bírok radírozni.

$r_1 \triangleq \{(2,a); \underbrace{(3,b)}_{\text{reláció elemei}}\}$

$r_2 \triangleq \{(1, \underbrace{b}_{\text{komponensek}})\}$

$r_3 \triangleq \{\}$

na most ezt a matematikusok imádják, de az informatikusok a szemléletesebb dolgokat szeretik:

r_1

A	B
2	a
3	b

attribútumok nevei

reláció elemei

reláció fokszáma: 2 (oszlopok száma)

attribútumok

A attribútum értelmezési tartománya r (hívják még domain-nek is)

r_2

A	B
1	b

reláció elemszáma: 1 (sorok száma)

$|r_2| = 1$

r_3

A	B

relációs séma

$R_3(A, B)$

relációs séma, neve nagy

az attribútum-oszlopok sorrendje lényegtelen, amennyiben a neveik egyediek.

- Műveletek

Ha már eltettünk egy rakat adatot, valahogy ki is kell szedni:

DML alapjai

vegyük észre, hogy ez az egész cucc egy halmaz!
sorok: érték N-esek (tuple)

valami olyan műveletek kellenek, amik relációkból relációkat gyártanak.

- halmazalgebrai műveletek: $(\cup, \setminus)$ [a metszet ezekből már kifejezhető]

probléma: különböző fokszámú relációk

↳ műveletek CSAK azonos fokszámú relációkra működjenek

probléma: relációk desc. szorzata?

↳ X azért mégis tehetem, tetszőleges fokszámú relációk között is!

- projekció (vetítés): $\Pi_{A_{i1}}(r(R))$ vagy $\Pi_{A_{i1}, A_{i2}}(r(R))$
  r: reláció
  R: r sémája, pl. $R(A_1, A_2, \ldots, A_n)$
  oszlopok közötti válogatás,
  ahol $i_1, i_2 \in [1, \ldots, n]$

- szelekció (kiválasztás): $\sigma_F \quad (r(R))$ F: 0-ad rendű formula
  r: reláció, R: r sémája ↳ attribútum-nevek+ik, konstansok, $<, >, =, \neq, \wedge, \vee, \neg$
  sorok közötti válogatás
  F kiértékelésével megkapom, hogy a sor kell-e vagy nem

Szeptember 28.

Ezek voltak eddig az alapműveletek.
ezekből kifejezhetőek:
<u>származtatott műveletek</u>

- természetes illesztés (natural join)

$R(A_1, A_2, \dots, A_n)$ $S(B_1, B_2, \dots, B_m)$

A_i-k és B_j-k között páronként van néhány azonos nevű

legyen: A_{i1} azonos nevű B_{j1}-el (mondjuk "név")

$\vdots$ $\vdots$

A_{ik} B_{jk}

Háttér: van valami közös szemantikájuk

$$\underbrace{r(R) \bowtie s(S) \triangleq \Pi_{R \cup S}\, \sigma_{A_{i1}=B_{j1} \wedge \dots \wedge A_{ik}=B_{jk}}(r \times s)}_{}$$

ez kb azt jelenti, hogy az azonos nevű attribútum-párosoknak az értékei is legyenek ugyanazok
+ 1 projekció az elejére

Kérdezzük meg: ez mi a másra jó?

pl: $r(R)$

A	B
a	b
a	c
b	b

$s(S)$

A	C
a	c
b	c

$r \times s$

R.A	B	S.A	C	
a	b	a	c	✓
a	b	b	c	–
a	c	a	c	✓
a	c	b	c	–
b	b	a	c	–
b	b	b	c	✓

$$r \bowtie s = \Pi_{A,B,C}\left[\sigma_{R.A = S.A}(r \times s)\right] \Rightarrow$$

$$\sigma_{R.A = S.A}(r \times s)$$

R.A	B	S.A	C
a	b	a	c
a	c	a	c
b	b	b	c

$\Downarrow$

$r \bowtie s$

A	B	C
a	b	c
a	c	c
b	b	c

Ha még mindig nem világos:

__pl2.:__ DOLGOZO(O_ID, D_NEV, ...)
OSZTALY(O_ID, TITKARNO, ...)

xy melyik osztályon dolgozik?

$\pi_{O\_ID}\,\sigma_{D\text{-}NEV='xy'}(DOLGOZO) \Rightarrow DOLG \bowtie OSZT$

O-ID	D-NEV	TIT...
⋮	⋮	⋮

- Theta - join

$$r(R) \underset{i \Theta j}{\bowtie} s(S) \triangleq \sigma_{R.A_i \Theta S.B_j}(r \times s)$$

__pl.:__ $i = 2$
$j = 1$
$\Theta = \leq$

R.A	B	S.A	C
a	b	b	c
⋮	⋮	⋮	⋮

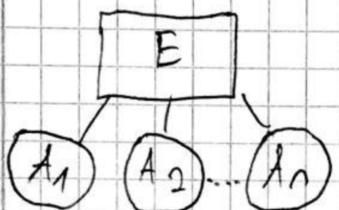
Definíció:

__Relációs teljesség:__ olyan relációs lekérdező nyelv, ami tudja a fenti 5 műveletet az relációsan teljes

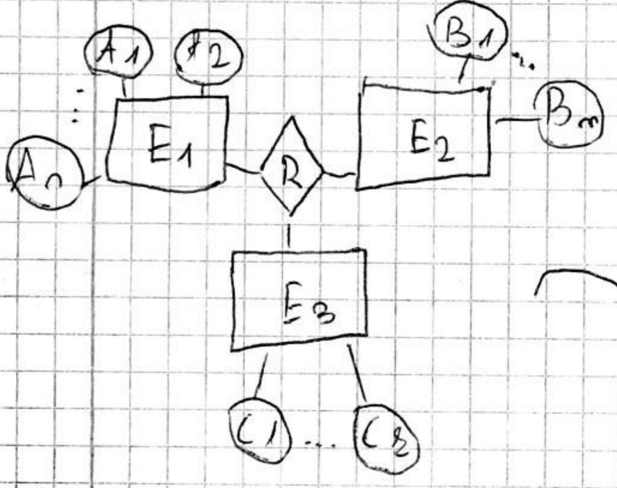
(ez az alap. az SQL persze tudja ezt, de ennél sokkal többet) probléma:

Hogyan lesz egy ER-modellből egy konkrét adatmodell? (relációs séma)

ER-diagramok transzformációja relációs adatszerkezetekre



$$E(A_1, A_2, \dots, A_n)$$



eddig:

$$R(\underbrace{A_1, A_2, \dots, A_n}_{E_1 \text{ elemét határozza meg}}, \underbrace{B_1, \dots, B_m}_{E_2\ -\|-}, \underbrace{C_1, \dots C_k}_{E_3\ -\|-})$$

jobb ennél:

legyen E_1 kulcsa $\underline{A_{i1}}$

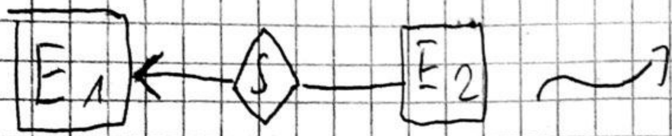
legyen E_2 kulcsa $\underline{B_{j1}}$

legyen E_3 kulcsa $\underline{C_{k1}}$

$$R'(\underline{A_{i1}}, \underline{B_{j1}}, \underline{C_{k1}})$$

idegen kulcs (foreign key)

referenciális integritási kényszer



eddig:

$E_1(A_1, \dots, A_n)$

$E_2(A_1, \dots, B_m)$

$S(\underline{A_1}, \underline{B_1})$

ez nem jó, mert nem 1..N, hanem N..N: ER-hez képest szemantikát változtat

jobb:

$E_1(A_1, \dots, A_n)$

$E_2(B_1, \dots, B_m, \underline{A_1})$

Szeptember 30.

egy összetettebb példa:

ARU(ARUKOD, ARUNEV, E_AR)

BEVETEL(DATUM, ÖSSZEG)

MENNY(DATUM, ARUKOD, DB)

BEFIZ(OSSZEG, BEF)

- - - - - - - - - - - -

2016.09 utáni napok bevételei:

$$\sigma_{DATUM>201609} BEVETEL$$

2016.09.29. bevétele és a bankba fizetett összeg:

$$\pi_{OSSZEG, BEF}\, \sigma_{DATUM=20160929} (BEVETEL \bowtie BEFIZ)$$

2016.09.29 termékkód alapján termék neve & eladott db.

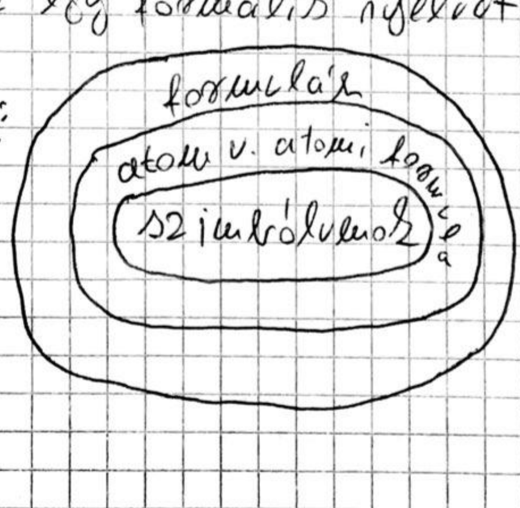
$$\pi_{ARUNEV, DB}\, \sigma_{DATUM=20160929 \wedge ARUKOD=XY} (ARU \bowtie MENNY)$$

Alternatíva: $\boxed{\text{sor \& oszlopkalkulus}}$

logikai jellegű leírás

definiálunk egy formális nyelvet (elsőrendű)

Felépítése:



→ sorkalkulus kiegészítés

↓

relációk leírására

Megengedett szimbólumok:

- ( )
- $> < \geq \leq = \neq$
- $\vee \wedge \neg$
- $s^{(n)}$ n komponensű sorváltozó
- $s^{(n)}[i]$ ——— " ——— i. komponense
- $R^{(n)}$ relációs konstans
- c konstans
- $\forall \exists$

Elemi formulák:

- $R^{(n)}(s^{(n)})$ n-es hozzátartozik
- $s^{(n)}[i] \Theta u^{(n)}[i]$ Θ aritmetikai összehasonlítás
- $s^{(n)} \Theta c$
- $R^{(n)}(c_1, c_2, \ldots, c_n)$

Hogyan értelmezzük?

- minden atomi formula is formula
- Ha ψ_1 formula és ψ_2 is az, akkor $\psi_1 \vee \psi_2$, $\psi_1 \wedge \psi_2$, $\neg \psi_i$ is formulák
- Ha ψ formula, akkor $\exists t^{(n)} \psi(t^{(n)})$, $\forall t^{(n)} \psi(t^{(n)})$ is formulák

adott sorkalkulus formuláiból ($\psi(t)$)

$\{ t^{(n)} \mid \psi(t^{(n)}) \}$ sorkalkulus kifejezés

($t^{(n)}$-kből az, amire ψ igaz) (ψ-nek t egyetlen sorváltozója)

A fenti formalizmus egy lehetséges értelmezése:

„A" legyen az összes, számítógépen ábrázolható szám halmaza

TFH $s^{(n)} \in A^n$, $R^{(n)} \subseteq A^n$, $c \in A$ – logikai értékek hozzárendelése

- $R^{(n)}(s^{(n)})$ pontosan akkor igaz, ha $s^{(n)}$ adott helyettesítési értéke $\in R^{(n)}$
- $s^{(n)}[i] \odot v^{(n)}[j]$ pontosan akkor igaz, ha $\odot$ teljesül
- $s^{(n)}[i] \odot c$ is
- $R^{(n)}(c_1, c_2, \dots, c_n)$ pontosan akkor igaz, ha $(c_1, c_2, \dots, c_n) \in R^{(n)}$
- $\Psi_1 \vee \Psi_2$ pontosan akkor, ha közülük legalább egy igaz
- $\Psi_1 \wedge \Psi_2$ ... (ezt le nem írom)
- $\neg \Psi_1$ ...
- $\exists t^{(n)} \Psi(t^{(n)})$ ha van olyan helyettesítés
- $\forall t^{(n)} \Psi(t^{(n)})$ ha minden helyettesítés olyan

Fordítsuk le az óra elejei példát először a hieroglifákra!

a.) $\{ t^{(2)} \mid BEVETEL(t^{(2)}) \wedge t^{(2)}[1] > 20160901 \}$

b.) $\{ t^{(2)} \mid BEFIZ^{(2)}(t^{(2)}) \wedge (\exists v^{(2)}) BEVETEL^{(2)}(v^{(2)}) \wedge v^{(2)}[2] = t^{2}[1] \wedge v^{(2)}[1] = 20160929 \}$

október 5.,

előző sémák:

ARU(ARUKOD, ARUNEV, E_AR)
BEVETEL(DATUM, OSSZEG)
MENNY(DATUM, ARUKOD, DB)
BEFIZ(OSSZEG, BEFIZ)

---

relációs algebra: imperatív jellegű
oszlop & sorkalkulus: deklaratív jellegű

sorkalkulus & relációalgebra kifejezőereje:

∀ relációalgebrai kifejezéshez lehet találni olyan sorkalkulus kifejezést, amely pontosan ugyanazt a relációt állítja elő, mint a relációsalgebrai kifejezés, úgy, hogy legfeljebb azokat a konstans relációkat használja fel, melyek a relációalgebrai kifejezésben is benne voltak

Megfordítva ez már nem igaz:

$\{t^{(n)} \mid \neg R^{(n)}(t^{(n)})\}$: elvből simán kijöhet ∞ sok reláció, ezt relációalgebrával nem tudom megcsinálni

ennek annyira nem örülünk, mert simán gyárthatunk mérnöki szempontból használhatatlan kifejezéseket.

ezeket találták ki

Biztonságos kalkulus

Formula doménje: $DOM(\varphi) = \{\varphi$-ben található összes konstans és a konstans relációk komponensei$\}$

(köznyelven: a táblázatok $\forall$ cellájának halmaza)

def:

a.) $\forall$ φ-t kielégítő t $\forall$ komponense $DOM(\varphi)$-beli legyen

b.) $(\exists u)\, \omega(u)$ esetén u $\forall$ helyettesítési értéke, amely ω-t igazzá teszi, $DOM(\omega)$-beli legyen

magyarázóként:

$\varphi(t) = \underbrace{R(t^{(n)})} \wedge \ldots$

ez akkor igaz, ha a sok cucc, amit $t^{(n)}$-be teszek, a táblázatból valók (a relációban vannak). tehát ha olyan n-eseket teszek bele, amik a táblázat részei, a kifejezés biztonságos.

még egyszerűbben: nem tudok kitalálni olyan dolgokat, amiket nem tárolok, mert olyanok, mint a nyulak: sokan vannak. amit tárolok, azt meg elő tudom szedni.

kérdés?
nem öntöttük-e ki a fürdővízzel együtt a gyereket is? (!! Gajdos 2016.10. csü.)

a válasz az, hogy nem:

a biztonságos sorkalkulus kifejezőereje a relációalgebrával azonos (pontosan akkora)

és ez nekünk jó.

hiánypótlás:

<u>Oszlopkalkulus:</u>

sorkalkulus: n változóból álló sorváltozók
oszlopkalkulus: a sorokat változókra szétbontjuk:

$$t^{(n)} \longmapsto x_1, x_2, \ldots, x_n$$

ezt leszámítva tök ugyanaz.
arra jó, hogy egy rakat dolgot szebben/kompaktabban le lehet vele írni.

új fejezet, ami elvileg para:

Normalizálás

olyan információs rendszert kéne tervezni, ami kiaja ha egyszerre jó sokan piszkálgják.

fő probléma: redundancia. sok felesleges adat.
akkor lesz valami adatelem redundáns, ha valamilyen más adatelemekből össze lehet rakni.

def.: reláció redundanciája
egy reláció akkor redundáns, ha a relációban található valamely attribútum értékét ki tudjuk következtetni a relációban lévő más attribútumok segítségével.

pl.: ha tárolok születési időt és kort is

OLTP: online transaction processing
ilyen környezetben jelentkeznek az **anomáliák**:

- módosítási
- beszúrási
- törlési

<OFF>
- mitől ipari egy megoldás?
- attól, hogy működik
</OFF>

(pls ne gányoljunk, ne tároljunk redundáns adatokat köszipuszi.)

Megoldás:
séma dekompozíció.
szétszedjük a szemetünket több táblába

az univerzális reláció (egy baszott nagy tábla) kényelmes, de szar (tele lesz redundanciával).

De mi alapján akarunk dekomponálni?

ezt csak $\underbrace{\text{adat-szemantika}}_{}$ alapján lehet felbontani

függőségi viszonyok,
constraint-ek

Kényszerek:

- értékfüggő: pl. cipőméret $15 < cm < 60$
- értékfüggetlen:
  - tartalmazási függés:

    pl.: alkalmazott (O_ID, ...)
    osztály (O_ID, ...)

    ha jó az adatbázis, az alkalmazott.O_ID csak olyan lehet, ami benne van valahol az osztály táblában (osztály.O_ID)
  - funkcionális függés:

    pl. személyi szám -> név

~~többértékű függőség~~

  - többértékű függőség

def.:

R séma, $A \in R$, $B \in R$ attribútumok

B funkcionálisan függ A-tól, ha a reláció $\forall$ elemében A értékei egyértelműen meghatározzák meg B értékeit. Formálisan: $A \rightarrow B$

ógörögül:

$$\boxed{\forall r : \forall t, t' \in r(R) : t[A] = t'[A] \text{ esetén } t[B] = t'[B]}$$

érdemi funkcionális függés ("$\forall r$" nélkül eseti)

érdemi: adatstruktúrák tervezésére
eseti: adatstruktúrák elemzésére

október 12.

implikáció:

P	Q	P -> Q
0	0	1
0	1	1
1	0	0
1	1	1

P: tél van

Q: esik a hó

Funkcionális függőségek:

valamilyen függvény alapján függ egymástól a két attribútum

$$\underbrace{\forall r(R)\ \forall t, t' \in r(R)\quad t[A] = t'[A] \Rightarrow t[B] = t'[B]}_{A \to B}$$

(nyilván ez itt egyes attribútumokra van felírva, de nem kell túl nagy fantázia ahhoz, hogy általánosítsuk halmazokra)

unalmas rész:

egy halom definíció felírása

jelölésrendszer: $r(R)$ — r: reláció, R: relációs séma

Ω univerzum

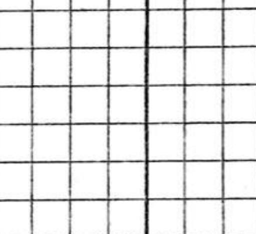
$X, Y \subseteq \Omega$ attribútumhalmazok

$A, B \in \Omega$

def: Determináns:

$X, Y \quad X \to Y \quad \nexists X' \subset X \quad X' \to Y$

„X az Y determinánsa"



def: ilyenkor Y az X-től teljesen függ

def: részleges függés: $\exists X' \subset X \quad X' \to Y$

def.: <u>kulcs</u>

X kulcs az univerzumban ha:

$X \rightarrow \Omega \quad \nexists X' \subset X \quad X' \rightarrow \Omega$

(minimális kulcs)

def.: <u>szuperkulcs</u>

$X \rightarrow \Omega$ (ha nem tudjuk a minimalitást megmutatni)

def.: <u>X és Z kulcsok:</u>

egyikük: elsődleges (primary) kulcs

többi: kulcsjelölt (candidate)

def.: $R, R' \quad \Omega, \Omega'$

$X \subseteq \Omega \cap \Omega'$ (közös rész)

X kulcs Ω-ban

X neve Ω' kontextusában idegen (foreign) kulcs

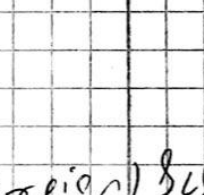
def.: <u>egyszerű kulcs</u>

ha egy kulcs egy attribútumból áll

(simple key)

ha nem: def.: <u>összetett kulcs</u> (composite)

def.: <u>elsődleges attribútum</u>



ha K_i kulcsok, akkor $\cup K_i$ elemei

elsődleges attribútumok, $\Omega \setminus \cup K_i$ másodlagosak

tétel:

∀ relációs sémának ∃ legalább egy kulcsa

biz:

addig hagyjuk el a nem kulcs attribútumokat, míg nem marad más

def.:
normálformák: relációs sémákra tett állítások, melyek segítenek csökkenteni a redundanciát.

1NF: egy relációs séma 1NF, ha ∀ attribútum értéke $\underbrace{\text{atomi}}_{}$ (1NF(R))

oszthatatlan: az adott attribútum értékére vonatkozóan: nem próbáljuk részekre vágni, csak egyben használni

ha ez nem teljesül, akkor 0NF, vagy nem normalizált a séma.

1NF: normalizált

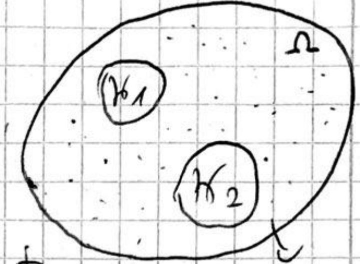
klasszikus példa: dátumok: egy attribútum → nem normalizált
: 3 külön attribútum → normalizált

ha egyben tárolom, akkor ha a nap kell nekem, akkor ki kell bányászni: nem atomi.

ezen a szinten nem csökken a redundancia, ez csak egy közös platform a többinek:

2NF: 1NF és ∀ másodlagos attribútum a séma bármely kulcsától teljesen függ

K_1-nek & K_2-nek nem lehet olyan részhalmaza ami egyértelműen meghatároz egy attribútumot



K_1, K_2 kulcsok

másodlagos attribútumok

na ez már segít csökkenteni a redundanciát

$\neg$2NF(R): R(A, B, C, D)

ez egy valós példában az adatszemantikában található ki

$F = \{AB \to C,\ B \to D\}$

funkcionális függőségek

AB: séma kulcsa, de egy részhalmaza (B) meghatározza D-t

A, B: elsődleges kulcs

C, D: másodlagos attribútum

részben ez miért rossz?

A	B	C	D
a_1	$\boxed{b_1}$	c_1	d_1
a_1	$\boxed{b_2}$	c_2	d_2
a_2	$\boxed{b_1}$	c_3	$\boxed{d_1}$

$B \to D$

ez kötött

mi ez? <u>redundancia</u>.

és ha ez sokszor van benne az adatbázisban, és nevet változtat mondjuk a polár akkor sokszor kell magát D-t átírni. ez lassú és el lehet rontani.

$B \to D$ mantra:

van olyan 2 sor a relációban, ami B-n megegyezik? mert ha igen, akkor azoknak D-n is meg kell egyezni.

ha ezeket nem tartom be, akkor sértem az adatok szemantikáját. pl. ha B a személyi szám, D meg a név, és nem tartom be $B \to D$-t, akkor pláne faszságot írok

Mit tanultunk ebből?

Ha nem 2NF a séma, akkor lehet, hogy tele lesz az adatbázisunk redundanciával, ami nem jó.

Észrevételek:

- Ha ∀ kulcs egyszerű, akkor 2NF automatikusan teljesül.

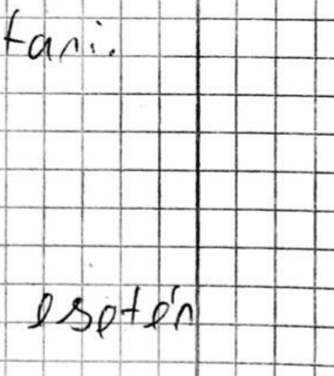
- Séma lehet olyan séma, hogy ∀ attribútum elsődleges kulcs. Ilyenkor ez biztos 2NF, definíció alapján.

- Ha valami nem 2NF, így annyi redundancia lehet benne amennyit csak nem szégyellsz. De ha elfogadod Bajdost uradnak és megváltódnak, akkor ezt szét tudod szedni több táblára, amik már 2NF-ek lesznek, ezekből pedig az eredeti tartalmat ∀ vissza lehet állítani.

def.: Tranzitív függés:

$X \to Y$, $Y \not\to X$, $Y \to A$, $A \notin Y$ esetén

A az X-től tranzitívan függ.



„az Y-on keresztül határozza meg az X az A-t" na ez arra jó, hogy

def: <u>3NF</u>

egy séma 3NF (3NF(R)), ha 1NF és másodlagos attribútum kulcstól tranzitívan nem függ.

lehet sejteni, hogy valamilyen sumákságon keresztül, ha valami $\neg$3NF, akkor ott megint redundanciák lesznek.

pl: $\neg$3NF: $R(A, B, C)$
$F = \{A \to B, B \to C\}$
} explicit $\neg$3NF

ezt nem könnyű ellenőrizni...
(mint: nagyon nem polinomidejű)
↓ ezért ekvivalens def.

3NF(R) 1NF
$\forall\ X \to A$ $A \notin X$ -re X szuperkulcs vagy
A elsődleges attribútum

ez nagyon ~~vagy~~ vad, el kell hinni, valami okos pofa belátta.

október 14

ismétlés:

3NF: 1NF és minden K kulcs és A másodl attribútum esetén $\nexists y$ $K \to y$, $y \not\to K$, $y \to A$, $A \notin y$

vagy

1NF és minden $X \to A$ $A \notin X$ esetén X szuperkulcs vagy A elsődleges attribútum

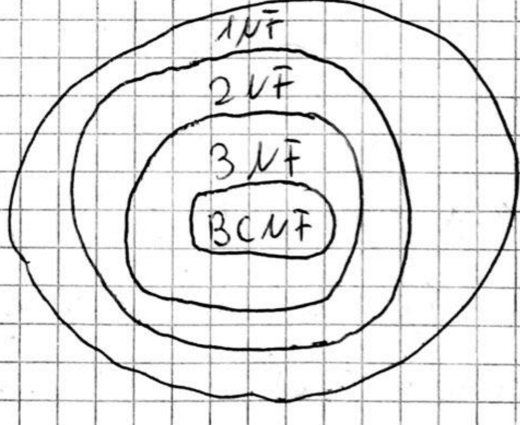
ezután még mindig lehet redundanciánk...
Hol van ennek vége?

BCNF: 1NF és $\forall$ K kulcs és A attribútum esetén $\nexists y$ $K \to y$, $y \not\to K$, $y \to A$, $A \notin y$

vagy

1NF és $\forall$ $X \to A$ $A \notin X$ esetén X szuperkulcs

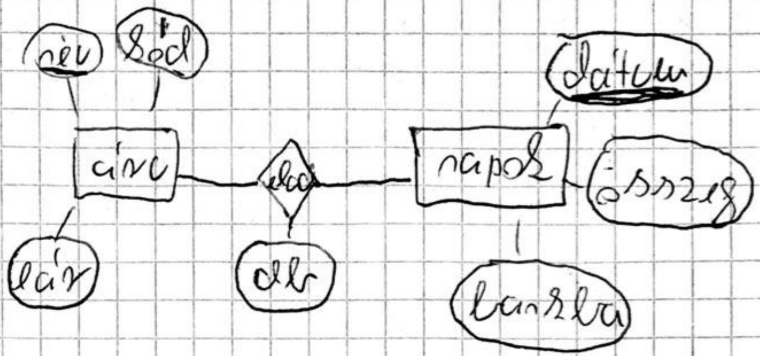
vizuális típusoknak:



tehát konyhanyelven, ha a függéseink bal oldala $\forall$ szuperkulcs, akkor nem lesz funkcionális függés miatti redundancia a sémában.

Hogy lehet ezt a gyakorlatban használni?

árukód
árunév
eár
dátum
összeg
darabszám
kassza



összeg - 4000 = kassza

ebből most simán lehet relációs sémákat gyártani.
most inkább funkcionális függésekkel fogunk foglalkozni, hogy célszerű legyen

F = { összeg → kassza;
árukód → árunév, eár;
kassza → összeg;
árukód, dátum → darabszám;
dátum → összeg }

a dátum és az árukód együtt mindent meghatároz, de csak együtt, így együtt ők a kulcs

ez egy másik modell ugyanarról a dologról.
ugyanúgy lehetnek hiányosságai, ez egy másik szemüveggel nézve ilyen

univerzális séma: napi_helyzet(....)
kulcs: (dátum, árukód)
elsődleges: dátum, árukód
másodlagos: egységár, összeg, db, kassza, árunév

ez így hemzseg olyanoktól, hogy a kulcsnak részhalmaza meghatároz valamit ez így nagyon nem 2NF... szóval tele lesz redundanciával

próbáljuk újra, a függőségek alapján:

áru(<u>árukód</u>, árunév, egységár) -> BCNF

bevét(<u>dátum</u>, összeg, bankba) -> 2NF

elad(<u>dátum</u>, <u>árukód</u>, db) -> BCNF

2 tök jó lett, de a bevét azért annyira nem. mit teszünk ezzel? bontsuk tovább:

bev(<u>dátum</u>, összeg) -> BCNF

bank(<u>dátum</u>, bankba) -> BCNF

megjegyzés: ∀ 2 attribútumos séma BCNF

Voltak eseti, meg érdemi funkcionális függések

eseti ⊇ érdemi

ismert függések | ismeretlen függések

$\underbrace{\text{ismert függések} \quad \text{ismeretlen függések}}_{F^+}$

$F \rightarrow \boxed{BB} \rightarrow$ új F^+

fekete doboz, függések alapján újakat ad

elvárások:
- igaz legyen a kimenet
- állítsa elő az összeset
- egyszerű legyen, kevés szabály

ezt egy Armstrong nevű csávó találta ki

Armstrong axiómái:
(igazakból következtetési szabályok)

mikor helyes egy funkcionális függőség?
<u>def.:</u>
F: FF-ek halmaza

$X \to Y$

helyes, ha $\forall$ olyan reláción ($r(R)$) teljesül, amelyben F valamennyi függősége fennáll.
jelölés: $F \models X \to Y$

<u>def.:</u>
funkcionális függés előállítása/levezetése
$F \vdash X \to Y$

<u>tétel:</u> (igazságtétel)
az Armstrong axiómák igazak.
~~...~~ $F \vdash X \to Y \Rightarrow F \models X \to Y$
$\forall$ amit előállítok, helyes.

<u>tétel:</u> (teljességtétel)
az Armstrong axiómák teljesek.

$F \models X \to Y \Rightarrow F \vdash X \to Y$

a.) reflexivitási axióma
$Y \subseteq X \Rightarrow X \to Y$
b.) bővíthetőségi axióma
$X \to Y \Rightarrow XZ \to YZ \quad (X \cup Z \to Y \cup Z)$
c.) tranzitivitási axióma
$X \to Y$ és $Y \to Z \Rightarrow X \to Z$

Növetkezik-e egy $W \to L$ függés egy adott F-ből?

$$F^+ = \{X \to Y \mid F \models X \to Y\}$$ // <u>def.:</u> F lezártja

innentől, ha $W \to L \in F^+$, akkor igaz, ha nem akkor meg nem. ezzel csak az a baj, hogy n függés esetén F^+ az 2^n számú függés lesz, szóval nagyon exponenciális.

szóval ez nem jött be

<u>def.:</u>
attribútumhalmaz lezártja

$$X^+ = \{A \mid F \models X \to A\}$$

ez már jó gyorsan számítható

pl.:

$R(A,B,C,D)$

$F = \{A \to B,\ B \to D\}$

$X = AC$

$X^+ = ?$ $\quad X^{(0)} = AC$

$X^{(1)} = AC \cup \{B\} = ABC$

$X^{(2)} = ABC \cup \{D\} = ABCD$

így tudjuk, hogy X szuperkulcs

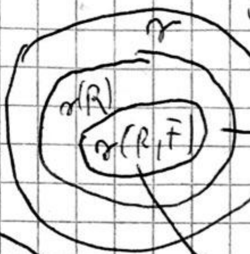
$$\boxed{\underbrace{F \models X \to Y}_{X \to Y \in F^+} \Leftrightarrow Y \subseteq X^+(F)}$$

október 19.,

ismétlés:

$F \models \underbrace{X \to Y}$

emellett a függéshalmaz mellett $\underbrace{\text{ez}}$ is igaz



összes létező reláció

sémára illeszkedő relációk (R séma)

olyan R-re illeszkedő relációk, melyekre F függéshalmaz igaz

lezárt:

$F^+ = \{X \to Y \mid F \models X \to Y\}$ ez túl sok, gyakorlatban nem igazán használható

attribútumhalmaz lezártja:

$X^+(F) = \{A \mid F \models X \to A\}$ na ez már gyorsan számolható

$$F \models X \to Y \iff Y \subseteq X^+(F)$$

nem kell hozzá nagy látnoki képesség, hogy lássuk: F^+ generálható F-ből. de kevesebből (F részhalmazaiból) lehet?

def.: függéshalmazok ekvivalenciája

F és G ekvivalens, ha $F^+ = G^+$

F_{min}: - függés jobb oldalán csak egyszerű attribútum áll
- függés bal oldaláról attribútum nem hagyható el
- függés nem hagyható el

F_{min} olyan halmaz, hogy $F^+ = F_{min}^+$

tétel:

F_{min} $\forall \exists$.

nem mindig célszerű az axiómákkal dolgozni a gyakorlatban, vannak hatékonyabb — de persze az axiómákból levezethető — összefüggések is.

dekompozíciós szabály:

$$X \to Y \text{ és } Z \subseteq Y \Rightarrow X \to Z$$

valami szabály

$$X \to Y \text{ és } WY \to Z \Rightarrow XW \to Z$$

egyesítési szabály:

$$X \to Y \quad X \to Z \Rightarrow X \to YZ$$

pl.: $X \to A_1 A_2, \dots, A_n$

$\left.\begin{array}{l} X \to A_1 \\ X \to A_2 \\ \vdots \\ X \to A_n \end{array}\right\}$

Hogyan essünk neki két függőséghalmaz ekvivalenciájának eldöntésének?

$$F^+ = G^+ \Leftrightarrow F \subseteq G^+ \text{ és } G \subseteq F^+$$

$X \to Y \in F$-re eldöntendő, hogy eleme-e G^+-nak

$$\Longleftrightarrow$$

$$Y \subseteq X^+(G)$$

itt egy jó recept a függőséghalmazok ekvivalenciájának eldöntésére

$F' = \{ \underbrace{\dots\dots}_{\text{függések}}, x_1, x_2, \dots, x_m \to A_1, \dots, Q \to C, \dots \}$

$\{ \dots, \quad x_2, \dots, x_m \to A_1, \dots, Q \to C, \dots \}$

ebből az irányból triviálisan működik a dolog

→ elhagytam x_1-et. ekvivalensek maradtak?

$$x_2, x_3, x_m \to A_1 \overset{?}{\equiv} F'$$

$$\Longleftrightarrow$$

$$(x_2, x_3, \dots x_m)^+ (F') \ni A_1$$

hiszen ha $x \to y$ igaz, akkor $zx \to y$ is, bármit hozzácsaphatok a bal oldalhoz

ezzel így szépen sorban lehet tesztelni a dolgokat, hogy elhagyhatóak-e.

megjegyzés: a minimális függéshalmaz nem egyértelmű! elképzelhető, hogy a sorrendtől függően más jön ki.

Relációs sémák felbontása

$$\rho = \{R_1, R_2, \dots, R_n\} \quad \cup R_i = R$$

→ valahányadik ebben legyen benne az összes attribútum

→ ez R-nek egy felbontása

hogyan tudunk jó sémafelbontásokat készíteni?

pl.: $R(A,B,C)$ $F = \{A \to B\}$

$r(R)$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

ha most a kérdés, hogy a szétszedés után ezt a relációt vissza tudom-e kapni

$$\rho_1 = \{\underset{R_1}{AB},\ \underset{R_2}{BC}\}$$

$$\rho_2 = \{AB,\ AC\}$$

de alapvetően nyilván valami olyasmit kéne általánosítani, ami $\forall$ relációra igaz lesz

project-join-mapping: $m_\rho(r)$ =

$m_\rho(r) \supseteq r$

<u>def</u>: veszteségmentes sémafelbontás

$\boxed{\forall\, r(R) = \bowtie_i \tilde{\pi}_{R_i}\, r(R)}$ esetén $\rho = \{R_i\}$ veszteségmentes

lássunk neki: $\tilde{\pi}_{AB}(r)$

A	B
a_1	b_1
a_2	b_1

, $\tilde{\pi}_{BC}(r)$

B	C
b_1	c_1
b_1	c_2

, $\tilde{\pi}_{AC}(r)$

A	C
a_1	c_1
a_2	c_2

hiányzik a többi!

veszteséges felbontás (tudás veszett el: több az adat)

ρ_1: $\tilde{\pi}_{AB}(r) \bowtie \tilde{\pi}_{BC}(r) =$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_2

$\neq r(R)$

meg lehet veszteségmentes felbontás, de ez csak egy

ρ_2: $\tilde{\pi}_{AB}(r) \bowtie \tilde{\pi}_{AC}(r) =$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

$= r(R)$

na most akkor hogyan döntsük el, hogy akkor mi van?

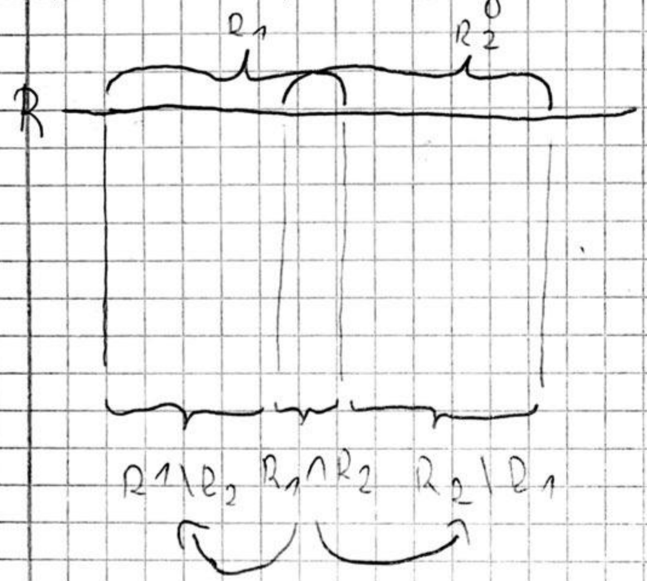
tétel:

$\rho(R_1, R_2)$ sémafelbontás, F

ρ veszteségmentes $\iff R_1 \cap R_2 \to R_1 \setminus R_2$ vagy

$R_1 \cap R_2 \to R_2 \setminus R_1$

na ez már a mágiának a határait feszegeti...



szintézis: $F = \{ \dots, X \to Y, \dots \}$

$$\rho(R_1,R_2) \quad \left.\begin{array}{l} R_1 = XY \\ R_2 = R \setminus Y \end{array}\right\} \quad \left.\begin{array}{l} R_1 \cap R_2 = X \\ R_1 \setminus R_2 = Y \end{array}\right\} \quad \begin{array}{l} R_1 \cap R_2 \to R_1 \setminus R_2 \\ \text{teljesül.} \end{array}$$

Hurrá! itt a recept, így kell csinálni.

Y-nak azért ne legyen olyan attribútuma, ami benne van X-ben, mert akkor kiborul a lili.

több relációséma esete:

tétel:

$m_\rho(r) \supseteq r$

tétel:

$\rho = \{R_1, R_2, \dots, R_n\}$ veszteségmentes, akkor $\tau = \rho \cup S$ is veszteségmentes lesz.

biz.:

$$m_\rho(r) = \underbrace{\tilde{\pi}_{R_1}(r) \bowtie \dots \bowtie \tilde{\pi}_{R_n}(r)}_{r(R)} \bowtie \tilde{\pi}_S(r) = r(R)$$

tétel: $\rho(R_1, \dots, R_k)$ veszteségmentes

$\cancel{\rho(R_1, \dots R_i, A_{i_1}, A_{i_2}, \dots, A_{i_m}, \dots)}$

$$\rho'(R_1, \dots, R_{i-1}, \underbrace{R_{i_1}, R_{i_2}, \dots, R_{i_m}}_{\substack{\tau:\ R_i\text{-nek egy} \\ \text{veszteségmentes} \\ \text{felbontása}}}, R_{i+1}, \dots, R_k)$$

az igazi az lenne, hogy egy akármennyi relációsémából állót hogyan kell veszteségmentesen felbontani:

táblázatos módszer:

a felbontás pontosan akkor lesz veszteségmentes, ha azzal kijön

- $\rho(R_1, R_2, \dots, R_m)$ $R(A_1, A_2, \dots, A_n)$ $F = \{ \}$

formálisan nem írjuk le.

pl.: $R(S, A, I, P)$ $F = \{S \to A, SI \to P\}$

$$\rho = \{\underset{R_1}{SA}, \underset{R_2}{SI}, \underset{R_3}{IP}\}$$

	S	A	I	P
SA	a_1	a_2	b_{13}	b_{14}
SI	a_1	b_{22}	a_3	b_{24}
IP	b_{31}	b_{32}	a_3	a_4

így inicializáljuk a táblázatot.

a_i: benne van

b_{ij}: nincs benne

- erre kell ráeröltetni a függőségeket, így:

	S	A	I	P
SA	a_1	a_2	b_{13}	b_{14}
SI	a_1	$\cancel{b_{22}}$ a_2	a_3	b_{24}
IP	b_{31}	b_{32}	a_3	a_4

így ez a táblázat egy reláció, ami illeszkedik.

- tétel: a felbontás, pontosan akkor veszteségmentes, ha van csupa a sor.

október 26. (~~Fizikai adatszervezés~~)

mindjárt

de előtte veszteségmentes sémafelbontás (~~[illegible]~~):

R(Város, Utca, Irány)

$F = \{I \to V, VU \to I\}$ $\rho(IV, VI)$ felbontás

$$\left.\begin{array}{l} R_1 \cap R_2 = I \\ R_1 \setminus R_2 = V \end{array}\right\} R_1 \cap R_2 \overset{?}{\Rightarrow} R_1 \setminus R_2 \quad I \to V \checkmark$$

tényleg veszteségmentes lett? így lehet ellenőrizni.

V	I
Baja	2142
Baja	2143
⋮	

U	I
Fa u.	2142
Fa u.	2143
⋮	

I	V	U
2142	Baja	Fa u.
2143	Baja	Fa u.
⋮		

az történt, hogy a természetes illesztés miatt belekerültek olyan dolgok, amik eredetileg nem voltak ott.

$UV \to I$ nem teljesül, így valami baj van, ez itt ~~[illegible]~~ veszteségmentes, de nem függőségőrző.

nézünk valami olyan észre, ami függőségőrző is.

def.: Vetített függéshalmaz

$$\Pi_{R_i}(F) = \{X \to Y \mid X, Y \subseteq R_i \land F \models X \to Y\}$$

„szedd elő az összes olyan függést, amik igazak és az adott attribútumhalmazban benne vannak"

<u>def.</u>: függőségőrző sémafelbontás

$\rho(R_1, R_2, \ldots, R_n)$ sémafelbontás

ρ függőségőrző $\Leftrightarrow \bigcup_i \pi_{R_i}(F) \models F$

ez nem csoda, mert az előbb volt egy 3 attribútumos függőségünk ($VL \rightarrow I$), amit nyilván nem lehet 2 attribútummal tartani (hiszen csak ilyen részsémákat gyártottunk)

össz. kép: általános esetben a 3 feltételt, a redundancia csökkentését, a veszteségmentességet és a függőség-őrzést nem lehet garantálni.

de

a veszteségmentesség és a függőség-őrzés mellett 3NF-et lehet

vagy

BCNF és veszteségmentesen, de nem függőségőrzően

<u>tétel</u>: R, F_{min} $\quad F_{min} = \{X_1 \rightarrow A_1, X_2 \rightarrow A_2, \ldots, X_n \rightarrow A_n\}$

$$\rho \underbrace{\{X_i A_i\}}_{R_i} \Rightarrow 3NF(R_i) \text{ és } \rho \text{ függőségőrző}$$

<u>tétel</u>: $\rho' = \{X_i A_i\} \cup K \Rightarrow 3NF(R_i)$ függőségőrző és veszteségmentes

K: R egyik kulcsa

tudomány állása szerint magánnem -kánaán ha az nekünk nem felelne meg:

<u>tétel</u>: adott R és F esetén mindig $\exists \rho$, hogy ρ veszteségmentes $\forall i$ $BCNF(R_i)$

<u>biz</u>: $\neg BCNF(R) \Rightarrow \exists\, X \to A,\ A \notin X$, X nem szuperkulcs

$\rho_1(XA, R \setminus A)$ ez garantáltan veszteségmentes, na most ha ρ_1 elemei közül valamelyik nem BCNF, akkor azt a fentiek szerint tovább bontom és ezt iterálva csinálom, amíg nincs vége.

ez így veszteségmentes, de függőségőrzőséget nem garantál.

bebizonyítani, hogy valami vagy 3NF és vesztment és függőrző vagy BCNF és vesztmentes könnyű, az ellenkezőjére a példa az.

na most...

Fizikai adatszervezés

eddig ∀ ami volt, a középső, koncepcionális rétegbe tartozott.

Feltételek:
- rekord-szerű struktúrák
- háttértár blokkos szerkezetű (HDD)
- fájl alapú (az adatbáziskezelő az oprendszer felett ül)

mi csak azzal fogunk foglalkozni, hogy ennek a fájlnak a belsejét hogyan alakítsuk ki.

- támogatott műveletek:
  - keresés
  - törlés
  - beszúrás
  - módosítás

fájl -> blokk

blokk: legkisebb adategység, amit a háttértár és az operatív tár között mozgathatunk.

fizikai rekordok } azonos hosszúságúak blokkon belül

tipikus blokk: | | R_1 | R_2 | ... | R_k | /// | -> töredék-rekordokat nem tárolunk

(ha mégis, 0 pontos feladat a gitalon)

blokk header } mindenféle státusz-infók és mutatók

a háttértár olcsóbb, mint a számítás, ezért

rekord: | | u_1 | u_2 | ... | u_j |

rekord header

műveleteket akarunk optimalizálni: a költségeket blokkműveletekben definiáljuk (cpu gyors, háttértár lassú)

Fő cél: blokkműveletek számának minimalizálása

3 alapvető szervezési módszer

kötött/szabad rekordok

- heap
- hash
- indexelés

Heap: segédstruktúrák nélküli szervezés.
prosztó, egyszerű megoldás: ennél csak jobb lesz, de alapból így működik, ha nem kérjük explicite

Keresés: egyedi kulcs-érték alapú keresés:
„szedd elő abból a rekordból azt és azt a mezőt, amit ez az azonosító azonosít"

végig kell keresni az összes blokkot, amíg meg nem találjuk.

$$\text{időigény: } \left.\begin{array}{l} -\text{min: } 1 \text{ blokk} \\ -\text{max: } n \text{ blokk} \end{array}\right\} - \text{átlag } \frac{1+n}{2}$$

október 28.,

mezőkben mutatók:

- fizikai: konkrét fizikai cím, offszetek, stb.
- logikai: valahogy egyértelműen azonosít.

egy blokkon belül a rekord hossza alapból azonos
(a nem relációs modellekben nem feltétlenül)

↳ ilyen esetekben az a tipikus megoldás, hogy elöl vannak a fix hosszúak, utána még egy mutató a változó hosszúságúakra.

heap: nem léteznek segédstruktúrák: végig kell keresni az egész fájlt mondjuk egy sima kereséshez. a keresési idő a fájl méretének lineáris függvénye

törlés heapből:

(rekordazonosító alapján)
először egy keresés, aztán a törlés úgy néz ki, hogy a headerben bebillentünk egy "deleted" bitet (de az adatokhoz nem nyúlunk).
ezért az időigénye: keresés + 1

beszúrás heapbe:

ezt azért csináljuk, mert így vissza lehet hozni az adatot

valamilyen azonosítójú rekordot be akarok szúrni. meg kell nézni, hogy ilyen van-e már, mert biztosítani kell az egyediséget.
időigénye: N (végig kell menni az egész fájlon)
+
1 (a beszúrás maga)

módosítás heapben:

meg kell keresni azonosító alapján

ha érint egyediséget biztosító mezőt: lásd beszúrás

nem érint: keresés + 1

ha a heap nem elég gyors, azt látjuk, hogy a keresést kéne felgyorsítani.

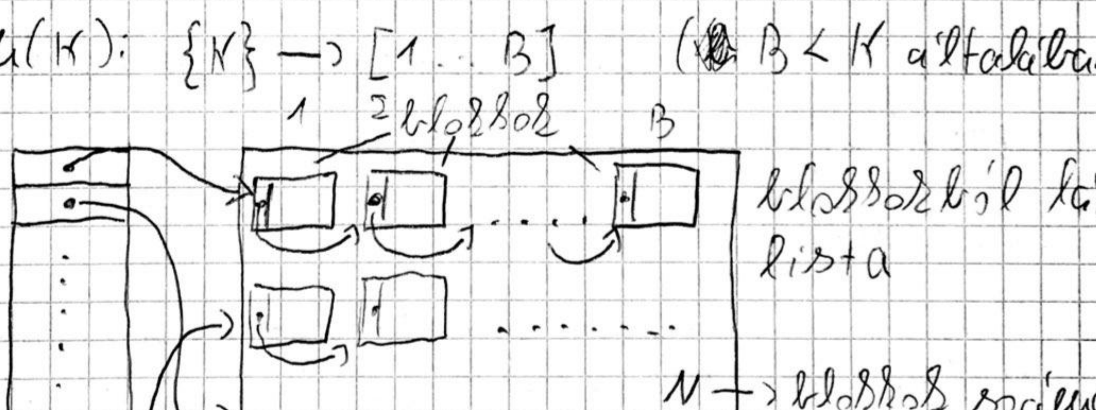
Hash

bucket-hashing: „vödrös hashelés"

keresés helyett $\underbrace{\text{számítás}}_{\text{hash-függvényen keresztül}}$.

$h(K)$ — azonosító

$h(K): \{K\} \to [1 .. B]$ ($B < K$ általában)



olvasd ki a hash tábla $h(K)$-adik elemét, azon a mutatón mutatott helyen kell valahol K-nak lennie, azon belül heap legegyszerűbb hash-fv: $C \cdot K \bmod B$

mit nyertünk ezzel?

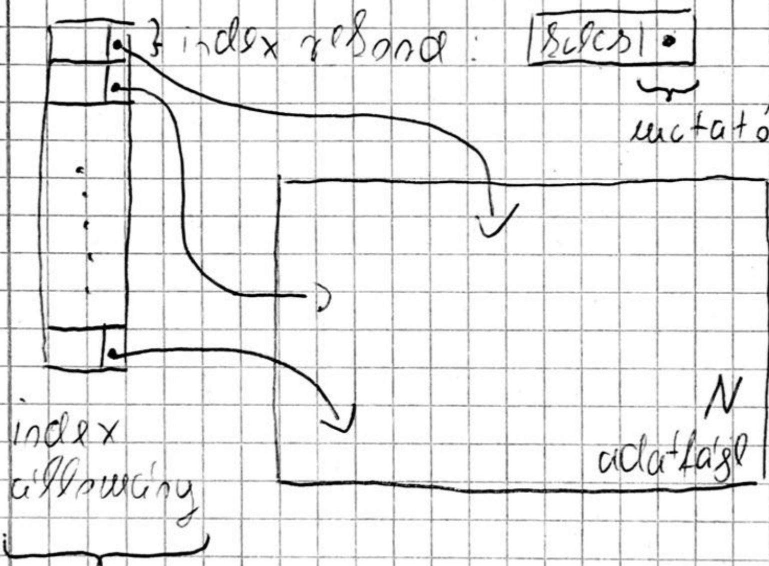
idők: - keresés: $\dfrac{1+\frac{N}{B}}{2}$

egy kis operatív tárral a keresés idejét a töredékére csökkenthetjük

legjobb esetben, ha $N = B$, akkor bármit elérek 1 blokkművelettel.

- törlés: keres + 1
- beszúrás: $\frac{N}{B} + 1$
  ↳ egy lánc hossza
- módosítás: - ha kulcsérték változik: töröl + beszúr
  - ha nem: keres + 1

Indexelt állományszervezés



az ∀ esetben a kulcsérték szerint rendezett

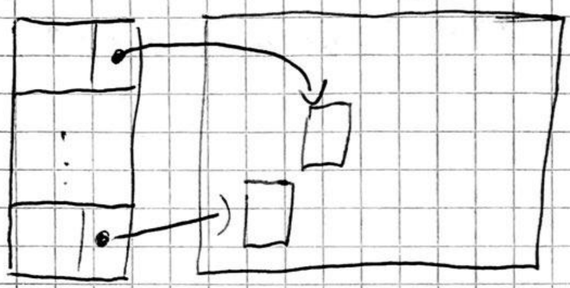
keresünk az index állományban, az meg valahogy megcímzi a rekord helyét

M blokk

indexben keresés: bin: $\underbrace{\lceil \log_2 M \rceil}_{\text{indexben keresés}} + \underbrace{1}_{\text{adatfájlból olvasás}}$ blokkművelet

típusai: – ritka index: kevesebb index rekord, mint adatrekord
– sűrű index: ezek megegyeznek

spec. ritka index: <u>ISAM</u>



a mutatók pontosan egy db adatblokkot címeznek

az indexben a blokkban található legkisebb kulcsérték lesz

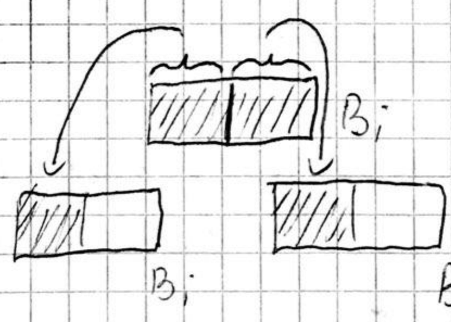
idők: – keresés: $\left[\log_2 M\right] + 1$

ezzel megvan az adatblokk és blokkon belül is rendezve vannak kulcs szerint a dolgok

keresd meg azt a kulcsot az indexben, ami kisebb, mint amit keresel, de ezek közül a legnagyobb

↳ tanulság: adatállomány az index kulcsa szerint rendezett legyen!

– beszúrás: a beszúrás helye adott. na most ott vagy van hely, vagy nincs. ha nincs, blokkhasítás



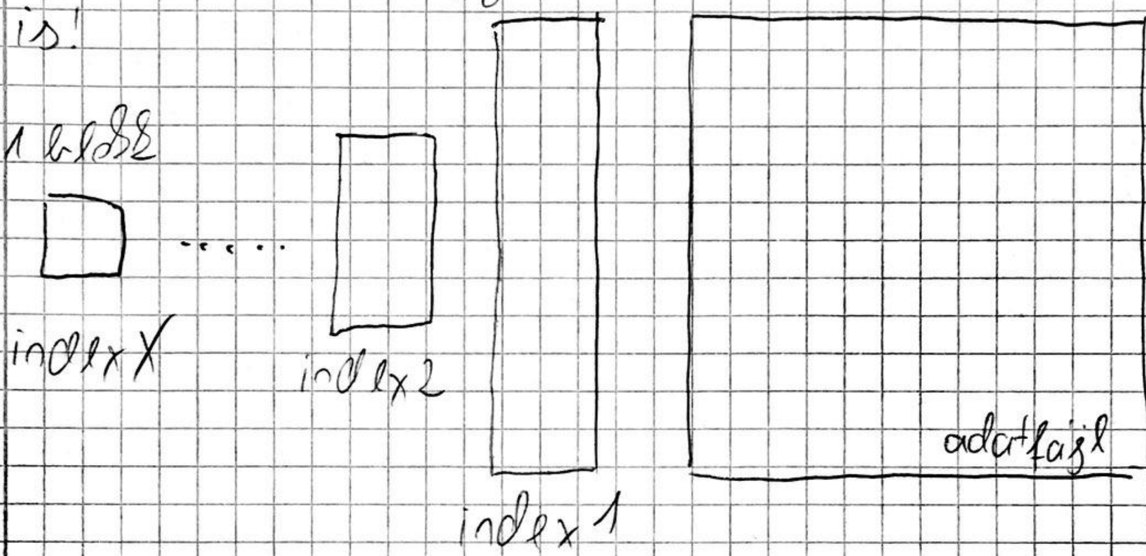
blokk-hasítás (block split)

$\underbrace{\text{+ indexrekord előállítása}}_{}$

ugyanezt kell csinálni, ha nincs hely, indexblokk-hasítás

- törlés: itt valahogy a block-split ellentettjét kellene csinálni: blokk-összevonás
- módosítás: ha kulcsérték változik: törölt + beszúr

mit kéne csinálni, hogy gyorsítsuk az indexben való keresést? előbb így gyorsítottuk az adatfájlban való keresést, hogy indexeltünk. indexeljük az indexet is!



$$k^{x-1} \geq M$$

$$\underbrace{(x-1)\log_k k}_{1} \geq \log_k M$$

$$\underbrace{x \geq 1 + \lceil \log_k M \rceil}$$

$1 = k^0$	1
$k = k^1$	2
k^2	⋮
⋮	
k^{x-1}	X

B* fa keresési ideje

többszintű ritka index
- block-based
- balanced
- Bayer

- keresés: $1 + \lceil \log_k M \rceil$

k akár 100 is lehet!

ez kurva gyors.

rekordszám: 7 MRD | kulcs: 32 bit
blokkméret: 8 kbyte

november 2. (fizikai adatszervezésnél: kulcs - ami szerint keresünk)

- emlékeztető: a keresés ideje dominál a műveleteknél

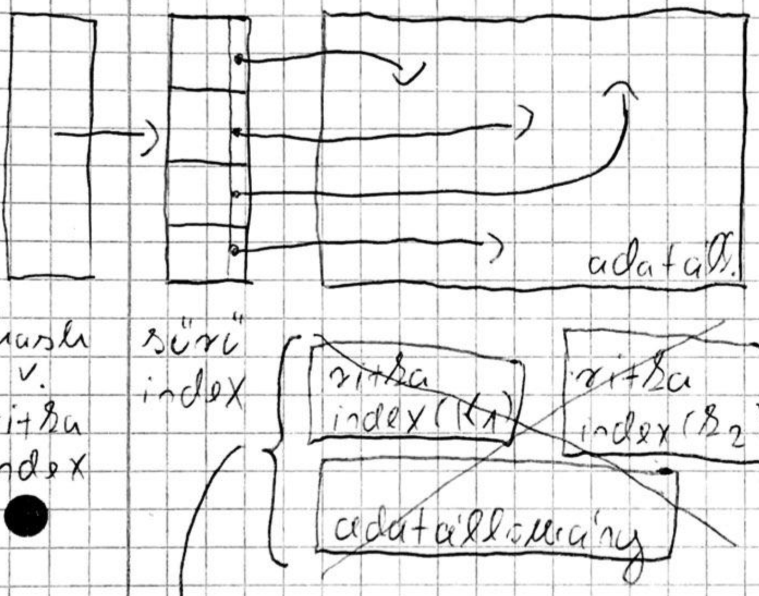
hashelés: keresés konstans
indexelés: - ritka
- sűrű

keresés valami log alapú tipikusan

kimaradt dolgok:

<u>sűrű indexek:</u> indexek száma = állományok száma.

- önmagában nem adatszervezési mód, csak segédeszköz

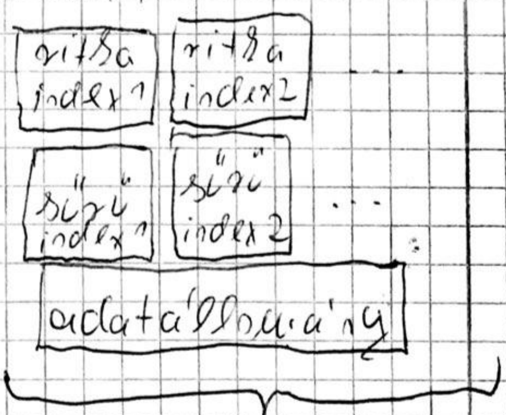


ilyet nem lehet! újra kéne rendezni az adatállományt

helyette

előnyei:
+ adatállomány nem rendezett
+ több kulcs szerinti rendezés
+ ha jóval kisebbek az indexek, mint az adatok, gyorsabb tud lenni

hátrányai:
+ blokkművelet
+ hátlértár
+ adminisztráció



több kulcs szerinti keresés

címszavak szintjén miről kellett még beszélni:

- intervallumkeresés
- egyediség nem garantálható
- részleges információ alapján keresés (particionált hash)

- 29

Particionált hash:

$$h(K) \triangleq \underbrace{h_1(K_1) \circledast h_2(K_2) \circledast \ldots \circledast h_N(K_N)}_{\sum_i h_i}$$

$K: K_1 K_2 \ldots K_N$

$\downarrow \quad \downarrow \quad \downarrow$

$h_1(K_1) \ldots h_N(K_N)$

bitek száma $m_1 \ldots m_n$

most, hogy vége a fizikai adatszervezésnek, visszatekintés:

<u>tétel</u>: $3NF(R) \Rightarrow 2NF(R)$

<u>biz</u>.: $\neg 2NF(R) \Rightarrow \neg 3NF(R)$

$\exists$ x kulcs és $\exists$ A másodlagos attribútum esetén: $X' \subset X \quad X' \rightarrow A$ (nem függ teljesen)

$$\underbrace{X \rightarrow X' \quad X' \not\rightarrow X \quad X' \rightarrow A \quad \underbrace{A \notin X'}}$$

✓ ez itt egy szép tranzitív függőség

igaz, hiszen A másodl. attrib.

definíciók ekvivalenciája:

def1: X kulcs, A másodl. attr.

$\nexists y$: $X \rightarrow y$, $y \not\rightarrow X$, $y \rightarrow A$, $A \notin y$

def2: $\forall$ $y \rightarrow B$ $B \notin y$: - y szuperkulcs vagy
- B elsődleges attrib.

tétel: def1 $\Leftrightarrow$ def2

biz.: a, def1 $\Rightarrow$ def2

$\underbrace{\neg def2}$ $\Rightarrow$ $\neg$ def1

másodlagos attribútum

$\exists\, y \to B$, $B \notin y$:
- y nemszuperkulcs és
- B másodlagos

$\Rightarrow$ $\underbrace{X \text{ kulcs } \; X \to y \;\; y \not\to X \;\; y \to B \;\; B \notin y}$

tranzitív függőség, indirekció következménye

$\exists \neg$ def1 ✓

b, def2 $\Rightarrow$ def1

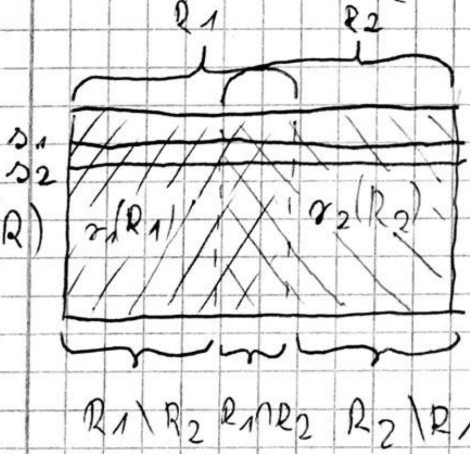
$\neg$def1 $\Rightarrow$ $\neg$def2 ... jegyzetben

tétel: (R, F), $\rho(R_1, R_2)$

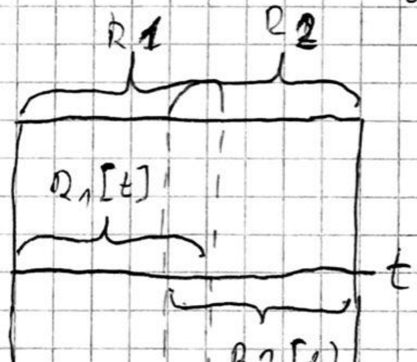
ρ veszteségmentes $\Leftrightarrow$
- $R_1 \cap R_2 \to R_1 \setminus R_2$, vagy
- $R_1 \cap R_2 \to R_2 \setminus R_1$

biz.: a, ha
- $R_1 \cap R_2 \to R_1 \setminus R_2$, vagy
- $R_1 \cap R_2 \to R_2 \setminus R_1$

$\Rightarrow$ ρ veszteségmentes



$r_1 \bowtie r_2 = m_\rho(r)$:



tetszőleges $t \in r_1 \bowtie r_2 = m_\rho(r)$-ről megmutatjuk, hogy $t \in r(R)$

$\exists\, s_1 \in r_1(R_1)$

$s_1[R_1] = t[R_1]$

$\exists\, s_2 \in r_2(R_2)$

$s_2[R_2] = t[R_2]$

$t[R_1 \cap R_2] = s_1[R_1 \cap R_2] = s_2[R_1 \cap R_2]$

$R_1 \cap R_2 \to R_1 \setminus R_2 \Rightarrow$ ha $s_1[R_1 \cap R_2] = s_2[R_1 \cap R_2]$
akkor $s_1[R_1 \setminus R_2] = s_2[R_1 \setminus R_2]$

$\Downarrow$

- $t \in m_\rho(r)$ és $\quad t = s_2$
- $s_2 \in r(R)$

nem keletkezne több sor, mint volt $\Rightarrow$ veszteségmentes ✓

b.)

ρ veszteségmentes $\Rightarrow$ - $R_1 \cap R_2 \to R_1 \setminus R_2$, vagy
- $R_1 \cap R_2 \to R_2 \setminus R_1$

indirekt: - $R_1 \cap R_2 \not\to R_1 \setminus R_2$ és ...
- $R_1 \cap R_2 \not\to R_2 \setminus R_1$ $\Rightarrow \nexists\, \rho$ veszteségmentes

megfelelő ellenpélda:

R_1				R_2
1 ... 1	1	1	1	0 ... 0
0 ... 0	1	1	1	1 ... 1

$R_1 \cap R_2$

$(R_1 \cap R_2)^+$

tetszőleges $X \to Y \in F$ teljesül-e?

a.) tfh. X kilóg $(R_1 \cap R_2)^+$-ból

nincs két azonos sor r-ben: X-en $\Rightarrow X \to Y$ teljesül

b.) tfh. $X \subseteq (R_1 \cap R_2)^+ = R_1 \cap R_2 \to X \left.\begin{matrix} \\ X \to Y \end{matrix}\right\} R_1 \cap R_2 \to Y$

✓ $Y \subseteq (R_1 \cap R_2)^+ \Leftarrow$