Lekérdezések optimalizálása osztott adatbázisokban

Költség: átvitt adatok mennyisége (mérete)

Relációk részekre (fragments) bontása

logikai relációk (virtuális relációk)
- részekből ∪ és ⋈ műveletekkel állíthatók elő

fizikai relációk:
- adatbázisban tárolt töredékei (fragments) a logikai relációknak

$R = R_1 \bowtie \dots \bowtie R_n$ R_i vertikális töredék

$R = R_1 \cup \dots \cup R_n$ R_i horizontális töredék

- a töredékek is lehetnek még logikai relációk és így tovább
- a töredékek különböző csomópontokon helyezkedhetnek el

Példa: BANK

számlák (fiók, szszám, egyenleg)
hitelek (fiók, hszám, összeg)
tulaj (szszám, unév)
tartozik (hszám, unév)
ügyfelek (unév, cím)

- a fiókok saját számláikat és hiteleiket tartják nyilván
  - előny: általában nincs szükség központi adatforgalomra

Számlák és Hitelek horizontális felbontása:

$$\text{számlák} = \text{számlák}_{f_1} \cup \text{számlák}_{f_2} \cup \ldots \cup \text{számlák}_{f_k}$$

$$\text{hitelek} = \text{hitelek}_{f_1} \cup \ldots \cup \text{hitelek}_{f_k}$$

ahol $f_1, \ldots, f_k$ a lehetséges fiókok

Hasonlóan

$$\text{tulaj} = \bigcup \text{tulaj}_{f_i}$$

$$\text{tartozik} = \bigcup \text{tartozik}_{f_i}$$

Az ügyfeleket is felbonthatnánk hasonlóan nem feltétlen diszjunkt unióra, de egyszerűség kedvéért feltesszük, hogy a központban tároljuk az összes ügyféladatot.

$$\underset{\substack{\uparrow \\ \text{logikai}}}{R} = \text{[illegible]}\ \underbrace{\text{tulaj} \bowtie \text{ügyfelek}}_{\text{vertikális felbontás}} = \left(\bigcup \text{tulaj}_{f_i}\right) \bowtie \text{ügyfelek}$$

Töredékekre vonatkozó lekérdezések

Relációs algebrai kifejezés $E(R_1, \ldots, R_k)$

- R_i logikai relációk: töredékek kifejezései

Más módon – a töredékekre vonatkozó kifejezésre az algebrai optimalizációt alkalmazzuk

Módosítás őrfeltételek segítségével:

horizontális töredék:

$$\text{hitelek}_{f_i} = \sigma_{\text{fiók}=f_i}(\text{hitelek}) \text{ miatt}$$

$$\sigma_{\text{fiók}=f_i}(\text{hitelek}_{f_i}) = \text{hitelek}_{f_i}$$

↖ őrfeltétel azonosan igaz a töredékre

Feltehető, hogy minden logikai és fizikai relációnak (R) van g őrfeltétele, azaz $\sigma_g(R)=R$

~~[illegible]~~

1) töredékeket őrfeltétellel írjuk fel
2) kiválasztások legjelentkezésénél ha g-nek ellentmond a feltétel, akkor kihagyjuk R-t.

Példa ~~legyen~~ fiók: $\{1,2,3\}$

Q = ügyfelek, akiknek 1-ben van számlájuk, melynek egyenlege >1000

legyen a séma most

$$R(\text{fiók}, \text{szszám}, \text{egyenleg}, \text{unév}) = R_1 \cup R_2 \cup R_3$$

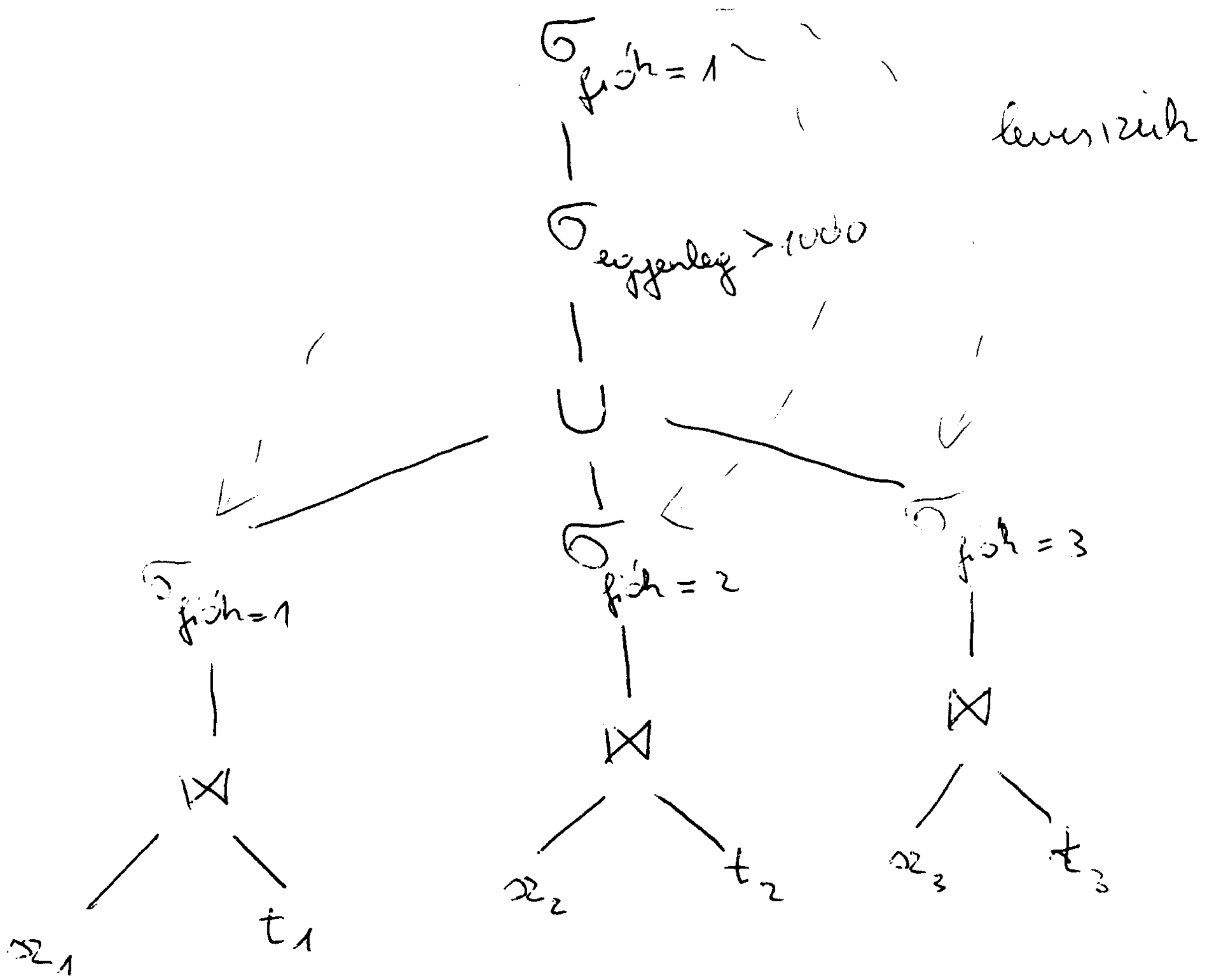
↑ logikai reláció

$$R_i = \text{számlák}_i \bowtie \text{tulaj}_i \qquad i=1,2,3$$

Ekkor R_i-k őrfeltétele: fiók $= i$

$$Q = \sigma_{fiók=1 \wedge egyenleg>1000}(R) = \sigma(R_1 \cup R_2 \cup R_3) =$$

$$= \sigma_{fiók=1 \wedge egyenleg>1000}\left(\sigma_{fiók=1}(számla_1 \bowtie tulaj_1) \cup \sigma_{fiók=2}(sz_2 \bowtie t_2) \cup \sigma_{fiók=3}(sz_3 \bowtie t_3)\right)$$

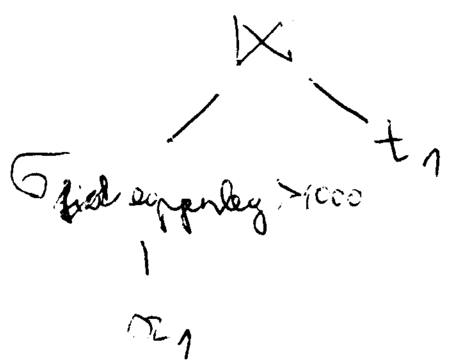


$\underbrace{\sigma_{fiók=1 \wedge fiók=2}}_{ellentmondás}$ → U-ból 1 marad: U-t elimináljuk

→ $\sigma_{egyenleg>1000}$-t levisszük

→ σ-feltétel triviálisan teljesül, elhagyjuk

Marad:



1 csomópontban ~~ki~~ kiértékelhető

Logikai relációk módosítása

Beszúrás (ha töredékek közt van logikai, akkor rekurzívan)

1) $R = R_1 \bowtie \ldots \bowtie R_n$

insert t into R:

$\forall i$-re insert $t[R_i]$ into R_i

2) $R = R_1 \cup \ldots \cup R_n$

insert t into R:

keressünk olyan R_i-t, amelynek az őrfeltételét t kielégíti és insert t into R_i

ha nincs ilyen: nem lehet beszúrni

ha több ilyen van, akkor ~~egy~~ lehetőleg azt az i-t válasszuk, ahol kiadták az insert utasítást (helyi műveletek előnye!)

Törlés:

1) $R = R_1 \cup \ldots \cup R_n$

delete t from R:

$\forall i$ delete t from R_i

2) $R = R_1 \bowtie \ldots \bowtie R_n$

delete t from R: - probléma,

$t[R_i]$-t nem biztos hogy R_i-ből törölhetjük, mert más t' sorhoz is tartozhat ($t[R_i] = t'[R_i]$ esetén)

Megvalósítás - beszúrásnál sorazonosítók használata

R és R_i sémáját kiegészítjük ~~TID~~ SORAZ oszloppal

$$R = R_1 \bowtie \dots \bowtie R_n$$

	SORAZ
t	a_t

	SORAZ
$t[R_1]$	a_t

	SORAZ
$t[R_n]$	a_t

így delete (t, a_t) from R

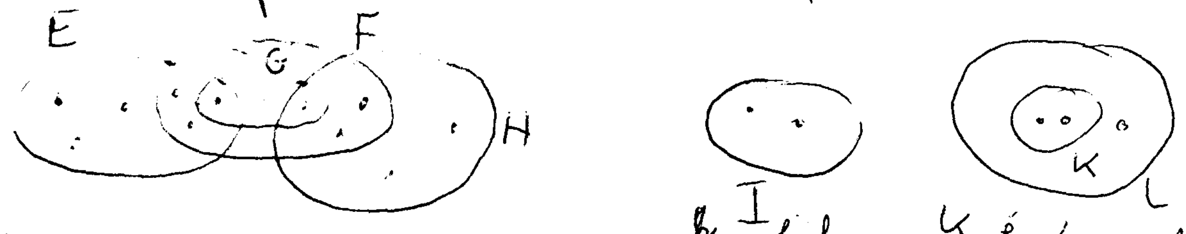
$\forall i$ delete $(t[R_i], a_t)$ from R_i

Megjegyzés: (t, a_t) többször is szerepelhet R-ben, ekkor $(t[R_i], a_t)$-ből is több lehet R_i-ben.

Aciklikus hipergráfok

Ötlet: fa aciklikus: - mindig leveleket letépve, végül nem marad semmi

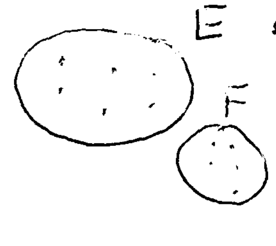
Def: E, F hiperélek (~~nem feltétlen különböző~~)
E fül F miatt, ha E − F csúcsai nem szerepelnek más hiperélben csak E-ben



~~ha~~ I fül K és L is fül

Megjegyzés: fül ⇔ levél (közönséges gráf)

Speciális eset: − E nem metsz más hiperélt ⇒ E fül



– 1 hipérél ~~ss~~ van ⇒ fül

Def: hipergráf aciklikus, ha mindig füleket eltávolítva nem marad hipérél. Ciklikus, ha nem aciklikus.

Füllevágás: GYO-redukció:

Graham – Yu – Ozsoyoglu [1979]

Kell: egyértelmű

Tétel: aciklikusság nem függ a füllevágás sorrendjétől

Bizonyítás:

Elég: ha egy fület nem vágok le, akkor továbbra is fül marad, tehát később is levághatom

Legyen E_1 fül E_2 miatt.

Ha nem E_2-t vágjuk le, akkor $E_1 - E_2$ csúcsai továbbra is csak E_1-ben szerepelhetnek, tehát E_1 fül marad

Ha E_2 fül E_3 miatt és E_2-t levágjuk.

Elég lenne: $E_1 - E_3 \subseteq E_1 - E_2$, mert akkor E_1 fül lesz E_3 miatt.

indirekten: $N \in E_1 - E_3$

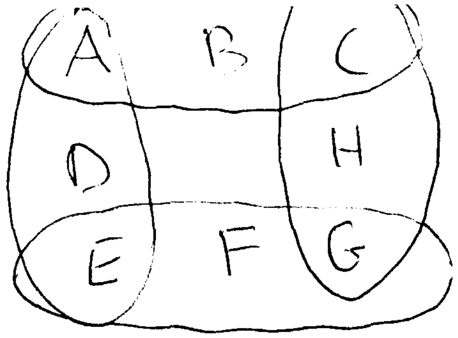
$N \notin E_1 - E_2 \rightarrow N \in E_2 \cap E_1$

$\Rightarrow N \in E_2 - E_3$, de $N \in E_1$ is, de akkor E_2 nem lehetne fül E_3 miatt

$\square$

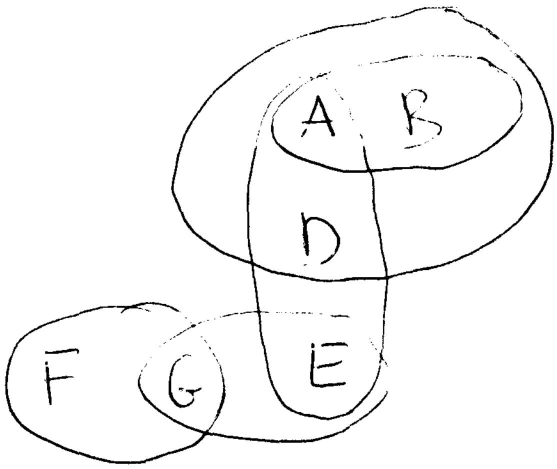
Példák

1)



ciklikus, mert nincs fül

2)



{FG} fül GE-re

GE fül ADE-re

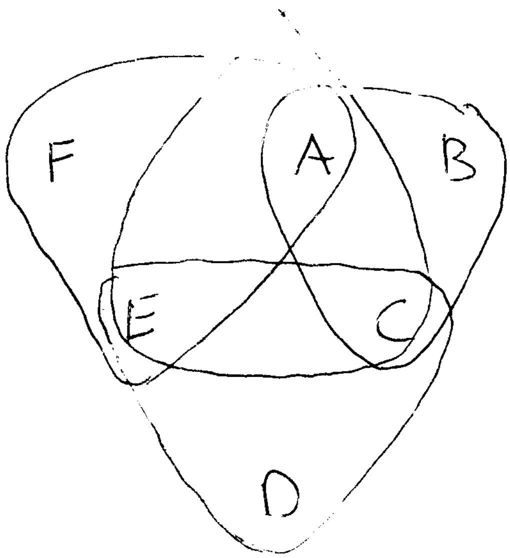
AB fül ABD miatt

ADE fül ABD miatt

ABD 1 hiperél eltá-
volítható

aciklikus

3)



ABC, CDE, EFA kör van,
mégis aciklikus

ABC fül ACE miatt

CDE —||—

EFA —||—

végül ACE marad, ami
eltávolítható.

Átviteli költség csökkentése félig összekapcsolásokkal

$R \bowtie S$ 1 csomópontban van:
nem érdemes a Wong-Youssefi algoritmusban
$\ltimes$-t használni (eliminációs lépés!)

Közel egyforma rekordméretet feltételezünk a különböző relációkra.

n sor átviteli költsége legyen $C_0 + n$

$$R \xrightarrow{R} S \qquad T_R + C_0$$

vagy

$$R \xleftarrow{S} S \qquad T_S + C_0$$

azaz $\boxed{C_0 + \min(T_R, T_S)}$

$R \bowtie S$ egyik helyen kell előállítani

Félig összekapcsolással

$$R \xleftarrow{\Pi_{R\cap S}(S)} S \qquad \Pi_{R\cap S}(S)$$

$$\underline{R \ltimes S = R \bowtie \Pi_{R\cap S}(S)} \longrightarrow$$

$$R \bowtie S = (R \ltimes S) \bowtie S$$

komplex

$$T'_S = |\Pi_{R\cap S}(S)|$$
$$T'_R = |\Pi_{R\cap S}(R)|$$
$$T''_S = |S \ltimes R|$$
$$T''_R = |R \ltimes S|$$

esetén

$$2c_0 + \min(T'_S + T''_R,\ T'_R + T''_S)$$

Mikor jobb?

Ha $C_0 + \min(T_S' + T_R'', T_R' + T_S'') < \min(T_R, T_S)$

1. Ha C_0 fix költség nem nagy még kevés rekord átviteli költségéhez képest sem. $C_0 \ll n$
2. Ha R, S nagyjából egyforma méretű
3. $T_S', T_S'' \ll T_S$

   R R R

   ~~kevés~~ sok fityegő sor van

   sok sor megegyezik a közös oszlopokon

Több tábla esetén tovább javul a költség, mivel egyre több fityegő sor keletkezhet. A félig összekapcsolások kiszűrik a fityegő sorokat.

Félig összekapcsolásos (semijoin) programok

$R = R_1 \bowtie \dots \bowtie R_k$

$R_i = R_i \ltimes R_j$ félig összekapcsolásos lépés ($\ltimes$-lépés)

Def. Félig összekapcsolásos program: $\ltimes$-lépések sorozata

Példa:

AB	BC	CD
1 2	1 2	1 2
2 4	2 4	2 4
3 6	3 6	3 6
4 8	4 8	4 8

$AB \ltimes BC \bowtie CD$

Mely fityegő sorok esnek ki?

$AB := AB \ltimes BC$ (3,6), (4,8)

$BC := BC \ltimes CD$ (3,6), (4,8)

$CD := CD \ltimes BC$ (1,2), (3,6)

$AB \bowtie BC \bowtie CD = (1,2,4,8)$ 1 sor

AB, BC, CD mindegyikében még maradt fityegő sor, nevezetesen a (2,4)

Cél: minél több, lehetőleg az összes fityegő sor eltüntetése, mivel felesleges fityegő sorokat nem akarunk a $\ltimes$-nal a hálózaton továbbítani

Teljes reduktáló

Def: R_i redukált $R_1, \dots, R_k$-ra nézve, ha nincs fityegő sora, azaz

$$R_i = \Pi_{R_i} (R_1 \bowtie \dots \bowtie R_k)$$

Félig-összekapcsolásos program teljes reduktáló $R_1, \dots, R_k$-ra, ha a program végrehajtása után minden R_i redukált, (függetlenül az R_i-k előfordulásától)

Példa:

$BC := BC \ltimes AB$

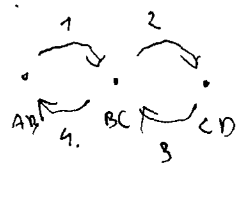
$CD := CD \ltimes BC$

$BC := BC \ltimes CD$

$AB := AB \ltimes BC$

teljes reduktáló

Ez jó lesz majd a tétel miatt tetszőleges előfordulásra.



Példa: nincs mindig teljes reduktáló

AB	BC	AC
$a_1 b_1$	$b_1 c_1$	$a_2 c_1$
$a_2 b_2$	$b_2 c_2$	$a_3 c_2$
$a_n b_n$	$b_n c_n$	$a_{n+1} c_n$

minden oszlopban különböző értékek

$AB \bowtie BC \bowtie AC = \emptyset$ → minden sor fityegő, mindet el kellene távolítani $\ltimes$ segítségével

$$\left\{\begin{array}{l} AB := AB \ltimes AC \\ BC := BC \ltimes AB \\ AC := AC \ltimes BC \end{array}\right.$$

(a_1, b_1) kiesik
(b_1, c_1) kiesik
(a_2, c_1) kiesik

{ -||- } 2. sorok esnek ki

Indukcióval belátható, hogy bármilyen $\ltimes$ program i-ik lépéssorozat után csak olyan sorok törlődhettek, amelyben az egyik tag indexe $\leq i$ vagy $n-i+1 \leq$.

[A végéről esnek ki a sorok, ha
$AC := AC \ltimes AB$ -vel kezdünk] kiesik (a_{n+1}, c_n)
$BC := BC \ltimes AC$ -||- (b_n, c_n)

Legyen egy $\ltimes$-program k utasítás. Legyen $n = 2k+2$

Ekkor k lépés után $(a_{k+1}\ b_{k+1})$ AB-ből nem törlődik
(b_{k+1}, c_{k+1}) BC -||-
$(a_{k+2}\ c_{k+1})$ AC -||-

$\}k$
a_{n+1-k} $b_{k+1+k+1}$
$\}k+1$

⇒ ∄ teljes reduktáló, mert függ a lépésszám a konkrét értékektől

Példa: mint előbb, de AC-ben (a_{n+1}, c_n) helyett (a_1, c_n)

Ekkor $AB \bowtie BC \bowtie AC = \emptyset$

viszont a $\ltimes$-k nem törölnek 1 sort sem!

Tétel: $R_1 \bowtie \dots \bowtie R_k$-nak $\exists$ teljes redukáló-$\ltimes$ program aesa, a hipergráf aciklikus

Bizonyítás: ciklikus $\Rightarrow$ $\nexists$ teljes redukáló-:

($\Rightarrow$) előző példát kell általánosítani

($\Leftarrow$) $k=1$ üres program teljes redukáló

$k>1$ és aciklikus $G \Rightarrow \exists$ S fül T-re

G-ből hagyjuk el S−T csúcsokat és S-t:

a kapott K aciklikus $\Rightarrow \exists$ teljes redukálója,

$T := T \ltimes S$

K teljes redukálója

$S := S \ltimes T$

teljes redukálója G-nek; mivel nem marad benne fityegő sor.

Hf: $s \in S$ (a programrész után) esetén s-hez kapcsolható $t \in T$, indukció miatt t-hez K minden relációjából kapcsolható sor $\Rightarrow$ s nem fityegő (ha még máshoz is kapcsolni kell, akkor a csak T [illegible] keresztül)

Elég: K-ban nem fityeg ⇒ G-ben sem fityeg

$T := T \ltimes S$ miatt minden $t \in T$ sor S-~~hez~~ beli s-hez és indukció miatt K több relációjának ~~a~~ valamelyik sorához kapcsolható.

K bármely relációjának u-sorához van a több relációban hozzákapcsolható sor, ezek között T-beli sor is szerepel, amihez S-beli sor kapcsolható, így u nem fityegő sor. □

Példa

AB fül BC-re — $BC := BC \ltimes AB$

BC fül CD-re — $CD := CD \ltimes BC$

CD 1 él nincs ütközés

$BC := BC \ltimes CD$

$AB := AB \ltimes BC$

Adott reláció redukáltjának meghatározása

$$\Pi_{R_i}(R_1 \bowtie \dots \bowtie R_k) = ?$$

Milyen fityegő sorokat kell R_i-ből elhagyni?

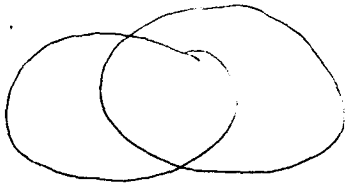
Tétel G aciklikus, E hiperél (nem feltétlen fül)
$\Rightarrow \exists$ olyan fülelvágási sorozat, hogy elfogy a gráf és E-t vágjuk le utoljára;

Biz.: indukció
1 él esetén triviális

TFH $n >$ hiperélek száma esetén igaz

Legyen G-nek n hiperéle

1) Ha E-n kívül van füle G-nek, akkor vágjuk azt le és indukció.
2) Ha E a G egyetlen füle:
belátjuk, hogy ez nem lehet, ha G-nek 1-nél több éle van



E F

$\exists F \quad E \cap F \neq \emptyset$
(különben G−{E} füle G-nek is füle volna)
1. eset

$K = G - \{E\}$ indukció miatt $\exists$ fülelvágás, hogy F az utolsó ékben. (lehet hogy van F-en kívül is fül?)
K-nak F-en kívül kell, hogy legyen éle, különben F G-nek füle lenne ↯ lenne
Legyen $G \neq F$ ~~éle~~ füle K-nak.
~~G~~ G egyedi csúcsai nem csúcsai F-nek $\supseteq F \cap E$
és ~~E-nek sem~~ E−F-nek sem, mert E fül,
tehát E−F egyedi, tehát $G \cap E = \emptyset$,
de akkor ~~K~~ G egyedi csúcsai nemcsak K-ban, de G-ben is egyediek, vagyis G füle G-nek ↯

Egy reláció-(R) redukáltjának meghatározása

- Vegyünk egy olyan GYO fülleválasztási sorozatot, amelyben R az utolsó.
- Vegyük a sorozatnak megfelelő $\ltimes$-programot
- Mivel R az utolsó, a program 2. felében R nem módosul, így ez a rész elhagyható.

1. fél
$$\left.\begin{array}{l} T := T \ltimes S \\ R := R \ltimes R' \end{array}\right\} \text{ez lesz az eredmény}$$

2. fél
$$\begin{array}{l} R' := R' \ltimes R \\ \underset{\text{fül}}{\underset{\uparrow}{S}} := S \ltimes \underset{\text{alapján}}{\underset{\uparrow}{T}} \end{array}$$

Példa: Vevő, Rendelés, Tartalmaz → 3 különböző helyen tárolják

Kik (név, cím) rendeltek mézet?

vcím, egyenleg, vnév, dátum, szám, cnév, mennyi

Vevő V — Rendelés R — $\sigma_{cnév='méz'}(\text{Tartalmaz})$ T

Kell Vevő redukáltja!

Fülleválasztás: jobbról balra

$$\left.\begin{array}{l} R := R \ltimes T \\ V := V \ltimes R \end{array}\right\} \text{eredmény } \Pi_{vcím,\, vnév}(V)$$

$$\begin{array}{l} R := R \ltimes V \\ T := T \ltimes R \end{array}$$

Példa: Mérsrendelések dátuma

(V-t továbbra is használjuk, hogy biztos tartozzon a rendeléshez vevő)

Most R az utolsó:

$$\left.\begin{array}{l} R := R \ltimes V \\ R := R \ltimes T \end{array}\right\} \text{Eredmény: } \Pi_{\text{dátum}}(R)$$

$$T := T \ltimes R$$
$$V := V \ltimes R$$

$\Pi_{A_1 \ldots A_n}(R_1 \bowtie \ldots \bowtie R_k)$ kiszámítása

---

Először a teljes redukáló költségét vizsgáljuk

Példa: A természetes összekapcsolás mérete a táblák számának exponenciális függvénye

- legalább exponenciális:

k-1 darab konstans (8 soros tábla):

A_i	A_{i+1}
1	2
1	4
2	1
2	3
3	2
3	4
4	1
4	3

$A_1 A_2 \bowtie \ldots \bowtie A_{k-1} A_k$

Az eredmény sorai az összes k hosszú alternáló sor:

$\underbrace{\begin{matrix} o\,e\,o\,e \dots \\ e\,o\,e\,o \dots \end{matrix}}_{k}$ $\quad$ o = páratlan (1, 3) $\quad$ e = páros (2, 4)

ilyen sorok száma: $2 \cdot 2^k = \underline{2^{k+1}}$ outputméret

inputméret $8 \cdot (k-1) = O(k)$

— kell még: több nem lehet (HF)

<u>Következmény</u> A köztes adatforgalommal (számítási költségével) nem várható, hogy a táblák számának kis függvénye legyen (még aciklikus esetben sem (példa aciklikus volt!))

Ki fog jönni, hogy <u>aciklikus</u> esetben az átviteli költség az <u>Input + Output</u> méret <u>kis fokú polinomja</u>.

<u>Példa</u>, Megjegyzés: $\exists$ 1 helyen végzünk mindent: átviteli költség helyett teljes számítási költség értendő.

Példa (ciklikus esetben nagy lehet a költség)

$A_1A_2 \bowtie A_2A_3 \bowtie \dots \bowtie A_{n-1}A_n \bowtie A_kA_1$ $\quad$ <u>k páratlan</u>

A_iA_{i+1} mint előbb (8 sor)

Ekkor 1) input méret $8 \cdot k = O(k)$

2) $\boxed{b}\ e\ \boxed{b}\ 4$ $A_k A_1$ -ben az a sorokban
$1\ 2\ 3\ 4$ különböző a pozíció
k

$\Rightarrow$ az eredmény üres output méret $= 0$

3) $R_i = R_i \ltimes R_j$ bármelyik két relációra
(mivel minden oszlopban $\{1,2,3,4\}$ szerepel

4) $2 \leq m < k$ bármely m reláció-összekapcsolása legfeljebb 2^{m+2} sort tartalmaz
(Előbbi példában $k-1$ $\ltimes$-ra jött ki 2^{k+1})
Ha nem egymás utániak, akkor $\ltimes$ helyett $\times$, és még több sor lesz.

5) Lehet, hogy félig-összekapcsolásokkal úgy redukáljuk (formailag) a relációkat, de a végül k (páratlan) relációt kell összekapcsolni.
Párosával összekapcsolva végül 2 táblát
$\underline{\qquad\qquad} \ltimes \underline{\qquad\qquad}$ kell összekapcsolni

Az egyik legfeljebb $\frac{k+1}{2}$ inputreláció-összekapcsolása: – $\ltimes$ nem változtat
– így a mérete legfeljebb

$$2^{\frac{k+1}{2}+2} = 2^{\frac{k+5}{2}}$$

input + output: $O(k)+0$ -nak exponenciális függvénye

Ezzel azt láttuk be, hogy ciklikus esetben,
ha csak $\bowtie$, és $\ltimes$ megengedett, akkor input + output méret exponenciális függvénye a költség.

~~Ebbő~~ Ez rendben, de ebből még nem következik, hogy ne létezne jobb algoritmus, ami esetleg mást is figyel (például $A_1 A_2 \bowtie \ldots \bowtie A_n A_1$ esetben semmit nem számol, hanem tudja, hogy $\emptyset$ az eredmény)

Tétel: Ha $\Pi_{A_1 \ldots A_n} (\underbrace{R_1 \bowtie \ldots \bowtie R_k}_{\text{tetszőleges gráf}})$ -ra létezne polinomiális (input + output méret) algoritmus (egyszerűbb esetre is!) akkor $P = NP$ következne (amiről azt sejtjük, hogy nem igaz).

* költséget csak számossal mérjük!

Aciklikus eset költsége*

$R_1 \bowtie \ldots \bowtie R_k$ esetén és fix G esetén

G – k hiperél, aciklikus

Teljes redukáló algoritmus 2(k−1) $\ltimes$-lépés

$F := F \ltimes E$ miatt $\Pi_{F \cap E}(E)$-t küldjük, mérete $\leq |E|$

$E := E \ltimes F$ miatt részeredményt küldjük vissza: méret $\leq |E|$

E ↑ fix, F ↑ alapján

Átviteli költség $\leq 2 \cdot \underbrace{(\text{inputrelációk méretösszege})}_{\text{inputméret}}$

fixköltség: $c_0 \cdot 2(k-1)$ elhanyagolható

ha G-t fixen tekintjük

1 helyi számítás költsége:

$\bowtie$ költsége legfeljebb $\bowtie$ költsége

– $R \bowtie S$ – rendezéses join ~~megfelelő módszer~~ esetén $\leq n \log n$

ahol $n = T_R + T_S$

[2 indexes join alapján:
$\approx O(n)$]

$2(k-1)$ darab $\bowtie$-képés számítási költsége:

$\leq O(k\, I \log I)$, ahol I az input mérete

Fix G (azaz k) esetén $\approx$ lineáris $(I \log I)$

Átviteli költség

Részeredményeket 1 helyre visszük:

$\leq I$

Át redukáltuk összehasonlítás egy helyen

$R_{i1} \bowtie \ldots \bowtie R_{ik}$ miatt (sorrend szerint) éppen U

éppen új feldolgozási sorrend: $R_{i1}, \ldots, R_{ik}$
(aciklikus) első utolsó

Összehasonlítás visszafelé

$R_i \bowtie (R_{i+1} \bowtie \ldots \bowtie R_k)$

A $\ltimes$-ok garantálják, hogy minden sorhoz legalább 1 sor kapcsolható

$$R_k := R_k \ltimes R_{k-1}$$

$$R_{h-1} := R_{h-1} \ltimes R_h$$

R_h nem módosul

így $R_{h-1} \bowtie R_h$

↑

minden sorához van kapcsolható sor

így $|R_{h-1} \bowtie R_h| \leq U$

$$|R_h| \leq U$$

indukcióval:

$$|R_i| \leq U \quad |R_{i+1} \bowtie \dots \bowtie R_h| \leq U$$

két tábla összekapcsolásának számítási költsége

$$\leq O(U \log U)$$

Teljes összekapcsolás költsége

$$O(k\,U \log U)$$

Kijött, hogy

Tétel: Aciklikus k reláció összekapcsolási költsége (átírási + számítási)

- $\leq O(k\,(I \log I + U \log U)) \leq O((k(I+U))^2)$

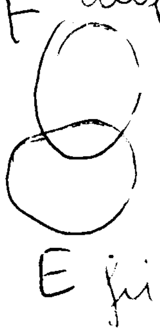
↑ inputméret ↑ outputméret

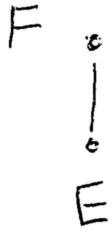
- az átírási költség $\leq O(I)$

$\Pi_{A_1,\dots,A_n}(R_1 \bowtie \dots \bowtie R_k)$ aciklikus

Fülkitépésnek megfelelő elemző fa:

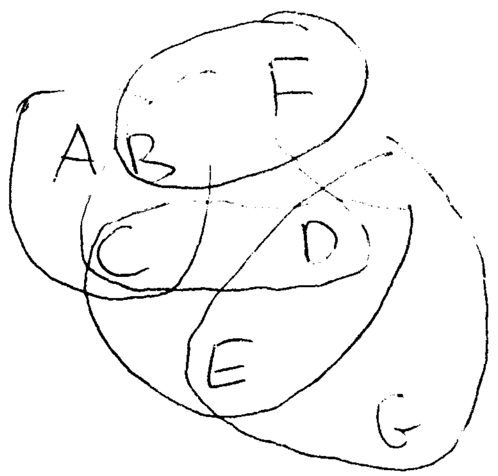
F alapján ha E-t kivágjuk F miatt, akkor

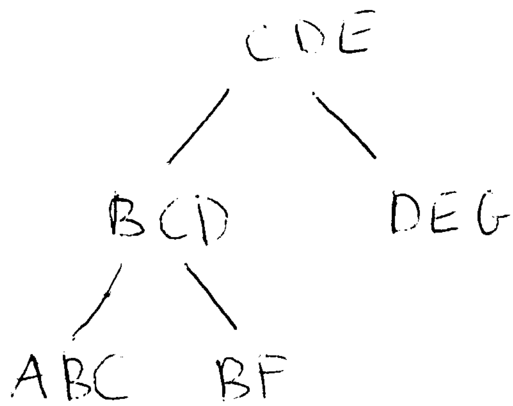




F gyermeke az E

Példa





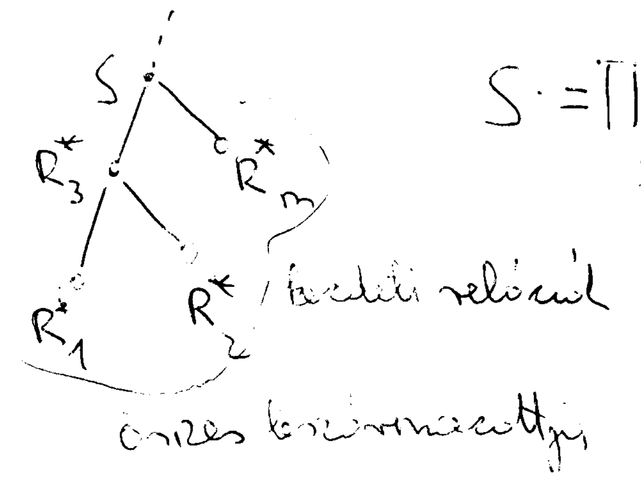
Yannakakis algoritmus

1. Redukáljuk R_i-ket egy teljes redukáló $\ltimes$ programmal
2. Készítsük el az elemző fát
3. A fülkivágás sorrendjében haladjunk az elemző fán végig (egy csúcs akkor jön, ha már minden gyereke sorrakerült)
   - ha R fület kivágjuk S miatt, akkor legyen $S := \Pi_{S\cup(X\cap R)}(R \bowtie S)$

4. Vetítsük a gyökérhez tartozó relációt X-re.
(Ezt az utolsó 3. lépéssel kell elvégezni:
P gyökér, R az utolsó fiú:

$$P := \Pi_X\left(\Pi_{P\cup(X\cap R)}(R \bowtie P)\right)$$

Megjegyzés: Belátható, hogy



$$S := \Pi_{S\cup(X\cap(R_1^*\cup\ldots\cup R_m^*))}\left(\overbrace{S \bowtie}\, R_1^* \bowtie \ldots \bowtie R_m^*\right)$$

↑ X-ből melyek szerepelnek a leszármazottak között

Példa: Előbbi példában

$$\Pi_{AG}(ABC \bowtie BF \bowtie BCD \bowtie CDE \bowtie DEG)$$

kezdeti relációk
↑
teljesen redukált

C	D	E
1	1	1
1	2	1

B	C	D
1	1	1
1	1	2

D	E	G
1	1	1
1	1	2
2	1	1

A	B	C
1	1	1
2	1	1

B	F
1	1
1	2

Kezdjük ABC-vel

$$ABCD := \prod_{\underbrace{BCD \cup \{AG \cap ABC\}}_{ABCD}} (ABC \bowtie BCD)$$

A	B	C	D
1	1	1	1
1	1	1	2
2	1	1	1
2	1	1	2

BF:

$$\prod_{ABCD \cup \{AG \cap BF\}} (BF \bowtie ABCD) = ABCD$$

$\emptyset$ mivel (ABCD ~~teljes~~ redukált

(Mindig kihagyható egy olyan gyereke X-a, amelyben nincs lényeges (X-beli) csúcs)

ABCD összekapcsolva CDE-vel:

$$ACDE := \prod_{\cancel{ABC}CDE \cup \{AG \cap \underbrace{A\cancel{BCDE}^{BCD}}_{A}\}} (A\cancel{CDE}^{BCD} \bowtie CDE)$$

1	1	1	1
1	1	2	1
2	1	1	1
2	1	2	1

DEG és ACDE összekapcsolás

$$ACDEG := \prod_{\cancel{DEG}^{ACDE} \cup \{AG \cap \underbrace{\cancel{ACDE}^{DEG}}_{G}\}} (\cancel{ACDE}^{DEG} \bowtie \cancel{DEG}^{ACDE})$$

1	1	1	1	1
1	1	1	1	2
1	1	2	1	1
2	1	1	1	1
2	1	1	1	2
2	1	2	1	1

Végül vetítés.

A	G
1	1
1	2
2	1
2	2

Belátjuk, hogy $I+U$-ban kvadratikus az algoritmus.

Lemma: Az algoritmus a 3. lépésben sosem készít [illegible] $2IU$ a-nál nagyobb relációt

Biz: A bejárt csúcsok száma szerinti indukcióval belátható, hogy egy S csúcs értéke mindig

$$\Pi_{S\cup Y}(S \bowtie S_1 \bowtie \dots \bowtie S_m) \text{ alakú,}$$

ahol S_i-k az S olyan C gyerekének leszármazottjai (nem feltétlen valódi), ~~amelyeket~~ hogy C-ben (és ezért az összes leszármazottjában) már jártunk,

és $Y \subseteq X$ az összes olyan lényeges attribútum, amely valamelyik S_i-ben benne van, de S-nek nem attribútuma.

Jelölje $T := S \bowtie S_1 \bowtie \dots \bowtie S_m$.

Ekkor $\Pi_{S\cup Y}(T) \subseteq \Pi_S(T) \times \Pi_Y(T)$ mivel $S \cap Y = \emptyset$

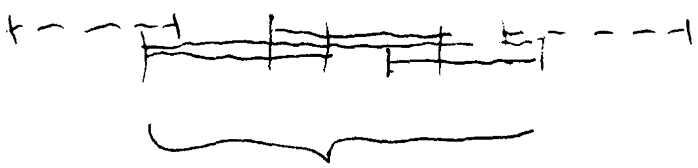
Kihasználva, hogy 3. lépésben a relációk redukáltak:

$$\Pi_S(T) = S$$

Emiatt $\Pi_S(T)$ mérete $=$ S mérete $\leq I$

Belátjuk, hogy $\Pi_Y(T)$ mérete $\leq U$:

1) $\Pi_Y(T) = \Pi_Y(R_1 \bowtie \dots \bowtie R_k)$, mivel a relációk redukáltak és ezért T minden sora kapcsolódik az R_i-k ~~minden sorát~~ legalább 1 sorához,



T

vagyis $\forall t \in T$ kiterjeszthető az $R_1, \dots, R_k$ összes attribútumára,

2. $Y \subseteq X$ miatt $\Pi_Y(R_1 \bowtie \dots \bowtie R_k)$ mérete $\leq$

$\Pi_X(R_1 \bowtie \dots \bowtie R_k)$ mérete $= U$

Kijött:

$$\Pi_{S \cup Y}(T) \subseteq \Pi_S(T) \times \Pi_Y(T)$$

mérete $\leq I$ mérete $\leq U$

$t_1 \cdot l_1$ $t_2 \, l_2$

sorok száma rekordméret

$$t_1 \cdot t_2 \underbrace{(l_1 + l_2)}_{\leq 2 l_1 l_2} \leq 2 t_1 l_2 \cdot t_2 l_2 \leq 2 \cdot I \cdot U$$

$l_1 = l_2 = 1$ miatt

$\square$

Tétel. Képzített $\prod_x (R_1 \bowtie \ldots \bowtie R_k)$ kiértékelésére az $\overbrace{\text{aciklikus}}$ algoritmus átviteli költsége és futási ideje polinomiális I, U, k paraméterekben

most $U = |\prod_X(R_1 \bowtie \ldots \bowtie R_k)|$ kisebb, mint az előbbi $U = |R_1 \bowtie \ldots \bowtie R_k|$

Bizonyítás:

1) Már láttuk, hogy a teljes redukáló alkalmazásának kommunikációs költsége $O(I)$

számítási költsége $O(k(I \log I + U \log U))$

2) Teljes redukáló (ellenőrző fa) megtalálási költsége $\leq$ lépések száma $\cdot$ csúcsok száma

$\uparrow$ belátható — $\uparrow$ k — $\uparrow$ $\leq I$ (ha nincs üres reláció)

$\uparrow$ ennek ellenőrzése $O(k)$

$\Rightarrow O(kI)$ elég a teljes redukáló és az ellenőrző fa megtalálásához, átviteli költség 0.

elhanyagolható $O(k(I \log I + U \log U))$-hoz képest

3) A csúcsok bejárása során összesen $k-1$ relációt kell továbbítanunk a szülőjük felé, lemma miatt ezek mérete $\leq 2IU$

$\Rightarrow$ összes átviteli költség $O(kIU)$

$|\# \geq O(I)$

$\Rightarrow$ teljes algoritmus átviteli költsége $O(kIU)$

Az összekapcsolások során ~~legfeljebb~~ minden (input output) tábla mérete $\leq$ 2IU,

rendezéses összekapcsolás miatt $O(IU\log(IU))$ összekapcsolásonként, így összesen

$O(kIU\log(IU))$

4) utolsó lépésnek nincs átviteli költsége
kiszámítási költsége $O(IU)$

Dominálótagokat véve

teljes átviteli költség $O(kIU)$
teljes számítási —— $O(kIU\log(IU))$