Optimalizácie pomocou redukcie hypergrafu

Ján Šturc

Jar, 2013

Reprezentácia datalógového pravidla hypergrafom

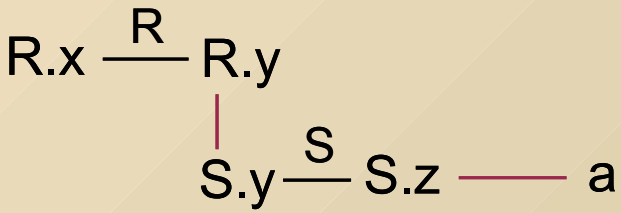
- Uvažujeme pravidlo $p \leftarrow q_1(x_{1,1} \ldots x_{1,k_n}), \ldots, q_n(x_{n,1} \ldots x_{n,k_n})$.
- Niektoré premenné môžu eventuálne byť rovnaké, alebo byť konštanty.
- Premenným a konštantám priradíme uzly hypergafu.
- Tie premenné a konštanty, ktoré sa vyskytujú v jednom predikáte (nenegovanom, alebo negovanom) spojíme hranou a pomenovanou príslušným predikátom (q, a ($\neg$q)) sú rôzne hrany.
- Hrany zodpovedajú DB predikátom konečným, alebo selekčným (zabudovaným, matematickým) potenciálne nekonečným a agregačným predikátom. Zo selekčných predikátov špeciálne vyčleňujeme rovnosť.
- Spolu 7 typov hrán.

Reprezentácia select príkazu hypergrafom

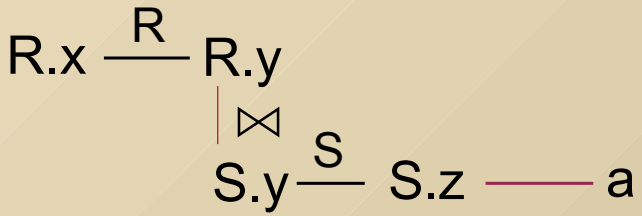
- V SQL je situácia jednoduchá. Všetky premenné sa vyskytujú v DB hranách zodpovedajúcích EDB predikátom a majú rôzne mená.
- Rovnosti premenných sú explicitne určené vo where (alebo join) klauzule.
- Selekčné podmienky zodpovedajú jednotlivým konjuktom (ktoré nie sú rovnosti premenných) where klauzy.
- Jednoduchý selekt neobsahuje iné hrany.
- Vnoreným selektom zodpovedá nový jednoduchý selekt. Výsledná relácia sa „joinuje“ (in, exists), alebo „antijoinuje“ (not in, not exist).
- Premenné vnorených selektov môžu byť prepojené s premennými hlavného dotazu, alebo iných vnorených dotazov selekčnými hranami (aj rovnosti).

Príklad.

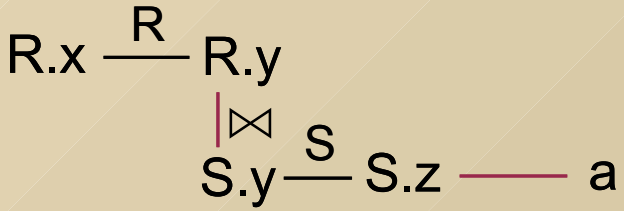
select R.x **from** R, S
where R.y = S.y and S.z = 'a'



select R.x **from** R
where R.y **in** (**select** S.y **from** S
where S.z = 'a')



select R.x **from** R
where exists (**select** * **from** S
where R.y = S.y
and S.z = 'a')



Ø ≡ false
{Ø} ≡ true

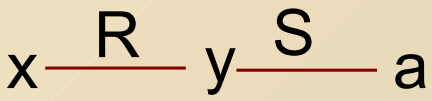
Nie celkom jasné prečo sa v SQL manuáloch tvrdí, že join je rychlejší (efektívnejší) ako vnorený dotaz ? Asi implementácie SQL prekladajú vnorený dotaz priamo na „nested loop“.

Transformácia na štandardný hypergraf.

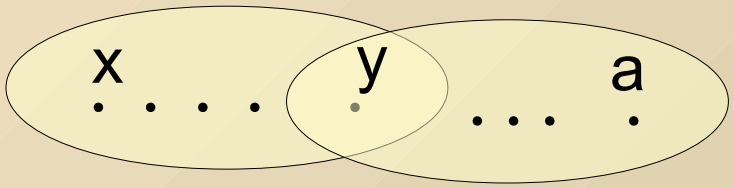
- V štandardnom hypergrafe sa nevyskytujú hrany rovnosti.
- Ich odstránenie je jednoduché. Ak sa rovnajú dve premenné, vyberieme jedneho reprezentanta a obe transformujeme do tohto reprezentanta.
- Ak sa premenná rovná konštante, nahradíme ju konštantou. Na hranu aplikujeme príslušnú selekciu.
- Pri náhrade premenných urobíme všetky hrany s nimi incidentné, incidentné s jedinou inštanciou ich reprezentanta.
- Ak sa v zápise vyskytuje join (kartézsky súčin) nahradíme ho spojením podľa príslušných premenných
- Ak sa jedná o zložitejšie vnorené dotazy, vyžaduje to často pomocnú hranu pre medzivýsledky.

Príklad – pokračovanie

Všetky tri dotazy na slide 4 sa transformujú na štandardný hypergraf:



Ak v reláciach R a S existujú aj iné atribúty:



$(\sigma_{z=a}S) \bowtie R$ alebo $R\ (\bowtie \circ \sigma_{z=a})\ S$

Optimalizácia jednoduchých dotazov (SPJ)

Budeme sa zaoberať dotazmi tvaru:

$$\Pi\,(\sigma_{F_1 \wedge \ldots \wedge F_m}(R_1 \times \ldots \times R_n))$$

Po našich transformáciach je to join:

$$\Pi\,(R_1 \bowtie \ldots \bowtie R_n \bowtie F_1 \bowtie \ldots \bowtie F_k)$$

Niektoré podmienky rovnosti pohltili joiny. Hypergraf obsahuje len dva typy hrán. Hrany zodpovedajúce pozitívnym EDB predikátom a hrany zodpovedajúce pozitívnym selekčným predikátom.

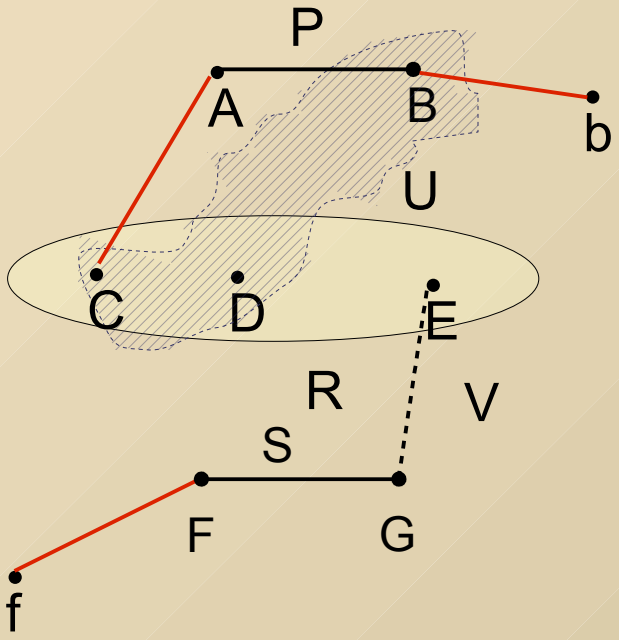
Nakoniec k takémuto dotazu pridáme záverečnú projekciu (select – from – where).

Príklad

Schéma: P(A,B), R(C, D, E), S(F, G)

select P.A, R.D, R.E, S.G **from** P, R, S

where ((P.A = R.C) **and** (P.B < R.C $\vee$ P.B < R.D)
and (S.G < R.E) **and** (S.F="f") **and** (P.B="b"));

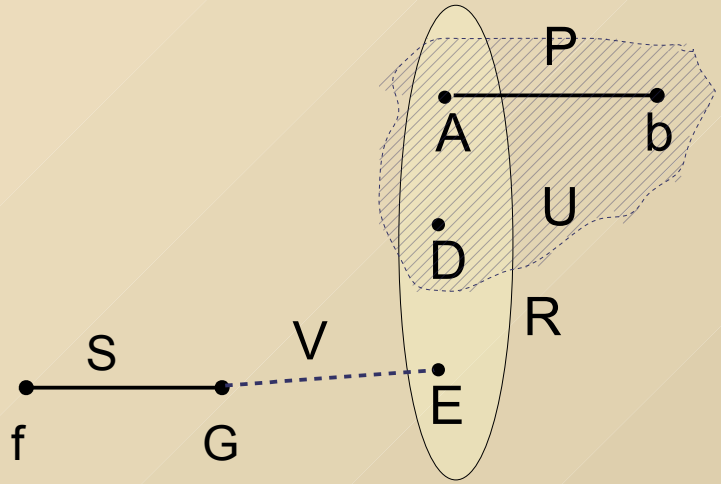


Príklad (pokračovanie)

P(A,B), R(C,D,E), S(F,G)

$W \equiv (A = C) \wedge \underbrace{(B < C \vee B < D)}_{U} \wedge \underbrace{(G<E)}_{V} \wedge (F=f) \wedge (B=b)$

Štandardný hypergraf:



Hrany:

$P \equiv \sigma_{B=b} P$

$R \equiv \Pi_{C \leftarrow A,D,E} R$

$S \equiv \sigma_{F=f} S$

U a V sú selekčné hrany

$U \equiv A > b \vee D > b$

Wong Youssefiho algoritmus
QUEL redukcia hypergrafu

1. Ak hypergraf $\mathcal{G}$ je zjednotenie viacerých „nesúvislých“ komponent $\mathcal{H}_1 \ldots \mathcal{H}_k$, potom
   - prog($\mathcal{G}$) je prog($\mathcal{H}_1$); …; prog($\mathcal{H}_k$);
   - res($\mathcal{G}$) := res($\mathcal{H}_1$)× … ×res($\mathcal{H}_k$)
2. Ak v hypegrafe $\mathcal{G}$ eliminujeme selekčnú hranu U a výsledný hypergraf je $\mathcal{H}$. Potom
   - prog($\mathcal{G}$) je prog($\mathcal{H}$);
   - res($\mathcal{G}$) := σ_U(res($\mathcal{H}$)) (res($\mathcal{H}$)⋈U)
3. Ak po odstránení relačnej hrany R sa hypergraf $\mathcal{G}$ rozpadne na k súvislých komponent (k=1 je možné.) $\mathcal{H}_1 \ldots \mathcal{H}_k$, potom
   - prog($\mathcal{G}$) = **for** each EDB edge S that intersect R **do** {S := S ⋉ R}; prog($\mathcal{H}_1$); …; prog($\mathcal{H}_k$);
   - res($\mathcal{G}$) := R ⋈ res($\mathcal{H}_1$)⋈ … ⋈ res($\mathcal{H}_k$)

Malé (redukované) relácie
poradie odstraňovania hrán

1. Reláciu R považujeme za malú, ak jej hrana obsahuje konštantu (selekcia tvaru x = a).
2. Reláciu R považujeme za malú potom, čo jej n-tice boli redukované semijoinom.

- Vieme si predstaviť aj iné dôvody pre považovanie relácie za malú, tie sa ale v tejto optimalizácii neuplatnia.

Projekcia významné uzly

- Intuitívne významné (zaujimavé) premenné (uzly) sú tie, ktoré sa vyskytujú vo výsledku a tie, ktoré potrebujeme pri spájaní a testovaní podmienok.

1. Ak $\mathcal{G}$ je počiatočný hypergraf dotazu, potom uzol N je významný práve vtedy, keď sa vyskytuje vo výsledku.
2. Ak hypergraf $\mathcal{H}$ vznikol z $\mathcal{G}$ elimináciou hrany E. Potom uzol $N \in \mathcal{H}$ je významný práve vtedy, ak bol významný v $\mathcal{G}$, alebo $N \in E$.
3. Ak $\mathcal{G}$ je zjednotenie nesúvislých komponent $\mathcal{H}_1 \ldots \mathcal{H}_k$. Potom uzol N je významný v $\mathcal{G}$ práve vtedy, ak je významný v niektorej z komponet $\mathcal{H}_i$.

Podmienky a preferencie pri eliminácii hrán

- Relačnú hranu môžeme odstrániť, ak pretína (má neprázdny prienik) s aspoň jednou inou relačnou hranou a nepretína žiadnu selekčnú hranu.
- Preferencie pri odstraňovaní relačných hrán:
  (a) Ak existuje kandidát „malá relácia“, vyradíme všetkých kandidátov, „ktorí nie sú malí“.
  (b) Ak niektorá zo zvyšných hrán rozbíja hypegraf na viac komponent súvislosti, vyradíme všetkých kandidátov, ktorí ho nerozbíjajú. (Preferencia multiway join.)
  (c) Zo zvyšných kandidátov si vyberieme ľubovolne.
- Ak nemôžeme odstrániť relačnú hranu musíme odstrániť selekčnú hranu. Vyberieme takú, aby sme po jej odstránení mohli použiť (a), ak taká neexistuje, tak aspoň (b). Ak ani taká neexistuje, uspokojíme sa s ľubovolnou selekčnou hranou.

Multiway join – prečo rozbíjači ?

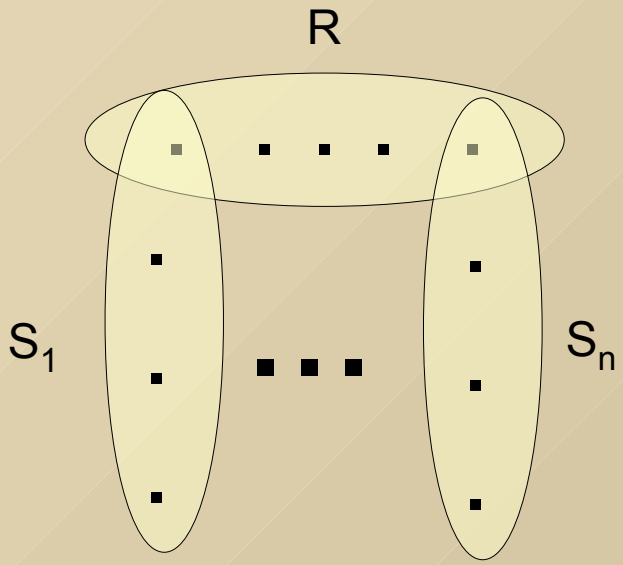
Treba vypočítať $R \bowtie S_1 \bowtie \ldots \bowtie S_n$, predokladáme, že:

- ak $i \neq j$, potom S_i a S_j nemajú spoločné atribúty a
- pre všetky i má S_i nejaký spoločný atribút s R.

for each tuple $\mu \in R$ **do**

{ **for** i := 1 **to** n **do** $T_i := S_i \ltimes \{\mu\}$;

output $\{\mu\} \bowtie T_1 \bowtie \ldots \bowtie T_n$ }



Jednoducho povedané. Pre n-ticu μ najprv určíme: s ktorými n-ticami sa spája.
Ak relácie majú vhodnú organizáciu (index), to môže byť rýchla operácia.
Ak sa s nejakou reláciou nespája, môžeme rovno prejsť na nasledujúcu n-ticu.

Wong Youssefiho algoritmus (QUEL)

Vstup: výraz tvaru $\Pi_D \sigma_F(R_1 \times \ldots \times R_n)$

Výstup: program (strom) na výpočet výrazu

Metóda: rekurzívna procedúra *EVAL* riadená úpravou a dekompozíciou hypergrafu. Zostrojíme štandardný hypergraf výrazu G (Konštrukcia transformuje vstupné relácie na podvýrazy zodpovedajúce hranám.) a zavoláme *EVAL(G,D)*.

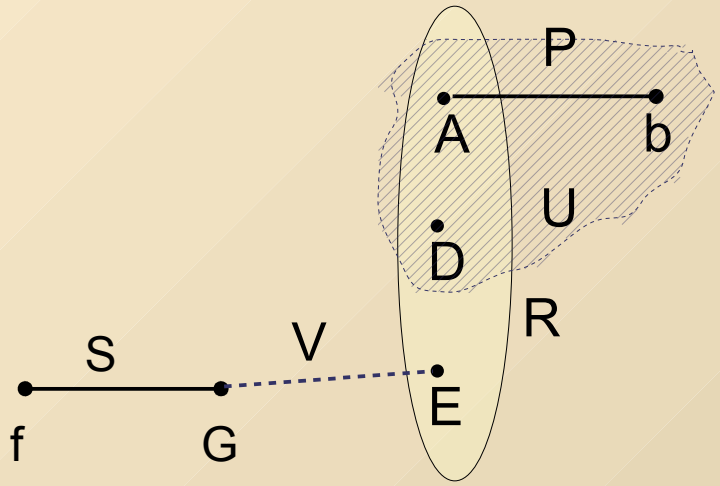
Procedúra *EVAL* si vyberá pre dekompozíciu hrany podľa pravidiel preferencie uvedených na predošlých slidoch.

Na výslednom strome ešte zlúčime binárne operácie s predchádzajúcimi unárnymi operáciami.

Zhustená schéma procedúry *EVAL*

- Prípad disjunktných komponent
  for i:=1 **to** k **do** { E_i:=nodes($\mathcal{H}_i$); *EVAL*($\mathcal{H}_i$,Ei) }
  emit "res($\mathcal{G}$) := res($\mathcal{H}_1$)× … ×res($\mathcal{H}_k$)"
- Odstránenie relačnej hrany R
  for each EDB edge S that intersect R **do** emit "S := S ⋉ R";
  – **for** i:=1 **to** k **do** { E_i:=(D∪R)∩nodes($\mathcal{H}_i$); *EVAL*($\mathcal{H}_i$,E_i) }
  emit "res($\mathcal{G}$) := Π_D(R ⋈ res($\mathcal{H}_1$)⋈ … ⋈res($\mathcal{H}_k$))"
- Odstránenie selekčnej hrany F
  E := (D∪F)∩nodes($\mathcal{H}$);
  EVAL($\mathcal{H}$,E);
  emit "res($\mathcal{G}$) := Π_D(σ_F(res($\mathcal{H}$))"

Príklad (dokončenie)



Hrany:

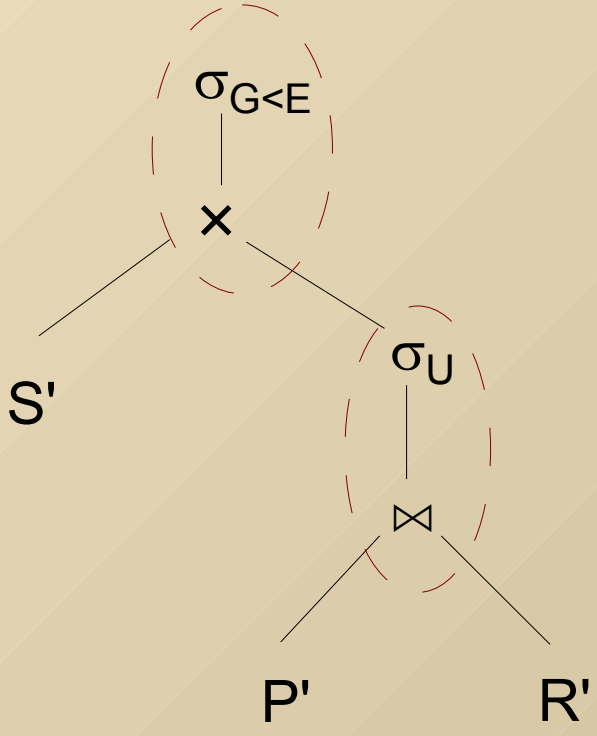
$P' \equiv \sigma_{B=b}P$

$R \equiv \Pi_{C \leftarrow A,D,E}R$

$S' \equiv \sigma_{F=f}S$

U a V sú selekčné hrany

$U \equiv A > b \vee D > b$



Program:

$P' := \sigma_{B=b}P;$

$S' := \sigma_{F=f}S;$

$R' := R \ltimes P';$

Result: $S'(\times \circ \sigma_{G<E})(P'(\bowtie \circ \sigma_{U})R')$

Veľkosť joinu

- Uvažujme join $A_1A_2 \bowtie A_2A_3 \bowtie \ldots \bowtie A_{n-1}A_n$.
- Nech každý atribút A_i má obor definície {1, 2, 3, 4}.
- Nech každá relácia pozostáva z ôsmych dvojíc takých, že každá relácia obsahuje všetky dvojice s nepárnym súčtom (tvaru <o,e> alebo <e,o>).
- Výsledok joinu pozostáva zo všetkých n-tíc $A_1A_2 \ldots A_n$ takých, že sa v nich striedajú nepárne a párne čísla. Spolu 2^{n+1} n-tíc.
- Hoci celková veľkosť n - 1 relácií na vstupe je $8\times(n-1)$.

Ak chceme rozumne oceniť zložitosť výpočtu joinu, musíme ju oceňovať v termínoch veľkosti vstupu aj výstupu.

Reduktory – úplný reduktor

- Je možné, že výpčet $R_1 \bowtie R_2 \bowtie \ldots \bowtie R_n$ bude prebiehať takým spôsom, že najprv budú medzivýsledky narastať a nakoniec sa zredukujú. Vstup malý, výsledok malý, ale veľká priestorová a výpočtová náročnosť.
- Reduktorom nazývame program pozostávajúci s postupnosti príkazov tvaru $R := R \ltimes S$.
- Reduktor slúži na redukciu počtu n-tíc, ktoré vchádzajú do výpočtu.
- Úplný reduktor je taký reduktor, ktorý zabezpečuje, že do výpočtu joinu nevôjde žiadna zbytočná n-tica.

Db verzia GYO redukcie – trhanie uší (ear removing)

Definícia: Hrana E hypergrafu $\mathcal{H}$ sa nazýva ucho, ak je jediná izolovaná hrana hypegrafu, alebo v hypergrafe $\mathcal{H}$ existuje, taká hrana F, svedok uchovitosti pre hranu E, že vrcholy E - F nie sú incidentné so žiadnou inou hranou hypergrafu $\mathcal{H}$.

Postupné trhanie uší, pokiaľ nezostane prázdny hypergraf je pre redukovateľné hypergrafy ekvivalentné GYO redukcii. Svedok uchovitosti F hrany E je tá hrana, korá pokryje (bude nadmnožinou) hranu E, ak z hrany E ostránime vrcholy, ktoré sú incidentné len s hranou E.

Bezstrátové spojenia

- $\Pi_{R_i}(R_1 \bowtie \ldots \bowtie R_n) = R_i$ (1) global consistency
- Pre každé i, j platí $\Pi_{R_i}(R_i \bowtie R_j) = R_i$ (2) local consistency

Veta: (1) $\Rightarrow$ (2).

Ak hypergraf $R_1 \bowtie \ldots \bowtie R_n$ je GYO redukovateľný, potom (2) $\Rightarrow$ (1).

Dôkaz:

Prvé tvrdenie dostaneme podľa pravidla 9 (slide 12) z predošlej prednášky dosadením výrazu $R_1 \bowtie \ldots \bowtie R_n$ za R aj S a položením $A=B=R_i$ a $C=R_j$.

Dôkaz druhého tvrdenia

Indukciou vzhľadom na počet relácii.

1. Pre n=1 a n=2 nie je čo dokazovať.
2. Predpokladajme, že program platí, pre n-1 relácií dokážeme, že platí pre n.

   Hypergraf je acyklický, teda v ňom existuje ucho E a svedok uchovitosti F. Po odstránení ucha E zvyšný hypergraf H obsahuje n-1 relácií. Medzi nimi aj reláciu zodpovedajúcu hrane F. Tieto relácie sa spájajú bezstrátovo.

   Každý riadok každej hrany (relácie) hypergrafu H sa podiela na výsledku.

   Hrany E a F sa spájajú bezstrátovo podľa predpokladu. Teda každá riadok E sa spája s nejakým riadkom relácie pre H. Atribúty, ktoré nie sú v F to nemôžu ovplyvniť, lebo sa vyskytujú iba v E.

Výpočet úplného reduktoru

- Znázorníme join hypergrafom
- Aplikujeme nasledujúcu rekurzívnu procedúru:

```
procedure reduce(G);
{ find an ear E with witness F;
  emit "F := F ⋉ E;";
  reduce(G-{E});
  emit "E := E ⋉ F;"
 }
```

Príklad - jednoduchý

A	B
1	2
2	4
3	6
4	8

B	C
1	2
2	4
3	6
4	8

C	D
1	2
2	4
3	6
4	8

Reduktor:

$AB := AB \ltimes BC;$

$BC := BC \ltimes CD;$

$CD := CD \ltimes BC;$

Úplný reduktor:

$BC := BC \ltimes AB;$

$CD := CD \ltimes BC;$

$BC := BC \ltimes CD;$

$AB := AB \ltimes BC;$

Pre acyklické joiny úplný reduktor obsahuje $2\times(k-1)$ príkazov priradenia a jeho generovanie je nezávislé od obsahu relácií.

Počas vyhodnotenia úplného reduktoru, žiadná relácia nerastie (často sa zmenší). Zložitosť vyhodnotenia úplného reduktoru je polynomiálna.

Príklady – cyklické

A	B
0	0
1	1

B	C
0	0
1	1

A	C
0	1
1	0

Každé dve relácie sa spájajú bezstrátovo.
Spojenie všetkých troch je prázdne.
Žiaden semijoin nič neredukuje.

A	B
1	1
2	2
...	...
n	n

B	C
1	1
2	2
...	...
n	n

A	C
2	1
3	2
...	...
n+1	n

1

Spojenie všetkých troch relácií je prázdne. Po i semijoinoch sa môžu vyredukovať iba dvojice obsahujúce čísla k ≤ i alebo k > n-i. „Úplný reduktor“ obsahuje n/2 príkazov priradení. Jeho veľkosť závisí od obsahu relácií.

Redukcia konkretnej hrany

Veta: Redukovateľný hypergraf s viac ako jednou hranou má aspoň dve uši.

Dôkaz: Indukciou vzhľadom na počet hyperhrán.

Pre n = 2 to platí. Trivialne. N = 3 preskúmanie všetkých možností.

Predpokladajme, že pre hypergrafy s menej ako n ≥ 3 hyperhranami to platí. Dokážeme, že to platí aj pre n. Nepriamo.

1. Nemá žiadne ucho spor s redukovateľnosťou.
2. Má jediné ucho E so svedkom uchovitosti F. Potom F je aj svedkom uchovitosti všetkým hranám, ktorým bolo svedkom uchovitosti E. Po odstránení E sa novým uchom, môže stať jedine F. To je spor s indukčným predpokladom.

Dôsledok: Môžeme si vybrať, ktorú hranu chceme odstrániť ako poslednú.

Yannakakisov algoritmus

Vstup: Relačný výraz $\Pi_X(R_1 \bowtie \ \ldots \bowtie R_k)$ s acyklickým hypergrafom G a relácie $R_1, \ \ldots, R_k$.

Výstup: Relácia hodnota výrazu $\Pi_X(R_1 \bowtie \ \ldots \bowtie R_k)$

Metóda: pozostáva zo štyroch krokov

1. Zbav sa zbytočných atribútov a redukuj relácie pomocou úplného reduktoru.
2. Zostroj dekompozičný strom P hypergrafu G.
3. Traverzuj strom P v nejakom poradí od spodu nahor. Ak relácia R je otec relácie S, urob $R := \Pi_{R \cup (X \cap S)}(S \bowtie R)$.
4. Vypočítaj záverečnú projekciu Π_X koreňa.

Spojenie prvých dvoch krokov – I. fáza

procedure reduce(G);
{ find an ear E with witness F;
  F := $\Pi_{Z \cap F}(F \ltimes E)$; /* Z sú zaujímavé atribúty */
  emit "hrana(F,E)"; /* F je otec E*/
  reduce(G-{E});
  E := $\Pi_{Z \cap E}(E \ltimes F)$;
 }

Z sú zaujímavé atribúty, je to množina atribútov výsledku X zjednotená s množinou Y tých atribútov, podľa ktorých sa relácie spájajú.
Výsledkom tohto algoritmu je dekompozičný strom a redukované (bezstrátovo sa spájajúce) relácie.

Post-order traverzovanie – II. fáza

procedure traverse(R, Z);
{ **for** i:=1 **to** k **do** { Y:= $R \cup (Z \cap S_i)$; traverse(S_i, Y) }
/* S_1, …, S_k sú synovia uzlu R */
R := $\Pi_{R \cup Z}(R \bowtie S_1 \bowtie \ldots \bowtie S_k)$ /* Z sú zaujímavé atribúty */
}

Hlavný program je:

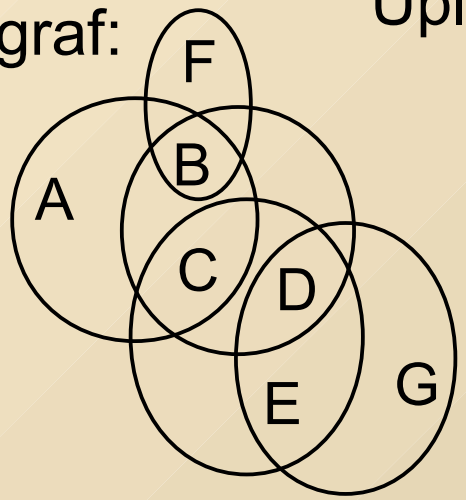
Traverse(root, root∪X);
$\Pi_X$root;

Výpočet joinu Yannakakisovým je polynomiálny vzhľadom na veľkosť vstupu a výstupu.

Príklad

$\Pi_{A,G}(ABC \bowtie BF \bowtie BCD \bowtie CDE \bowtie DEG)$

hypergraf:



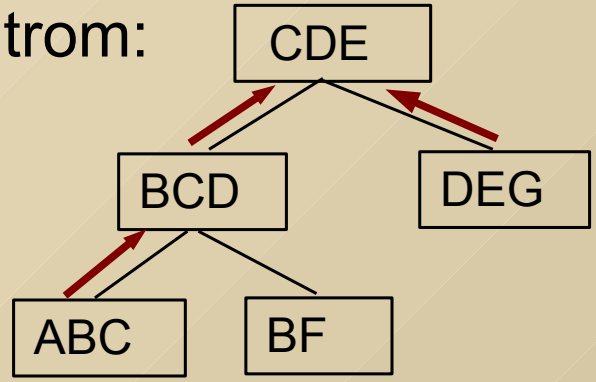
Úplný reduktor:

$BCD := BCD \ltimes ABC;$
$BCD := BCD \ltimes BF;$
$CDE := CDE \ltimes BCD;$
$CDE := CDE \ltimes DEG;$
$DEG := DEG \ltimes CDE;$
$BCD := BCD \ltimes CDE;$
$BF := BF \ltimes BCD;$
$ABC := ABC \ltimes BCD;$

Redukcia:

Ucho	Svedok
ABC	BCD
BF	BCD
BCD	CDE
DEG	CDE

Strom:

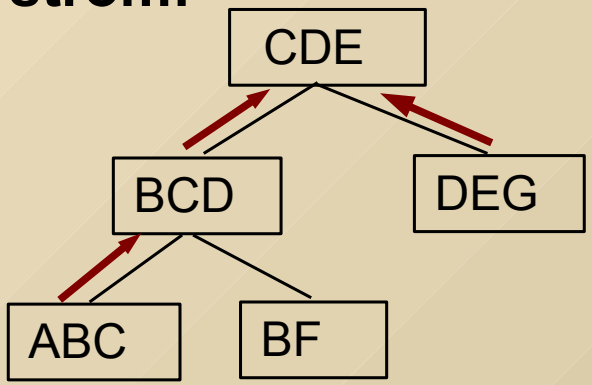


Príklad – pokračovanie

Naplnenie relácií po redukcii (priklad):

ABC	BF	BCD	CDE	DEG
$a_1b_1c_1$	b_1f_1	$b_1c_1d_1$	$c_1d_1e_1$	$d_1e_1g_1$
$a_2b_1c_1$	b_1f_2	$b_1c_1d_2$	$c_1d_2e_1$	$d_1e_1g_2$
				$d_2e_1g_1$

strom:



Výpočet:

ABCD	ACDE	ACDEG
$a_1b_1c_1d_1$	$a_1c_1d_1e_1$	$a_1c_1d_1e_1g_1$
$a_1b_1c_1d_2$	$a_1c_1d_2e_1$	$a_1c_1d_1e_1g_2$
$a_2b_1c_1d_1$	$a_2c_1d_1e_1$	$a_1c_1d_2e_1g_1$
$a_2b_1c_1d_2$	$a_2c_1d_2e_1$	$a_2c_1d_1e_1g_1$
		$a_2c_1d_1e_1g_2$
		$a_2c_1d_2e_1g_1$

Výsledok:

AG
a_1g_1
a_1g_2
a_2g_1
a_2g_2

Zovšeobecnenia

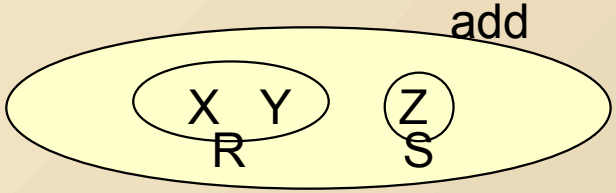
- Selekcie (zabudované predikáty)
  - doteraz pseudokľúč len všetky atribúty
  - využiť aj iné pseudokľúče
  - „zrovnoprávniť“ selekčné a relačné hrany
- Rozdiely
  - chýba niečo ako bezstrátovosť
  - skôr fungujúce heurestiky ako ucelená teória
  - zdá sa mi antijoin nádejnejší ako rozdiel
- Agregácie
  - mohlo by to byť zaujímavé (dátové kocky)

Plusy a úskalia prístupu selekcia je join.

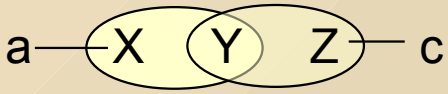
1

Dobré príklady:

$\sigma_{Z=x+y}(R(X,Y)\times S(Z)) = R(X,Y)\bowtie S(Z)\bowtie add(X,Y,Z)$



$\sigma_{X=a\wedge Z=c}(R(X,Y)\bowtie S(Y,Z)) = eq(X,a)\bowtie eq(Z,c)\bowtie R(X,Y)\bowtie S(Y,Z)$

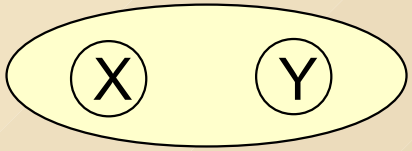


Plusy a úskalia prístupu selekcia je join.

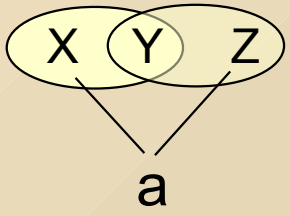
2

Zlé príklady:

$\sigma_{x<y}(R(X)\times S(Y)) = R(X)\bowtie S(Y)\bowtie less(X,Y)$



$\sigma_{X=a\wedge Z=a}(R(X,Y)\bowtie S(Y, Z)) = eq(X,a)\bowtie eq(Z,a)\bowtie R(X,Y)\bowtie S(Y, Z)$



Práca s nekonečnými reláciami

1. Všetkým inštanciam konštánt priraď rôzne uzly (aj keď niektoré nadobúdajú rovnaké hodnoty).
2. Počas redukcie s každou hranou eviduj dosiahnuté atribúty, sú to atribúty ktoré zodpovedajú viazným atribútom vo výpočte.
3. Na počiatku sú dosiahnuté atribúty hyperhrán:
   a) Všetky atribúty hrany pre konečné (EDB) relácie.
   b) Žiadné (prázdna množina) pre nekonečné relácie.
4. Vždy, keď sa hrana stane svedkom uchovitosti, pridaj k množine jej dosiahnutých atribútov, (zaujímavé) atribúty odtrhnutého ucha.
5. Ak je nekonečná relácia svedkom uchovitosti alebo uchom. Ucho možno odtrhnúť iba, ak zjednotenie dosiahnutých atribútov ucha a svedka uchovitosti pokrýva nejakú vstupnú množinu (dovolenú ozdobu) tejto nekonečnej relácie.

Neredukovateľné hypergrafy

- Wong Yussefiho algoritmus funguje
- Môžeme to eventuálne kombinovať. Vždy keď cyklus v hypergrafe spôsobuje selekčná hyperhrana. Eliminovať túto v štýle Wong Yussefiho algoritmu. Ak sa touto elimináciou hypergraf rozpadne na viac komponentov súvislosti optimalizovať každú komponentu vzlášť.
- Možeme ignorovať selekčné hrany optimalizovať len joiny a selekcie aplikovať, čo najhlbšie v strome.
- Nápadov je veľa, inžiniersky to aj funguje, ale nevieme dokázať „dobré vlastnosti“ algoritmov na redukovateľných grafoch.