Optimalizácie na úrovni relačnej algebry

Ján Šturc

Jar, 2013

Formy reprezentácie výrazov

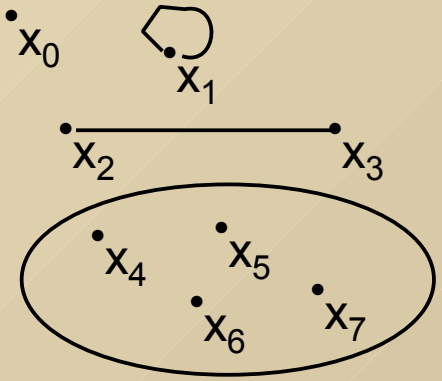
- Sú to algebraické výrazy klasická reprezentácia
  - stromy
    - syntaktický strom
    - strom výrazu
  - dagy (orientované acyklické grafy)
    - spoločné podvýrazy sa vyskytujú iba raz
- Pri relačnej algebre (videli sme to aj v datalógových optimalizáciach) nás veľmi často zaujímajú spoločné premenné (atribúty) výrazov a podvýrazov. Vhodná reprezentácia pre túto informáciu je:
  - hypergraf

Hypergrafy (hypergraphs)

Definícia:
Nech X je neprázdna množina a $\mathscr{E} = \{E_i\}_{i=0}^{m-1}$ je systém neprázdných podmožín X takých, že $X = U_{i=0}^{m-1} E_i$.
Dvojicu <X, $\mathscr{E}$> nazývame hypergrafom.
Prvky množiny X nazývame vrcholy (uzly) a množiny $E_0 \ldots E_{m-1}$ nazýme hyperhrany.
Ak pre všetky i < m, je $|E_i| = 2$. Je to obyčajný graf.

Kreslenie hypergrafov:
- Vrchol znázorňujeme bodom.
- Hranu kardinality 1, slučkou.
- Hranu kardinality 2, úsečkou.
- Všetky ostatné hrany oválom.

Často všetky hrany oválom.
Prípadne inou uzavretou čiarou.



Niektoré vlastnosti hypergrafov

- Úlohy, ktoré riešime pre grafy, sú často zaujímavé aj pre hypergrafy (súvislosť, cykly, kliky, farbenie).
- **Incidenčná matica** nech $X = \{x_0, \ldots, x_{n-1}\}$. Ku každému hypergrafu môžeme priradiť booleovskú maticu A typu $m \times n$ takú, že $a_{ij} = 1$ práve vtedy, keď $x_j \in E_i$.
- Naopak, každej booleovskej matici, ktorá má aspoň jednu jednotku v každom riadku a v každom stĺpci zodpovedá nejaký hypergraf.
- Nech H je hypergraf a s maticou A, potom hypergraf H*, ktorý zodpovedá A^T sa nazýva **duálnym** k hypergrafu H.
- Zrejme H** = H.

GYO redukcia hypergrafu (Graham, Yu a Ozsoyoglu)

- GYO redukcia je algoritmus postupnej dekompozície hypergrafu.

  Pokiaľ sa niečo dá ostrániť opakuj kroky:

  1. Ak nejaká hrana $E_i \subseteq E_j$, potom odstráň E_i.
  2. Ak pre nejaký vrchol $x \in E$ neexistuje hrana $E' \neq E$ taká, že $x \in E'$, potom odstráň vrchol x.

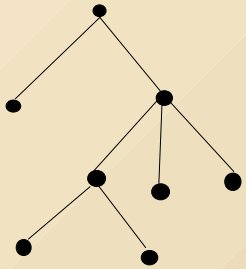
  Podľa definície hypergraf nemá prázdne hrany. V dôsledku toho pravidlo (2) likviduje izolované hrany.

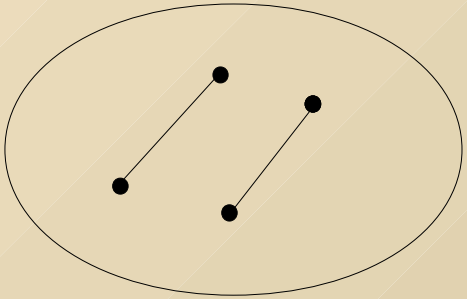
  GYO redukt (výsledok redukcie) je určený jednoznačne a postup má Church - Roserovú vlastnosť. (Nezáleží na tom, ktorý z možných krokov si vyberieme.)
  Hypergraf sa nazýva GYO redukovateľný, ak následkom GYO redukcie „zmizne“. (Výsledok je prázdna hrana.)
  V databázovej teórii sú to **acyklické** **hypergrafy**.

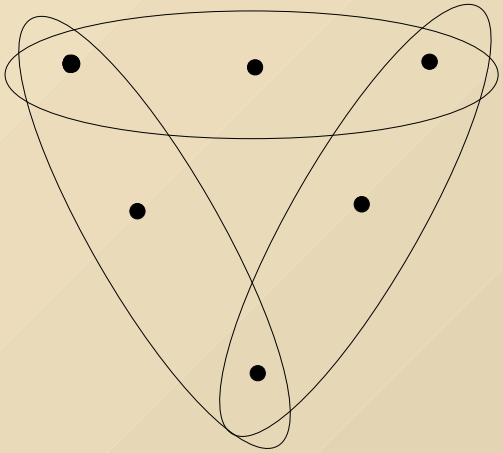
Trhanie uší

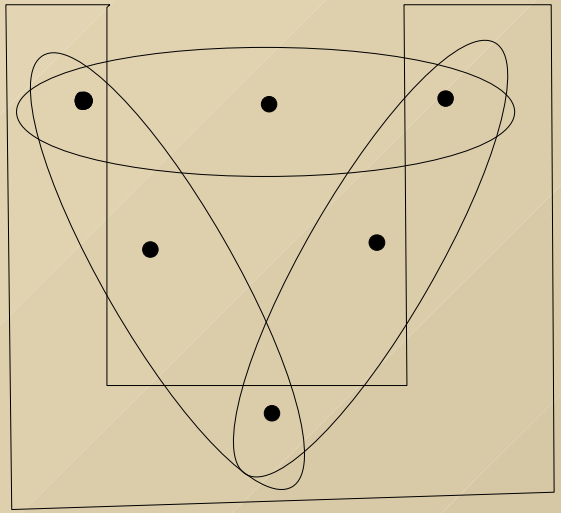
- Databázová podoba GYO redukcie
- Hyperhrana E hypergrafu $\mathcal{H}$ je ucho
  1. Ak E je izolovaná hyperhrana
  2. Ak existuje hyperhrana F (svedok uchovitosti) taká, že pre každý vrchol N $\in$ E – F platí, že nie je incidentný so žiadnou inou hyperhranou hypergrafu $\mathcal{H}$ okrem E.
- Algoritmus redukcie spočíva v postupnom odstraňovaní uší.
  - Hovoríme, že odstraňujeme ucho (hyperhranu) E v prospech svedka uchovitosti F.

Príklady









Algebra tabuliek versus algebra relácií

- Matematika
  - n-tica funkcia ***z usporiadanej*** množiny {1, …, n} do množiny argumentov (hodnôt).
  - Pri skladní n-tíc <a,b>,c> nie je to isté ako <a,<b,c>>.
- Databázy
  - n-tica (tuple) je funkcia z množiny mien (atribútov) ***neusporiadanej*** do množiny argumentov (hodnôt).
  - Pri skladní (join) n-tíc ich domény (obory definície) zjednocujeme.
  - Potreba technickej operácie premenovania (∏).
- Databázová relačná algebra nie je algebra relácií v matematike.

Zákony relačnej algebry 1

1. Join je komutatívny asociatívny a idempotentný
   - $R \bowtie S \equiv S \bowtie R$
   - $(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T)$
   - $R \bowtie R \equiv R$
2. Tie isté pravidlá platia pre zjednotenie
   - $R \cup S \equiv S \cup R$
   - $(R \cup S) \cup T \equiv R \cup (S \cup T)$
   - $R \cup R \equiv R$
3. Kartézsky súčin je join (vyžaduje disjunktné atribúty)
   - $R \times S \equiv S \times R$ (Nehovorte matematikom !)
   - $(R \times S) \times T \equiv R \times (S \times T)$
   - $R \times R$ (Kartézska mocnina, vyžaduje explicitné premenovanie atribútov)

Zákony relačnej algebry 2

4. Selekcie sú joiny (presnejšie prieniky s nekonečnými reláciami)
   - $\sigma_F(\sigma_G R) \equiv \sigma_G(\sigma_F R) \equiv \sigma_{F \wedge G} R$
   - Ak $F \Rightarrow G$, potom $\sigma_F(\sigma_G R) \equiv \sigma_F R$
5. Projekcia
   - Nech A a B sú množiny atribútov (premenných) a $A \subseteq B$, potom $\Pi_A(\Pi_B R) \equiv \Pi_A R$.
6. Selekcie a projekcie
   - Nech A je množina atribútov F podmienka. Označme $B = A \cup \text{atrib}(F)$. Potom $\Pi_A(\sigma_F R) \equiv \Pi_A(\sigma_F(\Pi_B R)$
7. Projekcia a zjednotenie
   - $\Pi_A(R \cup S) \equiv \Pi_A R \cup \Pi_A S$

Zákony relačnej algebry 3

8. Distributívne zákony
   - $R \bowtie (S \cup T) \equiv (R \bowtie S) \cup (R \bowtie T)$
   - $R \bowtie (S - T) \equiv (R \bowtie S) - (R \bowtie T)$
   - $R \bowtie (S \triangleleft\triangleright T) \equiv (R \bowtie S) \triangleleft\triangleright (R \bowtie T)$
   - $R \times (S \cup T) \equiv (R \times S) \cup R \times T$
   - $R \times (S - T) \equiv R \times S - R \times T$
   - $R \times (S \triangleleft\triangleright T) \equiv R \times S \triangleleft\triangleright R \times T$
   - $\sigma_F(R \cup S) \equiv \sigma_F R \cup \sigma_F S$
   - $\sigma_F(R - S) \equiv \sigma_F R - \sigma_F S$
   - $\sigma_F(R \triangleleft\triangleright S) \equiv \sigma_F R \triangleleft\triangleright \sigma_F S$

Zákony relačnej algebry 4

9. Projekcia a join: Nech R a S sú relačné výrazy. Nech $A \subseteq (atrib(R) \cup atrib(S)$, nech $B = (A \cup atrib(S)) \cap atrib(R)$ a $C = (A \cup atrib(R)) \cap atrib(S)$. Potom
   - $\Pi_A(R \bowtie S) \equiv \Pi_A(\Pi_B R \bowtie \Pi_C S)$
   - $\Pi_A(R \times S) \equiv \Pi_B R \times \Pi_C S$
10. Kombinácie selekcií a joinov: Nech R a S sú relačné výrazy. Nech F, G, H sú nekoneč predikáty také, že $atrib(F) \subseteq atrib(R)$, $atrib(G) \subseteq atrib(S)$ a $atrib(H) \subseteq atrib(R) \cup atrib(S)$. Potom
   - $R \bowtie S \bowtie F \bowtie G \bowtie H \equiv ((R \bowtie F) \bowtie (S \bowtie G)) \bowtie H$
   - $\sigma_{F \wedge G \wedge H}(R \bowtie S) \equiv \sigma_H(\sigma_F R \bowtie \sigma_G S)$
   - $\sigma_{F \wedge G \wedge H}(R \times S) \equiv \sigma_H(\sigma_F R \times \sigma_G S)$

Zákony relačnej algebry – agregácia

11. Agregácia a join
   - Ak B = A∩atrib(R) obsahuje nadkľúč v S a x ∈ atrib(R).
     $\Gamma_{A,q\leftarrow agg(x)}(R \bowtie S) \equiv \Pi_A((\Gamma_{B,q\leftarrow agg(x)}R) \bowtie S)$
12. Agregácia a projekcia
   - Ak A obsahuje nakľúč R a x ∈ A, potom:
     $\Pi_A(\Gamma_{B,q\leftarrow agg(x)}(R)) \equiv \Gamma_{A,q\leftarrow agg(x)}(R)$

Manipulácia s agregačnými výrazmi vyžaduje dodatočné informácie. Aspoň znalosť kľúčov resp. funkčných závislostí. Implementácie SQL preto väčšinou agregácie neoptimalizujú.

Dekompozícia selekčných predikátov

Pokiaľ je selekčný predikát v konjuktívnej forme zákony relačnej algebry ho umožňujú dekomponovať.

Čo s disjunkciou?

Príklady:

$$\sigma_{(x=a \vee y=b)}R(x,y) = \sigma_{x=a}R \cup \sigma_{y=b}R$$

$$\sigma_{(x=u \vee y=v)}R(x,y) \times S(u, v) = R \underset{x=u}{\bowtie} S \cup R \underset{y=v}{\bowtie} S$$

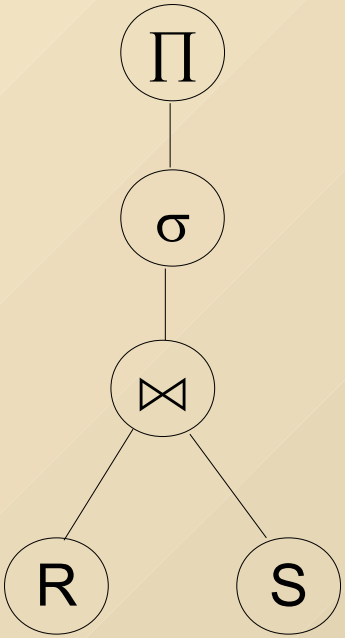
Relačná algebra ani SQL takéto pravidlá pri optimalizácii nepodporujú.

Pravidlá pre výpočet v relačnej algebre

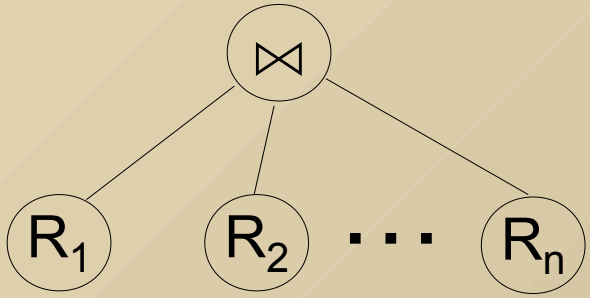
A. Vykonaj selekciu čo najskôr. Obzvlášť selekcie na rovnosť. Tieto pravdepodobne najviac zmenšujú výsledok. (10)
B. Vyhýbaj sa kartézskym súčinom. Nahraď ich joinami. Ak je aplikovaná selekcia na výsledok kartézského súčinu. Buď je to join, alebo selekcia môže byť posunutá na jednotlivé zložky kartézského súčinu.
C. Kaskáda unárnych operácii (selekcií, projekcií) sa dá spojiť do jednej unárnej operácie. Prípadne môžeme túto unárnu operáciu spojiť s predošlou binárnou operáciou.
D. Spoločné podvýrazy, ak ich veľkosť je rozumná, je výhodné si odložiť (prípadne aj do druhotnej pamäti). Ich zápis a čítanie môžu vyžadovať menej času ako opakované vyhodnotenie.

Strom výrazu

- Najjednoduchšie je znázorniť výrazy stromami.
- Vnútorné uzly reprezentujú operácie
- Listy relácie



V relačnej algebre sú unárne a binárne operácie. Faktor vetvenia 2.
Ak je binárna operácia asociatívna a komutatívna je niekedy výhodné použiť jeden uzol reprezentujúci všetky možné zoskupenia.



Algoritmus optimalizácie

1. Použi pravidlo (4) na separáciu selekcií do kaskády:
   $\sigma_{F_1 \wedge \ldots \wedge F_n}(E) = \sigma_{F_1}(\cdots(\sigma_{F_n}(E)\cdots)$
2. Pre každú selekciu použi pravidlá (6, 8 a 10), aby si ju „prepasíroval“ čo najblizšie k listom.
3. Použi pravidlá (5, 7 a 12) na „prepasírovanie“ projekcií čo najblizšie k listom. Pozor na alternáciu selekcií a projekcií! Eliminuj zbytočné projekcie.
4. Spoj späť za sebou idúce projekcie a selekcie znovu do jednej projekcie a jednej selekcie.

Algoritmus optimalizácie, pokračovanie

5. Rozdel vnútorne uzly stromu do skupín, nasledovne:
   - Každý vnútorný uzol reprezentujúci binárnu operáciu je v jednej skupine so svojími predchodcami označenými unárnou operáciou (Π, σ).
   - Každá skupina obsahuje aj postupnosť unárnych potomkov, končiacich listom.
   - Ak určujúci binárny uzol je kartézsky súčin (nie je predchádzaný vhodnou selekciou) jeho potomkov už do jeho skupiny nepridávame.
6. Výsledný plán môže počítať skupiny v ľubovolnom poradí, rešpektujúcom ich čiastočné usporiadanie v strome.

Príklad

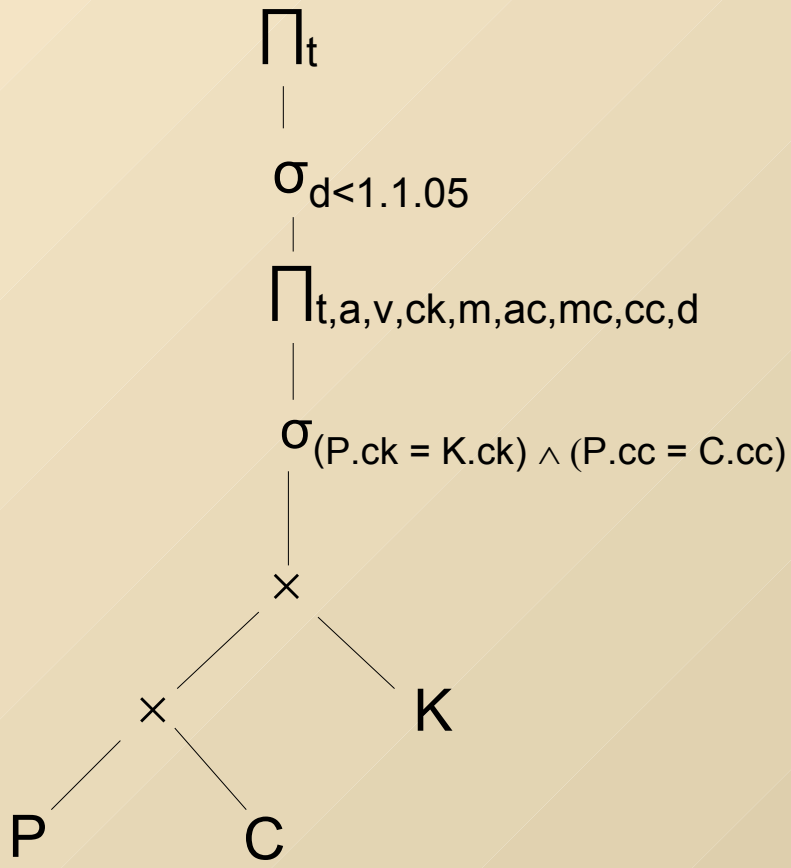
K(t, a, v, ck) – Kniha(titul, autor, vydadavateľ, číslo_k)
V(v, av, mv) – Vydavatelia(vydavateľ, adresa_v, mesto_v)
C(m, ac, mc, cc) – Čitateľ(meno, adresa_c, mesto_c, číslo_c)
P(cc, ck, d) – Požičal_si(číslo_c, číslo_k, dátum)

VL(t,a,v,ck,m,ac,mc,cc,d) ← P(cc,ck,d), C(m,ac,mc,cc), K(t,a,v,ck)
A(t) ← VL(t,a,v,ck,m,ac,mc,cc,d), d < 1.1.2005

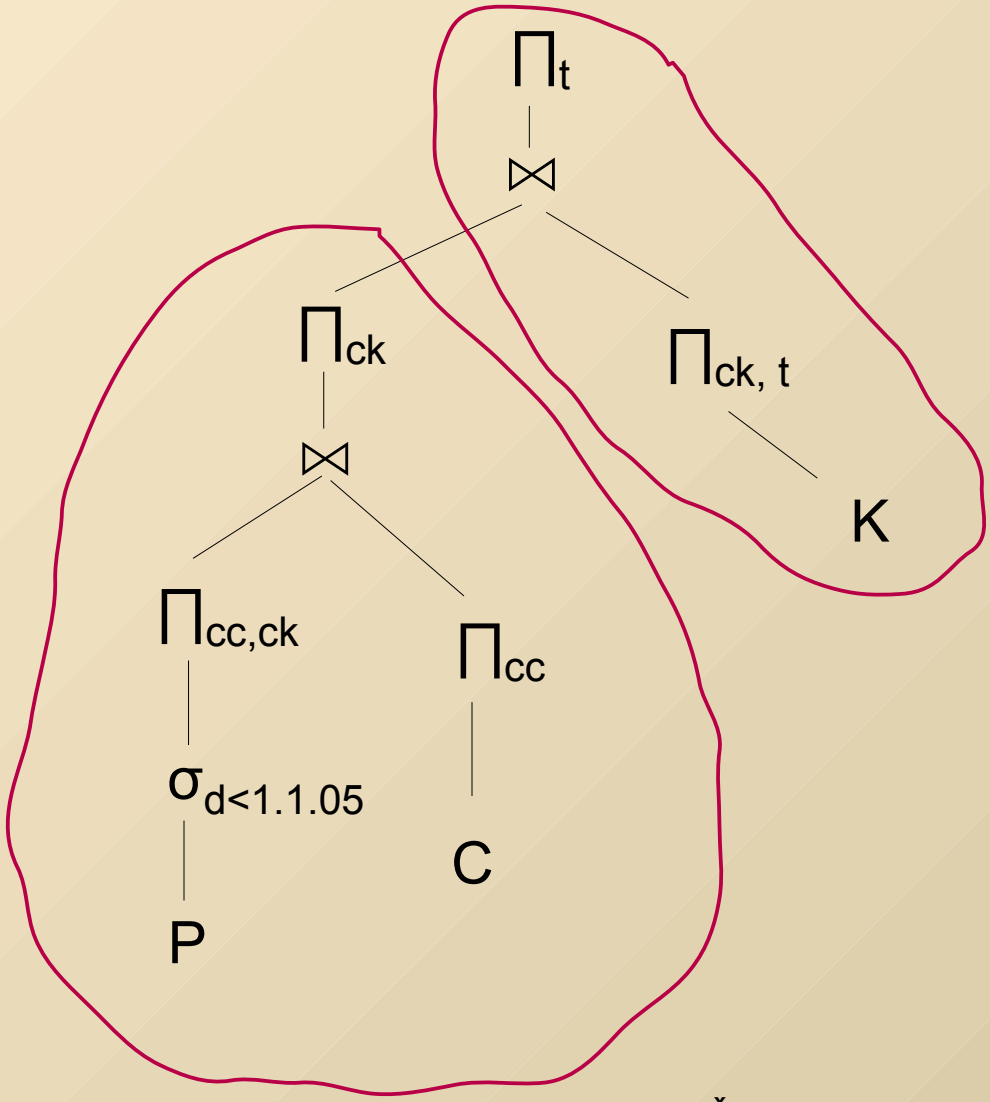
```
create view VL as select K.t,K.a,V.v,K.ck,C.m,C.ac,C.mc,C.cc,P.d
                         from P, C, T
                         where P.cc = C.cc and P.ck = K.ck;
select t from VL where d < 1.1.2005;
```

Na prvý pohľad sa to zdá nelogické (dotaz sa dá ľahko formulovať aj priamo), ale to view zodpovedá výpožičnému lístku a „tetušky“ z knižnice vytvoria dotaz cez grafický interface.

Príklad – strom dotazu



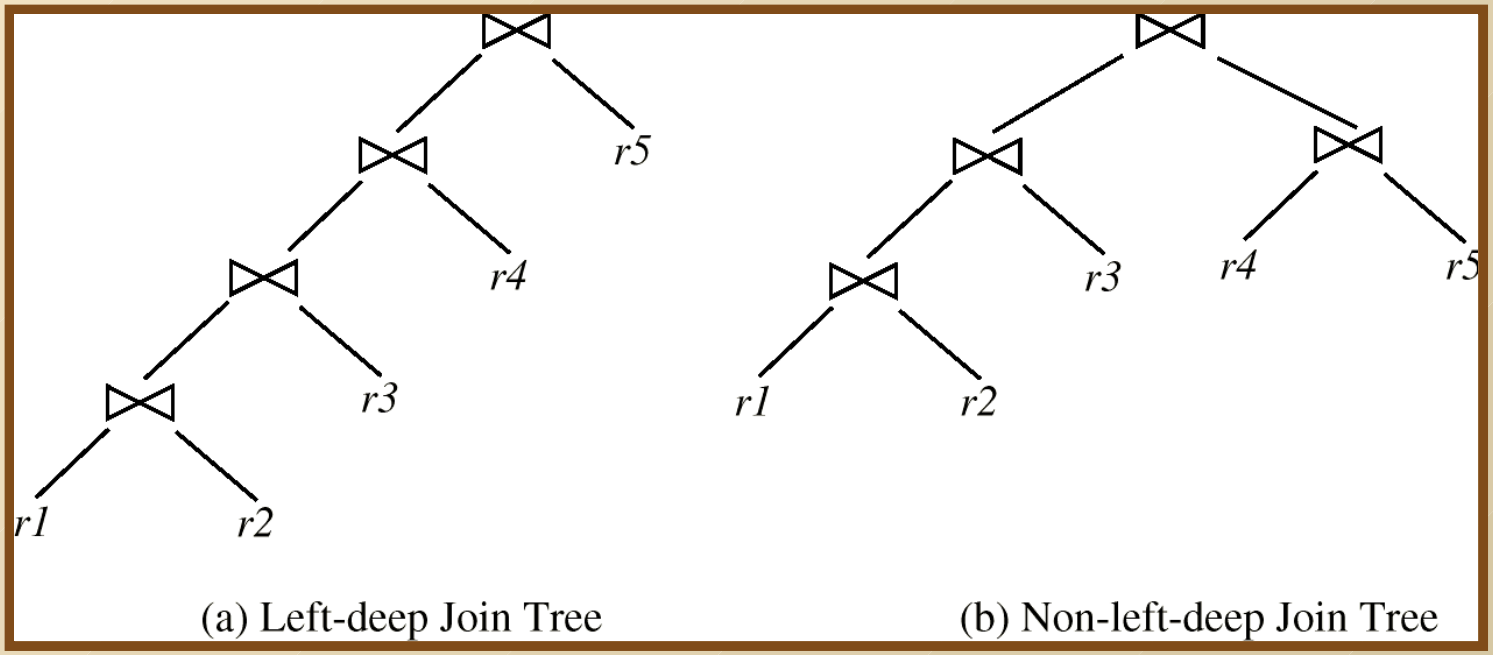
Príklad – po optimalizácii



Na prvý pohľad sa zdá, že vnorená komponenta sa dá zjednodušiť na: $\Pi_{ck}(\sigma_{d<1.1.05}(P))$.
Nie je to celkom pravda. Uvedené dotazy sú ekvivalentné iba v prípade, že databáza neobsahuje výpožičné lístky na neexistujúcich čitateľov.

Poradie (štruktúra) joinov

V ľavolineárnych (left-deep) a pravolineárnych spájacích stromoch (join trees) je vždy jeden operand EDB relácia a druhý výsledok predošlých joinov.



Počet usporiadaní joinov

Chceme nájsť najlacnejšie usporiadanie joinov

$$r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n.$$

Existuje $(2(n-1))!/(n-1)!$ rôznych usporiadaní (je to počet binárnych stromov s n rôznými listami). Pre $n = 7$, je ich 665 280. Pre $n = 10$, je to číslo rádove 10^{11} - 10^{12}.

(Všade to tak uvádzajú, ale pretože join je komutatívny asi treba výsledok deliť 2^{n-1}. Celé sa to dá odhadnúť pomocou Stirlingovho vzorca na $2^{n-1}(n-1)!/(\pi(n-1))^{1/2}$.)

To sa zdá byť prakticky nepoužiteľné, aj keby sme nebrali do úvahy iné operácie.

Našťastie, nie sme nútený generovať všetky binárne stromy. Použijeme dynamické programovanie.

Vypočítame cenu najlacnejšieho usporiadania pre každú podmnožinu $\{r_1, r_2, \ldots r_n\}$ iba raz a zapamätáme si ju.

Zložitosť je takto „len“ $O(2^n)$.

Dynamické Programming

Najdenie najlepšieho plánu na spojenie *n* relácií:
Aby sme našli najlepší plán joinu množiny *S,* pozostávajúcej z *n* relácií, budeme uvažovať všetky plány tvaru: $S_1 \bowtie (S - S_1)$, kde S_1 je neprázdna podmožina subset *S*.
Rekurzívne počítame ceny spájania podmnožín *S,* aby sme našli cenu každého plánu. Vyberieme najlepšiu (minimálnu) z $2^n - 1$ alternatív.
Aby sme sa vyhli zbytočnému počítaniu.
Zapamätáme si najlepší plán a minimálnu cenu pre každú už vypočítanú podmnožinu.

(Patricia G. Selinger, IBM project R)

Optimalizačný algoritmus – dynamické programovanie

procedure findbestplan(*S*);
{ if (*bestplan*[*S*].*cost* $\neq \infty$) **then return** *bestplan*[*S*]
else /* *bestplan*[*S*] has not been computed earlier, compute it now */
{ **for each** non-empty subset S_1 of *S* such that $S_1 \neq S$ **do**
{ P_1: = findbestplan(S_1)
P_2: = findbestplan(S - S_1)
A := algorithm for joining results of P_1 and P_2
cost = P_1.*cost* + P_2.*cost* + cost of *A*
if *cost* < *bestplan*[*S*].*cost* **then**
{ *bestplan*[*S*].*cost* = cost;
bestplan[*S*].*plan* = "execute *P*1.*plan*; execute *P*2.*plan*;
join results of *P*1 and *P*2 using *A*" }
}
return *bestplan*[*S*]
}
}

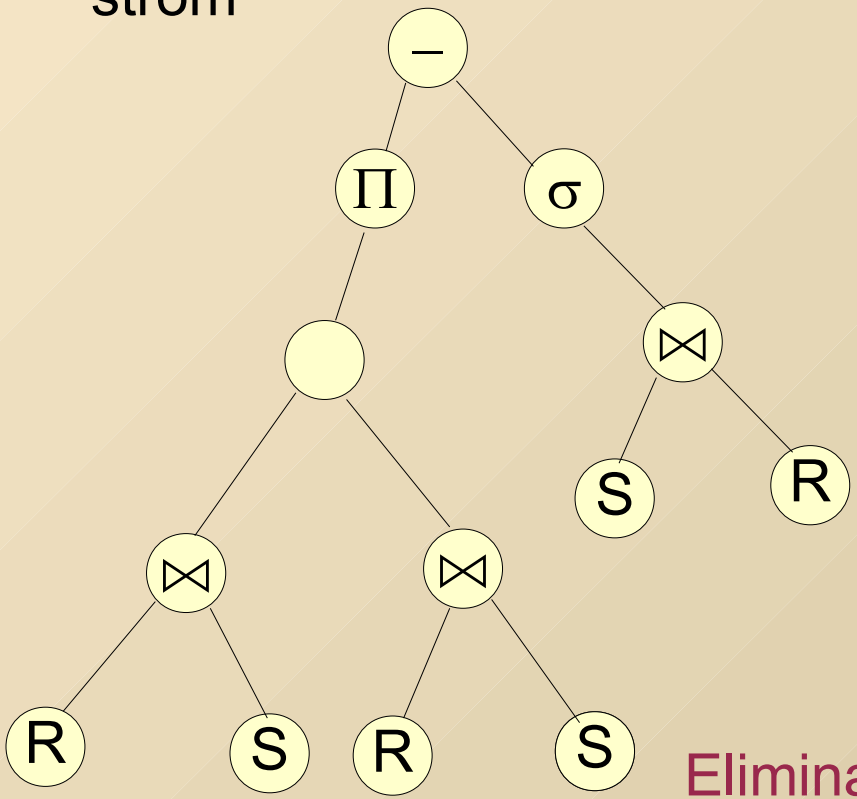
Kritika

- Mnohí považujú takúto optimalizáciu za pridrahú
- Navyše rozvetvené stromy vyžadujú pamäť pre medzivýsledky.
- Často sa preto používa len ľavo lineárny (pravo lineárny) výpočet.
- Poradie joinov sa vyberá „greedy“ metódou.

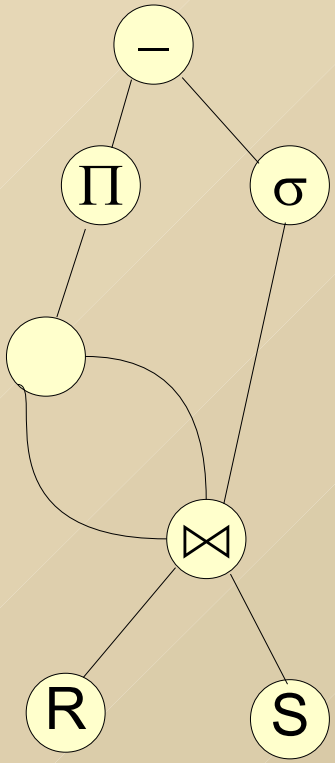
Výsledok takejto jednoduchej optimalizácie môže byť ľubovolne vzdialený od optima. V praxi je takýto jav dosť vzácny a často rozdiel nie je veľký. Možno sa vyplatí šetriť na optimalizácii.

Eliminácia spoločných podvýrazov

strom



dag



Eliminácia spoločných podvýrazov spočíva v transformácii stromu na dag (s minimálnym počtom uzlov).

Algoritmus eliminácie

Myšlienka algoritmu eliminácie je prekvapujúco jednoduchá a spočíva v opakovaní jediného cyklu.

repeat
find nodes with equal labels and the same successors;
/* be careful about the order of the successors */
make such nodes identical;
until no coincidence;

- Najprv sa stotožnia rovnako označené listy (majú rovnakých žiadných synov).
- Algoritmus môže eventuálne pokračovať, pretože už niektoré uzly majú tých istých synov.
- Ak uzly sú označené komutatívnou operáciou na poradí synov nezaleží. Inak sa synovia považujú za rôznych, aj keď sú v inom poradí.

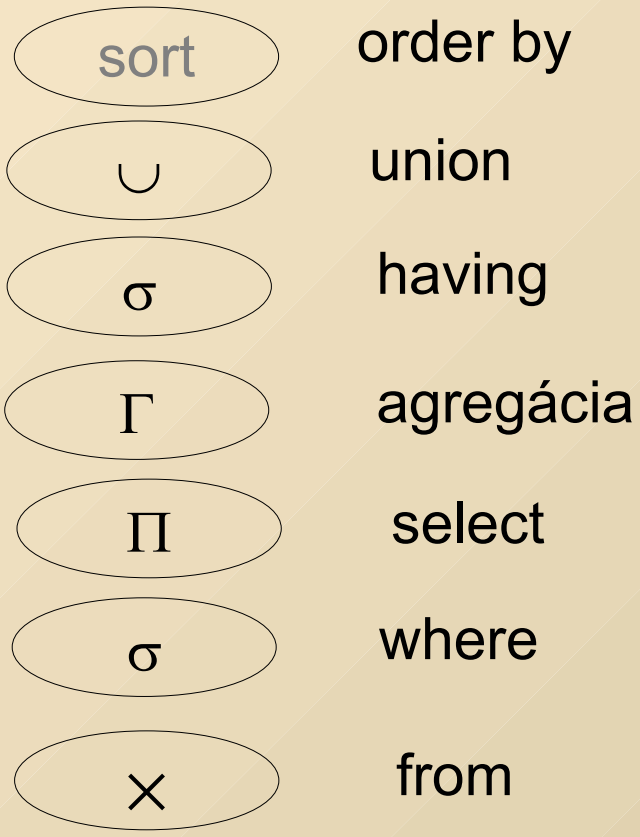
Detail algoritmu – stotožnenie uzlov

- Abstraktná (vizuálna) predstava je, že uzol sa bude posúvať tak, aby splynul s uzlom, s ktorým sa má stotožniť. Hrany sa pri tom deformujú.
- Implementačná predstava:
  - Node { id:_; label: operation; leftson: rightson: linktonode}
  - Zo stotožňovaných uzlov vyberieme uzol s najmenším id.
  - Všetky smerníky z rodičov stotožňovaných uzlov presmerujeme na tento uzol.
  - Uzly, do ktorých nevchádza žiadna hrana odstránime. S odstránením uzla sa odstránia aj hrany z neho vychádzajúce.

Syntéza

- Riešili sme jednotlivé úlohy
  - Presun unárnych zmenšujúcich operácii k listom.
  - Poradie komutatívny a asociatívnych operácií. Spojenia a kartézske súčiny. Pre zjednotenia je skoro jedno v akom poradí ich „nerobíme“.
  - Eliminácia spoločných podvýrazov.
- Optimalizáciu je vhodné robiť tak, že začneme so stromom, kde komutatívne a asociatívne operácie sú zlúčené do jedneho uzla. Ako prvé pretlačíme selekcie a projekcie, čo najbližšie k listom. Potom zlúčime selekcie a kartézske súčiny do joinov. Určíme poradie joinov a nakoniec eliminujeme spoločné podvýrazy.

Prirodzený strom pre SQL
(bez vnorených poddotazov)



V prípade vnorených poddotazov sa selekcie pre where a having nahrádzajú joinom resp. antijoinom, ktorého jedna vetva obsahuje pokračovanie dotazu a zvyšné vetvy vnorené poddotazy (okrem záverečného sortu). Proces vnárania môže rekurzívne pokračovať.
Ignorovali sme rozdiel (except), ale ten sa dá vždy nahradiť vnoreným poddotazom (not in / not exist). Mnohé implementácie SQL ho preto ani neobsahujú.

Redundancia relačnej algebry

- Relačná algebra je príliš „barokná“. Obsahuje priveľa operácií.
- Logickej spojke $\wedge$ (and) zodpovedajú až tri operácie:
  - Selekcia σ, ak sa jedná o zabudovaný (matematický) predikát.
  - Join $\bowtie$, ak sa jedná o DB predikáty so spoločnými premennými.
  - Kartézsky súčin $\times$, ak sa jedná o DB predikáty bez spoločných premenných.
- Vlastne štyri.
  - Antijoin $\triangleleft\!\triangleright$, ak sa jedná o DB predikát a negovaný DB predikát.

Minimalizácia relačnej algebry

- Dilemou je: či je lepšie mať veľa elementárnych operácii, alebo málo všeobecných.
- Obe rozhodnutia majú svoje výhody a nevýhody.
- Pokus o málo všeobecných:
  - Π, $\bowtie$, $\triangleleft\triangleright$, Γ, $\cup$
  - Join používame tak, že v prípade disjunkných množín atribútov (premenných) operandov je to kartézský súčin, v prípade rovnakých množín atribútov prienik, inak prirodzené spojenie.
  - Antijoin je len o málo všeobecnejší než rozdiel. $R \triangleleft\triangleright S = R - (R \ltimes S)$.
  - Ostatné operácie zostávajú ako v relačnej algebre.

Iné modifikácie relačnej algebry

- Joiny
  - ľavý, pravý ľavopravý join
  - Θ join. Symbolicky $R \bowtie_\Theta S = (R \bowtie S) \bowtie \Theta$
- Relácie ako multimnožiny (duplikáty)
  - fuzzy relácie
  - pravdepodobnostné relácie
  - Databases with uncertainty and lineage (Omar Benjelloun, Anish Das Sarma, Alon Halevy, Jennifer Widom)

Ohraničenia

Zovšeobecnené relácie vyžadujú isté ohraničenia. Tieto ohraničenia sa týkajú väzby premenných. V datalógu museli byť formuly bezpečné. V relačnej algebre a SQL sme zabudované (matematické) predikáty používali formou selekcie. Táto požaduje, aby všetky premenné, ktoré sa vyskytujú v selekčnom predikáte boli viazané. Ten istý požiadavok je kladený aj na rozdiel a agregačnú funkciu. Z hľadiska optimalizácie sú tieto obmedzenia možno zbytočne silné a bránia niektorým optimalizáciam.

Príklad: $\sigma_{z=x+y}(R(x,y)\times S(z))$; $(R(x,y)\bowtie S(z))\bowtie add(x,y,z)$

Poradie: $(R(x,y)\bowtie add(x,y,z))\bowtie S(z)$ bolo zakázané. Je to však dobre realizovateľný a v pripade, že relácia R obsahuje niekoľko málo dvojíc a relácia S je veľká, je to optimálny spôsob vyhodnotenia uvedeného výrazu.

SQL:
```
with sup(x,y,z) as select R.x, R.y, x+y as z from R
select sup.x, sup.y, sup.z from sup, S where S.z = sup.z ;
```

Predpoklady (výpočtový model)

Budeme predpokladať, že pre každý predikát v databáze sú definované minimálne povolené ozdoby. To je to isté ako minimálne podmnožiny atribútov (budeme ich nazývať pseudoklúče, input sets), pre ktoré v danej relácií existuje len konečný počet riadkov a databázový stroj ich vie efektívne vrátiť.
Príklady:
Pre EDB relácie je pseudoklúč Ø. (full scan)
Pre eq(x,y) ≡ x=y sú to {x}, {y}.
Pre zmienený predikát add(x,y,z) ≡ z=x+y to môžu byť všetky dvojice atribútov.
Predpokladáme, že v najhoršom je pseudokľúčom množina všetkých atribútov.

Uskutočniteľnosť operácií

- Unárne operácie

Π – projekcia vyžaduje aspoň jeden pseudokľúč operandu.

Γ – agregácia vyžaduje, aby grupovacie atribúty obsahovali pseudokľúč operandu.

- Binárne operácie

∪ – zjednotenie, bezpodmienečne aspoň jeden pseudokľúč pre každý operand.

⋈ – join, pseudokľúč pre každý operand v čase spájania.

◁▷ – antijoin aspoň jeden pseudokľúč prvého operandu. Druhý operand musí byť na základe spoločných atribútov aspoň rozpoznávač.

Modus operandi

Budeme predpokladať, komponentu výrazu vyhodnocujeme takým spôsobom, že si v nejakej postupnosti pomocných relácií zbierame čiastočné výsledky. Poradie operácií je len čiastočne určené, rsp. môžeme aplikovať zákony relačnej algebry. Zaujíma nás, kedy môžeme, ktorú operáciu použiť. Určujú to premenné, ktoré už vieme – viazané premenné.

Vlastne pre každý predikát máme určené minimálne dovolené ozdoby. Je to zaujímavé pre zabudované a negované predikáty. Pretože, „kľúč“ je tu skôr odraz úmyslu programátora než reálny kľúč relácie. Tie operácie, ktoré ho požadujú dovolujeme volať len pre predikáty s ozdobou b…b.

Znázornenie hypergrafom

- Grafy len operácie a operandy. Nedokážeme sledovať premenné a väzby.

Hypergraf

- Uzly zodpovedajú premenným
- Hrany zodpovedajú predikátom
  - nenegovaným DB predikátom
  - zabudovaným (matematickým) predikátom
  - predikátu rovnosti
  - agregačným predikátom
  - negovaným predikátom