FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
UNIVERZITA KOMENSKÉHO
KATEDRA INFORMATIKY

Optimalizácia relačných dotazov s agregačnými funkciami

(Revízia metód a algoritmov)

Diplomová práca

Martin Labanc
dipl. vedúci RNDr. Ján Šturc

Bratislava apríl 2005

Poďakovanie

Touto cestou vyslovujem poďakovanie RNDr. Jánovi Šturcovi za poskytnutie literatúry, odborné vedenie, cenné rady a pripomienky pri vypracovaní mojej diplomovej práce.

Čestne vyhlasujem, že diplomovú prácu som vypracoval samostatne a že som uviedol všetku použitú literatúru.

.....

V Bratislave, apríl 2005 Martin Labanc

Obsah

1 Úvod

Relačné dotazy obsahujúce agregačné funkcie majú vzhľadom na objemy dát v databázach v praxi veľký podiel na celkovom čase spracovania údajov. Mnoho testov výkonnosti - benchmarkov (napr. TPC-D) je založených práve na meraní času vyhodnotenia dotazu s agregačnou funkciou. Je preto na mieste zaoberať sa prevodom týchto dotazov do čo najoptimálnejšieho tvaru.

Predmetom nášho záujmu budú nerekurzívne dotazy formulované v revidovanej relačnej algebre obohatenej o operáciu agregácie. Pri skúmaní vlastností relačných dotazov sa budeme snažiť niektoré operácie zovšeobecniť a poskytnúť tak odlišný pohľad na ich vyhodnotenie. Tieto úvahy sa budú týkať predovšetkým operácie selekcie, ktorú nahradíme zovšeobecneným joinom a agregácie, ktorej záber rozšírime aj na projekciu.

Dotkneme sa aj problému nekonečných relácií, ich konečnej reprezentácie a operáciami nad nimi, nakoľko nahradením selekcie joinom mnohokrát vyvstane potreba relácie, ktorá reprezentuje určitú matematickú operáciu.

Úvahy o optimalizácii dotazov nebudú využívať platformovo špecifické implementačné detaily a budú čo najviac všeobecné. Vystačíme si pri nich s pojmami klasickej databázovej teórie.

Nakoniec zhrnieme teoretické výsledky do množiny pravidiel resp. odporúčaní ako optimálne vyhodnocovať relačné dotazy s agregačnými funkciami. Ďalej navrhneme rule-based optimalizátor realizujúci tieto optimalizačné pravidlá. Uvedieme aj motivačné príklady ilustrujúce využitie jednotlivých transformácií.

2 Modifikovaná algebra

2.1 Definície, základné vzťahy

V prvom rade uvedieme definície niektorých špecifických pojmov, ktoré budeme ďalej používať. Na definovanie operácií revidovanej relačnej algebry využijeme relačný kalkul. V ďaľších úvahách budeme potom pracovať len s pojmami modifikovanej algebry. Relácie reprezentujeme zodpovedajúcimi tabuľkami, ktoré chápeme ako multimnožiny (bags). V každej tabuľke implicitne uchovávame osobitný atribút m, ktorý obsahuje záznam o násobnosti daného riadku. Pri dosiahnutí násobnosti 0 riadok v tabuľke prestáva existovať. Kvôli jednoduchosti tento atribút v relačnej schéme nezobrazujeme. *Veľkosť* relácie je vyjadrená počtom riadkov tabuľky, ktorá ju reprezentuje. Pre zjednodušenie úvah predpokladáme, že tabuľky obsahujú hodnoty rôzne od $NULL$. V nasledujúcich definíciách zodpovedá relácii $R_n(a_1, a_2, ..., a_k)$ predikát $P_n(x_1, x_2, ..., x_k) \Leftrightarrow (x_1, x_2, ..., x_k) \in R_n$.

Definícia 2.1.1 (Zjednotenie). $R_1(x) \cup R_2(x) = \{x : P_1(x) \vee P_2(x)\}$.
Násobnosti jednotlivých riadkov sa aktualizujú sčítaním násobností záznamov jednotlivých tabuliek: $(R_1 \cup R_2).m := R_1.m + R_2.m$.

Definícia 2.1.2 (Prienik). $R_1(x) \cap R_2(x) = \{x : P_1(x) \wedge P_2(x)\}$.
Násobnosť záznamu sa rovná menšej násobnosti riadku vo vstupných tabuľkách: $(R_1 \cap R_2).m := min(R_1.m, R_2.m)$.

Definícia 2.1.3 (Rozdiel). $R_1(x, y) - R_2(x) = \{x : P_1(x, y) \wedge \neg P_2(x)\}$.
Násobnosť uchovávame len pre záznamy, ktoré patria rozdielu. Ostatné vo výsledku nefigurujú: $(R_1 - R_2).m := R_1.m$

Formuly jazyka relačného kalkulu sú tvorené analogicky ako v jazyku prvého rádu :

- Termami sú konštanty a atribúty relácií.
- Atomické formuly vzniknú aplikáciou binárnych operátorov $<, \leq, =, \neq, >, \geq$ a matematických operácií na termy.
- Zložené formuly tvoríme spájaním atomických formúl pomocou logických operátorov $\neg, \wedge, \vee$.
- Iné formuly sa v tomto jazyku nenachádzajú.

Definícia 2.1.4 (Selekcia)**.** Operátor selekcie vyberie z relácie riadky vyhovujúce selekčnej podmienke,ktorá je vyjadrená formulou F. $\sigma_F R(x_1, x_2, ..., x_n) = \{x_1, x_2, ..., x_n | P(x_1, x_2, ..., x_n) \wedge F\}$.

Definícia 2.1.5 (Projekcia)**.** Operácia projekcie Π_x vyberie z relácie určitú podmnožinu atribútov. $\Pi_x R(x, y) = \{x : (\exists y) P(x, y)\}$.

Definícia 2.1.6 (Kartézsky súčin)**.** Najdrahšia operácia algebry operujúca nad dvomi tabuľkami, vytvorí reláciu s mohutnosťou rovnou súčinu mohutností vstupujúcich relácií.
$R_1 \times R_2 = \{x_1, x_2, ..., x_n, y_1, y_2, ..., y_m | P_1(x_1, x_2, ..., x_n) \wedge P_2(y_1, y_2, ..., y_m)\}$.

Definícia 2.1.7 (Join)**.** *Joinom* štandardne rozumieme prirodzené spojenie. Formálne: $R_1(x) \bowtie R_2(y) = \{\widehat{xy} : P_1(x) \wedge P_2(y)\}$ Joinovať možno relácie podľa jedného alebo viacerých atribútov. Neskôr túto operáciu zovšeobecníme na prípad nulového počtu joinovacích atribútov. Atribút násobnosti riadkov výslednej relácie je súčinom atribútov násobnosti riadkov jednotlivých joinujúcich relácií: $(R_1 \bowtie R_2).m = R_1.m * R_2.m$

Definícia 2.1.8 (Antijoin)**.** Sémantika antijoinu je veľmi podobná sémantike množinového rozdielu relácií. Odčítať však môžeme jedine kompatibilné relácie. Antijoin naproti tomu vyberie z prvej relácie riadky, ktorých žiadny joinovací atribút sa nerovná zodpovedajúcemu joinovaciemu atribútu druhej relácie. Zapísané v relačnom kalkule : $R_1(x) \bar{\bowtie} R_2(y) = \{x : P_1(x) \wedge \neg P_2(y)\}$

Definícia 2.1.9 ($\Theta - join$)**.** Všetky druhy joinov možno chápať ako špeciálny prípad tejto operácie. Formálne : $R_1(x) \bowtie_F R_2(y) = \sigma_F(R_1(x) \times R_2(y))$, kde F je joinovacia podmienka (pri klasickom joine je to test na rovnosť hodnôt atribútov z množiny joinovacích atribútov).

Definícia 2.1.10 (Selektivita joinu)**.** Selektivita je číslo z intervalu $< 0, 1 >$, ktoré vyjadruje schopnosť joinu zmenšiť výslednú reláciu v porovnaní s kartézskym súčinom. Join považujeme za tým selektívnejší, čím je tento pomer menší. Selektivitu joinu konečných relácií vyjadruje podiel $\frac{|R \bowtie S|}{|R \times S|}$. Ak je jedna z relácií nekonečná, kladieme selektivitu $\frac{|R \bowtie S|}{|R|}$, kde R je konečná relácia. V prípade dvoch nekonečných relácií je selektivita ich joinu definitoricky 0. Analogicky možno definíciu rozšíriť aj na $\Theta - join$.

Definícia 2.1.11 (Funkčná závislosť)**.** Nech Ω je množina všetkých atribútov relačnej schémy R. Nech $X, Y \subseteq \Omega$ sú množiny atribútov. Hovoríme, že X určuje Y $(X \rightarrow Y)$, ak pre ľubovoľné 2 riadky r_1, r_2 prípustnej relácie R^* platí : ak sa r_1, r_2 zhodujú vo všetkých atribútoch množiny X, potom sa zhodujú aj vo všetkých atribútoch množiny Y.

Definícia 2.1.12 (Pseudokľúč)**.** Nech R je relačná schéma a Ω jej množina atribútov. Podmnožinu $I \subseteq \Omega$ nazveme *vstupnou množinou* atribútov, ak existuje len konečný počet n-tíc s rovnakými hodnotami atribútov z I. Minimálnu vstupnú množinu atribútov $P \subseteq \Omega$ v množinovom zmysle nazveme pseudokľúč. Pre všetky konečné relácie je pseudokľúč prázdna množina.

Definícia 2.1.13 (Nadkľúč)**.** Ak R je relačná schéma a Ω jej množina atribútov, tak $S \subseteq \Omega$ je nadkľúčom práve vtedy, keď $S \rightarrow \Omega$.

Definícia 2.1.14 (Kľúč)**.** Ak R je relačná schéma a Ω jej množina atribútov, tak $K \subseteq \Omega$ je kľúčom $R \Leftrightarrow K$ je nadkľúčom, ale žiadna jeho vlastná podmnožina nadkľúčom R nie je. Kľúč je teda minimálny nadkľúč v množinovom zmysle.

Agregačné funkcie nie sú štandardnou súčasťou relačnej algebry. Klasická relačná algebra má totiž rovnakú výrazovú silu ako jazyk logiky prvého rádu. To znamená, že v nej nemožno vykonať výpočet pevného bodu (teda napr. vypočítať tranzitívny uzáver grafu). Takisto nie je možné v takomto jazyku vyjadriť spočítavanie objektov, teda napr. porovnávať mohutnosti množín. Pre naše potreby preto rozšírime okruh relačných operátorov o operáciu agregácie.

Definícia 2.1.15 (Agregačná funkcia)**.** Agregačnú funkciu Γ formálne definujeme pomocou relačného kalkulu nasledovne:
$\Gamma_{\substack{x \\ agg(y)}}(R(x,y,z)) = \{xy : (\uplus agg(y), z)P(x,y,z)\}$, kde x je zoskupujúci(groupby) atribút a y je agregovaný atribút.

Definícia 2.1.16. Hovoríme, že agregačná funkcia Γ je *dekomponovateľná*, ak existujú agregačné funkcie $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$ také, že $\Gamma(S_1 \cup S_2)$ sa dá napísať ako aritmetický výraz nad $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$, kde S_1, S_2 sú množiny hodnôt a $\cup$ je operátor zjednotenia. Táto vlastnosť má veľký význam pre rozdeľovanie agregácií.

Platia nasledovné vzťahy:

- $Sum(S_1 \cup S_2) = Sum(Sum(S_1), Sum(S_2))$
- $Min(S_1 \cup S_2) = Min(Min(S_1), Min(S_2))$
- $Count(S_1 \cup S_2) = Sum(Count(S_1), Count(S_2))$
- $Avg(S_1 \cup S_2) = \frac{Sum(Avg(S_1) * Count(S_1), Avg(S_2) * Count(S_2))}{Sum(Count(S_1), Count(S_2))}$

V procese optimalizácie narazíme pri realizácii skorej agregácie na problém duplikátov. Tomu musíme prispôsobiť aj sémantiku modifikovanej algebry.
Citlivosť na duplicitu určuje, či sa odstránením duplikátov na vstupe zmení výsledok agregačnej funkcie. Sum, Count, Avg sú citlivé na duplikáty, t.j. duplicitné riadky musia byť zachované pred ich vyhodnotením. Naopak, Max a Min nie sú citlivé na duplikáty.

Definícia 2.1.17 (Úspora agregácie)**.** Pod úsporou agregácie rozumieme schopnosť operácie zoskupenia zmenšiť mohutnosť výslednej relácie. Vyjadruje ju podiel $\frac{|\Gamma_{agg(y)}^{x}(R)|}{|R|}$, pričom agregácia je tým úspornejšia, čím je tento pomer menší. Táto vlastnosť agregačných funkcií je kľúčová z pohľadu optimalizácie.

2.2 Nekonečné relácie

Nekonečné relácie nemožno v databáze uložiť ako (konečné) tabuľky, treba ich implementovať vo forme virtuálnych relácií pomocou zodpovedajúcich algoritmov.

Matematické relácie je potrebné implementovať ako programy. Takýto program sa navonok správa ako relácia. Za predpokladu rekurzívnosti všetkých relácií v databáze sme ku každej vstupnej množine schopní poskytnúť algoritmus(index), ktorý vráti všetky n-tice s danou hodnotou vstupnej množiny.

Príklad 2.1. : Sčítanie reprezentujeme matematickou reláciou $add(x, y, z) = \{xyz : z = x+y\}$. Vstupnou množinou pre túto reláciu sú všetky dvojprvkové množiny.

Operátory aritmetického porovnania intuitívne implementujeme ako virtuálne relácie $less(a, b), gr(a, b), eq(a, b)$. V relácii $less(a, b)$ pre všetky záznamy platí $x < y$, analogicky $gr(a, b), eq(a, b)$.

Pre eq sú pseudokľúčom množiny $\{a\}$ a $\{b\}$, pre gr a $less$ je to len množina $\{a, b\}$. Uvedené množiny sú pre tieto relácie aj kľúčmi.
Naproti tomu, pre reláciu $Factor(x, y) = \{xy : x \text{ delí } y\}$ je pseudokľúčom $\{y\}$, kľúčom je však len $\{x, y\}$.

Pri optimalizácii používame len matematické relácie s 2 alebo 3 atribútmi. Táto trieda relácií kompletne pokryje naše potreby. Relácie s 2 atribútmi slúžia na porovnávanie atribútov. Na implementáciu ľubovoľného algoritmu postačia 2 atribúty ako vstup a jeden ako výstup 3-atribútovej relácie. Zložitejšie vzťahy modelujeme príslušnými joinami. Nekonečné relácie využívame

pri simulovaní selekcie joinom. Bližšie je tento proces popísaný v nasledujúcej kapitole.

2.3 Zovšeobecnenie operácií

Táto kapitola pojednáva o špecifickom pohľade na operácie selekcie a projekcie. Tieto chápeme ako špeciálne prípady joinu, resp. agregácie. Odlišný pohľad na základné databázové operácie tak umožňuje špecifický prístup pri optimalizácii dotazov s agregačnými funkciami. Pokiaľ sme teda doposiaľ hovorili o **SPJA**(select, project, join, aggregate) dotazoch, odteraz budeme pracovať s tzv. **GPJ**(Generalized projection, Join) dotazmi.

2.3.1 Selekcia

Selekcia je základná databázová operácia pracujúca s jednotlivými riadkami. Slúži na vybratie riadkov do výslednej relácie na základe určitého kritéria. Základom reprezentácie selekcie joinom je nahradenie selekčnej podmienky zodpovedajúcou reláciou, ktorá prakticky realizuje vzťah medzi atribútmi. Táto relácia potom pri vhodnom joinovaní so vstupnou reláciou vyberie požadované riadky. Selekčná podmienka je vyjadrená formulou podľa definície v 2. kapitole. Okrem uvedených konštruktov musíme však brať do úvahy aj aritmetické operácie, nakoľko tie sa tiež vyskytujú vo formulách.

Pri tvorbe formúl vystačíme s :

- relačnými operátormi $<$, $>$, $=$
- logickými operátormi $\neg$, $\wedge$, $\vee$
- matematickými operáciami $+$, -, *, / ...

Kedže platia nasledovné vzťahy :

- $A \leq B \equiv (A < B) \vee (A = B)$
- $A \geq B \equiv (A > B) \vee (A = B)$
- $A \neq B \equiv \neg(A = B)$

a pomocou $\neg$, $\wedge$, $\vee$ možno vyjadriť každú binárnu logickú spojku, tieto operácie postačujú na tvorbu všetkých selekčných formúl.
Formuly selekčných podmienok na jazyku relácií definujeme induktívne :

- *Relačné operátory*

  $=$

  Test na rovnosť s konštantou reprezentuje jednoriadková relácia *E(vstupný atribút,konštanta)*.
  Test na rovnosť atribútov dvoch relácií je priamou aplikáciou klasického joinu.

  $<, >$

  Porovnanie s konštantou je realizované joinom s nekonečnou reláciou *L(vstupný atribút, konštanta)* pre operátor $<$, resp.
  G(vstupný atribút, konštanta) pre operátor $>$.
  V oboch prípadoch je relácia naplnená riadkami, ktoré majú prvý atribút menší (resp. väčší) ako hodnota konštantného druhého atribútu.

  Porovnanie atribútov relácie pre prípad aritmetického operátora $<$ uskutočníme joinom relácie $less(a, b)$ so vstupnou reláciou $R(a, b, x_1, ..., x_n)$ podľa atribútov a, b. Formálne:

  $$\sigma_{a<b}R(a, b, x_1, ..., x_n) = R(a, b, x_1, ..., x_n) \bowtie less(a, b)$$

  Operátor $>$ realizujeme analogicky.

- *Logické operátory*

  Prvým krokom prepisu selekčných podmienok je ich transformácia na konjunktívnu normálnu formu. Ďalej využitím DeMorganových pravidiel prepíšeme výrazy tvaru $\neg(F_1 \wedge F_2)$ na $(\neg F_1 \vee \neg F_2)$ a $\neg(F_1 \vee F_2)$ na $(\neg F_1 \wedge \neg F_2)$. V ďaľšom označme R reláciu zúčastnenú na selekcii.

  Nech F_1, F_2 sú selekčné formuly a P_1, P_2 sú im zodpovedajúce relácie. Potom $F_1 \wedge F_2$ reprezentujeme ako $(R \bowtie P_1) \cap (R \bowtie P_2)$.

  Nech F_1, F_2 sú selekčné formuly a P_1, P_2 sú im zodpovedajúce relácie. Potom $F_1 \vee F_2$ reprezentujeme ako $(R \bowtie P_1) \cup (R \bowtie P_2)$.

  Nech F je selekčná formula a nech P je relácia, ktorá jej zodpovedá. Potom $\neg F$ reprezentujeme ako $R \bar{\bowtie} P$.

- *Aritmetické operácie* možno za predpokladu znalosti vstupnej množiny realizovať joinom s vhodnou matematickou reláciou.

2.3.2 Projekcia

Projekcia je operácia nad celou reláciou manipulujúca s atribútmi. Táto operácia sa správa rovnako ako agregácia bez výpočtu agregátov. Konkrétne možno napísať:

SELECT DISTINCT D FROM R ≡ SELECT D FROM R GROUP BY D

V relačnej algebre :

$$\Pi_D R = \Gamma_{\substack{D \\ D}}(R)$$

Toto je myšlienka zovšeobecnenia operácie agregácie aj na projekciu [02]. Zavedieme teda novú operáciu, ktorú nazveme *Zovšeobecnená projekcia(Generalized projection - GP)*. Podobne ako agregácia, aj GP obsahuje zoskupujúce komponenty a agregované komponenty. Komponentami sú atribúty a funkcie, ktorých argumentami sú tieto atribúty. *Vyžadovanými atribútmi* GP nazveme zoskupujúce atribúty spolu s agregovanými atribútmi. V prípade, že GP nahrádza projekciu, neobsahuje žiadne agregované komponenty. Zoskupujúce komponenty sú kľúčom výslednej relácie, čo má dva dôležité dôsledky:

1. Ak vyžadované atribúty funkčne určujú zvyšné atribúty vstupnej relácie(napr. keď je súčasťou zoskupujúcich komponentov GP kľúč vstupnej relácie), agregáciu môžeme vynechať a GP bude jednoduchou projekciou na tieto atribúty.

2. GP eliminuje duplikáty vo výsledku.

   Projekcia zachovávajúca duplikáty je potom reprezentovaná agregačnou funkciou *count*. Táto spočíta riadky, ktoré sa zhodujú v zoskupujúcich atribútoch:

   SELECT D FROM R ≡ SELECT D, count(*) AS m FROM R GROUP BY D

   V relačnej algebre :

   $$\Pi_d R = \Gamma_{\substack{m=count(*) \\ d}}(R_m)$$

   kde R_m je pôvodná relácia R s pridaným atribútom násobnosti m. Kvôli dekomponovateľnosti je potrebné, aby m atribút modifikovala aj GP *avg*. Ostatné GP tento atribút využívajú len pri dodatočnej agregácii.

Technická poznámka : Operácia GP môže slúžiť aj na *premenovanie atribútov*. Je to len technická operácia bez reálneho dopadu na výsleok. V relačnom kalkule používame vo výrazoch premenné, naproti tomu operácie relačnej algebry manipulujú nad natívnymi atribútmi, ktoré treba pred aplikáciou joinu kvôli kompatibilite premenovať.

Príklad 2.2. : Výraz $P(x, y) \wedge P(u, v)$ preložíme do algebry ako $R(x, y) \bowtie \Gamma_{\substack{u \leftarrow x \\ v \leftarrow y}}(R(x, y))$.

Pri konštrukcii algoritmu optimalizácie nebudeme tejto operácii venovať špeciálnu pozornosť.

2.3.3 Algebraický stroj

Dotaz na databázu v relačnej algebre je spracovávaný algebraickým strojom. Uvedieme potrebné modifikácie klasického algebraického stroja na spracovanie dotazov modifikovanej algebry.

- Kartézsky súčin chápeme ako špeciálny prípad joinu s prázdnou množinou joinovacích atribútov. Priamej realizácii tejto operácie sa pri konštrukcii optimalizačných pravidiel snažíme vyhnúť, nakoľko selektivita zodpovedajúceho joinu je 1. Join reprezentujúci kartézsky súčin preto počítame pomocou techniky nested-loops.

- Join dvoch relácií, z ktorých aspoň jedna je nekonečná uskutočníme metódou nested-loops. $\Theta - join$ potom realizujeme nasledovne:
  Postupne generujeme výslednú reláciu ako pri kartézskom súčine. Každú novovytvorenú n-ticu pred jej zaradením do výsledku joinujeme s matematickou reláciou reprezentujúcou selekčnú podmienku $\Theta - joinu$. Pomocou tohto prístupu sa vyhneme konštrukcii veľkej dočasnej relácie. V modifikovanej relačnej algebre vyzerá realizácia $\Theta - joinu$ nasledovne:

  $$R_1 \bowtie_F R_2 = (R_1 \bowtie R_2) \bowtie R_F$$

  kde R_F je relácia reprezentujúca selekčnú podmienku vyjadrenú formulou F. Pseudokľúč pre R_F je súčasťou testovanej n-tice. Časová zložitosť takejto operácie je $O(|R_1| * |R_2|)$.

- Logické operátory sú v modifikovanej algebre definované pomocou množinových operácií. Takáto priamočiara realizácia by však nebola optimálna vzhľadom na spoločné podvýrazy vo formuliach. Vezmime prípad atomických formúl A_1, A_2 reprezentovaných reláciami R_{A_1} a R_{A_2}. Podobne ako pri výpočte $\Theta - joinu$ prechádzame riadky vstupnej konečnej relácie R. V prípade formuly $A_1 \wedge A_2$ joinujeme R najprv s R_{A_1} a vzniknutú reláciu s R_{A_2}. Formulu $A_1 \vee A_2$ realizujeme dvomi joinami: $R_1 = R \bowtie R_{A_1}$ a $R_2 = R \bowtie R_{A_2}$, pričom R_2 treba počítať len ak $R_1 = \emptyset$. Do výslednej relácie zaradíme ľubovoľnú neprázdnu z týchto dvoch n-tíc.

- Pri klasickom Joine dvoch konečných relácií preferujeme metódu *sort&merge*. Dôvodom je skutočnosť, že výpočet agregácie zahŕňa triedenie. Vezmime prípad výpočtu GP a následného joinu. Pokiaľ sú joinovacie atribúty súčasťou zoskupujúcich komponentov GP, ušetríme triedenie

pri sort&merge joine. Podobná situácia nastane pri opačnom poradí vykonania joinu a GP.

- Množinové operácie zjednotenie, prienik a rozdiel realizujú SQL príkazy UNION, INTERSECT a EXCEPT. Keďže každá z týchto operácií pracuje len s kompatibilnými množinami, vykonávame ich až po vyhodnotení jednotlivých zložiek. Operácie, ktoré relácie zmenšujú(prienik, rozdiel) vykonávame čo najskôr. Prienik sa dá reprezentovať joinom relácií podľa všetkých ich atribútov a rozdiel možno nahradiť antijoinom podľa všetkých atribútov. Tieto operácie však ponecháme v nezmenenej podobe, nakoľko v stratégii optimalizácie majú osobitné miesto.

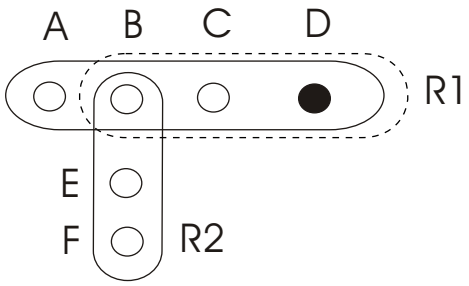
Definovali sme transformácie a rozšírenie klasickej relačnej algebry, ktorú využijeme v ďaľšom skúmaní. Naša revidovaná relačná algebra má teda rovnakú výrazovú silu ako SQL a pozostáva z nasledovných operácií:

- Join pokrývajúci kartézsky súčin, join a v kombinácii s matematickými reláciami aj selekciu a $\Theta - join$
- Generalizovaná projekcia $\Gamma_{\substack{x \\ agg(y)}}(R)$ zahŕňajúca projekciu a agregáciu
- Množinové operácie $\cup, \cap, -$

3 Techniky optimalizácie

V tejto kapitole preskúmame rôzne transformácie plánu vyhodnotenia z pohľadu korektnosti ich použitia. Naša modifikovaná algebra obsahuje okrem množinových operácií len GP a joiny, transformácie sa budú preto týkať najmä vzťahov týchto dvoch operácií. Pojednaniu o vhodnosti použitia týchto úprav je venovaná samostatná kapitola.

Na vizualizáciu dotazu využije hypergraf. Uzlami grafu sú atribúty jednotlivých relácií. Uzly zodpovedajúce joinovacím atribútom stotožníme. Tieto budeme v ďaľšom označovať termínom **Join Point(JP)**. V prípade kartézskeho súčinu pridáme do grafu fiktívne JP. Agregované atribúty označíme plným a ostatné atribúty prázdnym krúžkom. Ak je nejaký atribút agregovaným atribútom jednej GP a súčasne zoskupujúcim atribútom inej GP, označíme ho plným krúžkom. Plnými hyperhranami spojíme atribúty patriace jednej relácii. Do hypergrafu zahrnieme aj pomocné nekonečné relácie a im zodpovedajúce hrany. Čiarkovanými hranami spojíme atribúty zúčastňujúce sa na GP (agregované aj zoskupujúce). Ku grafu podľa potreby pridáme názvy relácií a ich atribútov.



Obrázok 1: Reprezentácia dotazu hypergrafom

Príklad 3.1. Daná je relačná schéma $R_1(A, B, C, D)$, $R_2(E, F, G)$. Na obrázku 1 je znázornený SQL dotaz

SELECT $B, C, agg(D)$
FROM R_1, R_2
WHERE $R_1.B = R_2.G$
GROUP BY B, C

v modifikovanej algebre zapísaný ako $\Gamma_{\substack{B,C \\ agg(D)}}(R_1 \bowtie R_2)$.

3.1 Spájanie/rozdeľovanie agregácií

Nasledovné techniky sú súčasťou Eager/Lazy transformácii a využívajú dekomponovateľnosť GP.

V procese optimalizácie môžeme naraziť na prípad dvoch alebo viacerých GP idúcich za sebou bez joinu medzi nimi. V takomto prípade je výhodné *spojiť* tieto výpočty do jedného, ak je to možné. V ďaľšom označme prvú agregáciu v poradí vykonania GP_1 a druhú GP_2.

Transformácia 3.1.1. Ak GP_2 nie je citlivá na duplicitu (min, max, projekcia) a všetky jej vyžadované atribúty sú zoskupujúcimi atribútmi GP_1, výpočet GP_1 môžeme vynechať.

Transformácia 3.1.2. Ak agregovaný atribút GP_2 je agregovaným atribútom GP_1 a súčasne GP_1 aj GP_2 sú rovnakého typu, výpočet agregácie vykonáme vrámci GP_2. V prípade, že nejaký zoskupujúci atribút GP_2 je agregovaným atribútom GP_1, nemôžeme GP_1 a GP_2 spojiť.

Transformácia 3.1.3. Kaskáda GP. Ak prvá GP v kaskáde neobsahuje žiadne agregované atribúty (simulácia projekcie), potom platí :

$$\Gamma_{\substack{A\\A}}(\Gamma_{\substack{B\\B}}(R)) = \Gamma_{\substack{A\\A}}(R)$$

$$\Gamma_{\substack{A\\A}}(\Gamma_{\substack{C\\agg(D)}}(R)) = \Gamma_{\substack{A\\A}}(R)$$

pričom C je ľubovoľný atribút (vrátane A).

Rozdelenie výpočtu GP do kaskády viacerých GP idúcich za sebou je opačným postupom k spájaniu. Po rozdelení GP musí platiť, že vzniknuté zložky spojením vytvoria pôvodnú GP. Pri optimalizácii využívame najmä čiastočné grupovanie podľa jednotlivých atribútov z(ale aj mimo) množiny zoskupujúcich atribútov. Kvôli korektnosti musia všetky zoskupujúce atribúty GP_2 figurovať medzi zoskupujúcimi atribútmi všetkých čiastočných agregácií.

Transformácia 3.1.4. Daná je relácia $R(A, B, C, D)$. Potom

$$\Gamma_{\substack{A\\agg(D)}}(R) = \Gamma_{\substack{A\\f(Y)}}(\Gamma_{\substack{A,B\\Y=agg(D)}}(R)) = \Gamma_{\substack{A\\f(Y)}}(\Gamma_{\substack{A,C\\Y=agg(D)}}(R)) = \Gamma_{\substack{A\\f(Y)}}(\Gamma_{\substack{A,B,C\\Y=agg(D)}}(R))$$

kde agg je dekomponovateľná agregačná funkcia a f je dodatočná agregácia z definície dekomponovateľnosti(napr. pre $agg \equiv count$ je $f \equiv sum$). Uvedená transformácia je priamou súčasťou skorej agregácie.

3.2 Eager transformácie

Trieda transformácií využívajúca schopnosť agregácie významne zredukovať počet riadkov vo výslednej relácii. Samotná agregácia je vykonaná skôr ako pri priamočiarom prepise dotazu do algebry, preto ju zvykneme označovať **skorá** agregácia(Early, alebo tiež Eager aggregation). Technika využíva výpočet čiastočných grúp pred joinom. Takto sa zredukuje počet riadkov vstupujúcich do joinu, na ktoré potom ale treba aplikovať dodatočný výpočet agregácie. Tá je odvodená z vlastnosti dekomponovateľnosti pôvodnej agregácie. Za určitých podmienok možno túto agregáciu vynechať. Táto technika sa dá s úspechom použiť na podporu pre tvorbu materializovaných view obsahujúcich čiastočné agregáty. V praxi sa táto metóda využíva hlavne pri distribuovaných systémoch. Problematikou view sa však nebudeme bližšie zaoberať.

Eager transformácie nemenia sémantiku dotazu, len spôsob jeho vyhodnotenia. Musia preto spĺňať určité podmienky:

1. Keďže GP projektuje vstupnú reláciu na svoje vyžadované atribúty, sú medzi nimi joinovacie atribúty všetkých joinov nasledujúcich po tejto GP.

2. Zjednotenie množín zoskupujúcich atribútov čiastočných(skorých) GP je nadmnožinou množiny zoskupujúcich atribútov pôvodnej GP. Táto podmienka pochádza z pravidiel o rozdeľovaní agregácií.

3. V prípade GP citlivých na duplikáty je po aplikovaní čiastočných GP zachovaná mohutnosť relácií bez agregovaných atribútov vstupujúcich do joinu, ktorý predchádza pôvodnej GP. Túto podmienku zabezpečíme použitím m atribútu relácií.

Na základe týchto sémantických obmedzení konštruujeme jednotlivé transformačné pravidlá. Ukážkové prípady sú kvôli prehľadnosti zjednodušené, priamočiaro ich však možno zovšeobecniť na zložitejšie varianty. Viac príkladov sa nachádza v poslednej kapitole. Na popis jednotlivých transformácií využijeme relačnú schému z úvodu kapitoly. Podobne ako pri rozdeľovaní-spájaní agregácií označíme pôvodnú agregačnú funkciu agg a dodatočnú agregáciu f.

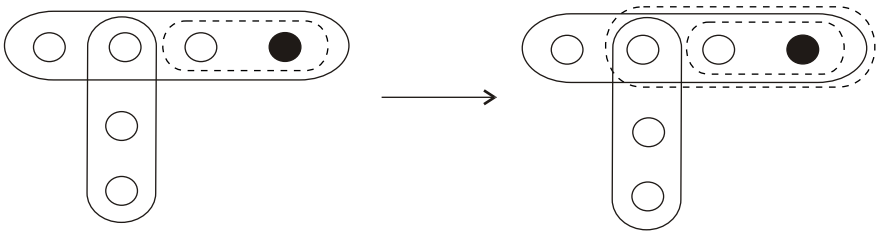
Transformácia 3.2.1. $\Gamma_{\substack{B,C \\ agg(D)}}(R_1 \bowtie R_2) \equiv \Gamma_{\substack{B,C \\ agg(D)}}(R_1) \bowtie R_2$

Situácia pre tento prípad je znázornená na obrázku 1. Namiesto aplikovania agregácie na join relácií R_1, R_2 vykonáme agregáciu na R_1 a výsledok následne joinujeme s R_2. Takáto transformácia je najjednoduchšia(vymení sa

len poradie joinu a GP) a v praxi najefektívnejšia. Na jej korektné uskutočnenie je však nutné splniť pomerne silný predpoklad: Všetky joinovacie atribúty relácie R_1 sú zoskupujúce atribútmi GP(podmienka 1) a zároveň $R_1 \bowtie R_2$ je joinom na cudzí kľúč. V našom konkrétnom prípade teda $R_2.G$ je cudzím kľúčom v R_1. Dôvod tohto obmedzenia je nasledovný[03]: Musí platiť, že riadky prislúchajúce rôznym grupám po aplikovaní skorej GP budú prislúchať rôznym grupám aj vo výsledku. Cudzí kľúč sa joinuje maximálne s jedným riadkom danej relácie, pričom klasický join by mohol eventuelne duplikovať niektoré hodnoty zoskupujúcich atribútov. To by ale nebolo sémanticky korektné, nakoľko agregačná funkcia počíta agregáty práve na základe týchto hodnôt. V prípade, že uvedená podmienka neplatí, je potrebné vykonať dodatočnú agregáciu: $\Gamma_{\substack{B,C \\ f(D)}}(\Gamma_{\substack{B,C \\ agg(D)}}(R_1) \bowtie R_2)$

Transformácia 3.2.2. $\Gamma_{\substack{C \\ agg(D)}}(R_1 \bowtie R_2) \equiv \Gamma_{\substack{C \\ f(D)}}(\Gamma_{\substack{B,C \\ agg(D)}}(R_1) \bowtie R_2)$

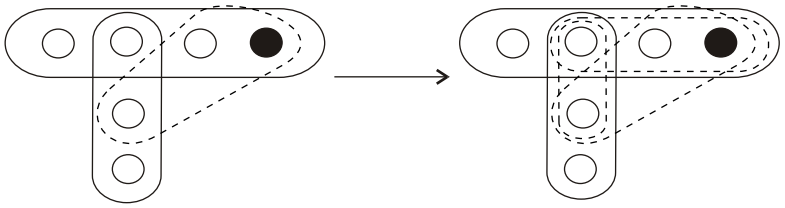
V tomto prípade nie je splnená podmienka 1, výpočet agregácie treba teda rozdeliť na 2 fázy : agg a f. Pre $\Gamma_{\substack{B,C \\ agg(D)}}(R_1)$ sú však už splnené všetky podmienky, možno ju teda realizovať pred joinom s R_2. Po vykonaní tohto joinu sú splnené podmienky aj pre dodatočnú agregáciu pomocou ktorej z čiastočných grúp vyrobíme konečný výsledok.



Obrázok 2: Eager transformácia č. 2

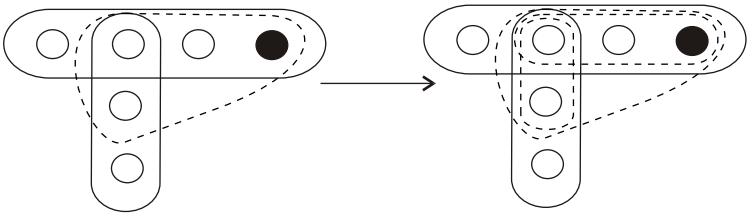
Transformácia 3.2.3. $\Gamma_{\substack{C,E \\ agg(D)}}(R_1 \bowtie R_2) \equiv \Gamma_{\substack{C,E \\ f(D)}}(\Gamma_{\substack{B,C \\ agg(D)}}(R_1) \bowtie \Gamma_{\substack{E,G \\ E,G}}(R_2))$

Uvedený dotaz porušuje podmienku 1, nakoľko joinovací atribút nie je súčasťou zoskupujúcich atribútov ani pôvodnej GP. Tú preto rozdelíme na 2 zložky, ktoré operujú nad jednotlivými joinujúcimi reláciami R_1, R_2. Pre ne už platia podmienky vykonania skorej agregácie.



Obrázok 3: Eager transformácia č. 3

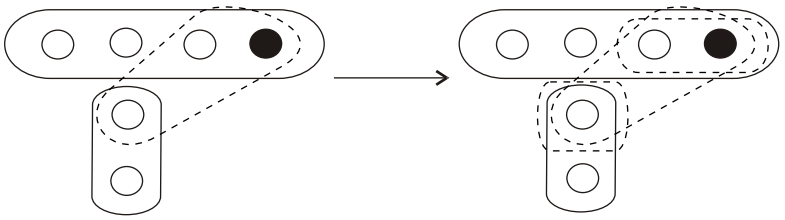
Transformácia 3.2.4. $\Gamma_{\substack{E,B,C \\ agg(D)}}(R_1 \bowtie R_2) \equiv \Gamma_{\substack{E,B,C \\ f(D)}}(\Gamma_{\substack{B,C \\ agg(D)}}(R_1) \bowtie \Gamma_{\substack{E,G \\ E,G}}(R_2))$
Situácia podobná predošlej, až na to, že pôvodná GP zahŕňa vrámci svojich zoskupujúcich atribútov všetky joinovacie atribúty. Priamemu vykonaniu skorej agregácie bráni však neprítomnosť všetkých vyžadovaných atribútov v čase jej aplikácie. Zoskupujúci atribút E je totiž súčasťou relácie R_2 a k dispozícii je až po vykonaní joinu R_1 s R_2. Riešením je rovnaký postup ako v predchádzajúcom prípade.



Obrázok 4: Eager transformácia č. 4

Transformácia 3.2.5. $\Gamma_{\substack{C,E \\ agg(D)}}(R_1 \bowtie R_2) \equiv \Gamma_{\substack{C,E \\ f(D)}}(\Gamma_{\substack{B,C \\ agg(D)}}(R_1) \bowtie \Gamma_{\substack{E \\ E}}(R_2))$
Vezmime 3. transformáciu a aplikujme ju na prípad joinu s prázdnou množinou joinovacích atribútov. Pri realizácii skorej agregácie v tomto prípade teda stačí overovať prirodzenú požiadavku na prítomnosť všetkých vyžadovaných atribútov danej GP. Zoskupujúcimi atribútmi parciálnych GP budú zoskupujúce atribúty pôvodnej GP vrámci jednotlivých joinujúcich relácií. Z

dôvodu uvedeného v 1. pravidle treba nakoniec aplikovať dodatočnú GP.



Obrázok 5: Eager transformácia č. 5

Na relačnej schéme z Príkladu 3.1 popíšeme rolu m atribútu pri jednotlivých transformáciách na konkrétnych GP. Po joine s R_2 má záznam vo výslednej relácii vždy $R_2.m$-krát väčšiu mohutnosť ako mal v R_1. Tomu treba náležite prispôsobiť výpočty agregátov v dodatočných GP citlivých na duplicitu.

- $\Gamma\limits_{\substack{B,C,E\\ min(D)}}(R_1 \bowtie R_2) \equiv \Gamma\limits_{\substack{B,C,E\\ min(D)}}(\Gamma\limits_{\substack{B,C\\ min(D)}}(R_1) \bowtie \Gamma\limits_{\substack{G,E\\ G,E}}(R_2))$
- $\Gamma\limits_{\substack{B,C,E\\ max(D)}}(R_1 \bowtie R_2) \equiv \Gamma\limits_{\substack{B,C,E\\ max(D)}}(\Gamma\limits_{\substack{B,C\\ max(D)}}(R_1) \bowtie \Gamma\limits_{\substack{G,E\\ G,E}}(R_2))$
- $\Gamma\limits_{\substack{B,C,E\\ sum(D)}}(R_1 \bowtie R_2) \equiv \Gamma\limits_{\substack{B,C,E\\ sum(D*R_2.m)}}(\Gamma\limits_{\substack{B,C\\ sum(D)}}(R_1) \bowtie \Gamma\limits_{\substack{G,E\\ m=count(*)}}(R_2))$
- $\Gamma\limits_{\substack{B,C,E\\ count(*)}}(R_1 \bowtie R_2) \equiv \Gamma\limits_{\substack{B,C,E\\ sum(R_1.m*R_2.m)}}(\Gamma\limits_{\substack{B,C\\ m=count(*)}}(R_1) \bowtie \Gamma\limits_{\substack{G,E\\ m=count(*)}}(R_2))$
- $\Gamma\limits_{\substack{B,C,E\\ avg(D)}}(R_1 \bowtie R_2) \equiv \Gamma\limits_{\substack{B,C,E\\ \frac{sum(avg(D)*R_1.m*R_2.m)}{sum(R_1.m*R_2.m)}}}(\Gamma\limits_{\substack{B,C\\ avg(D),m=count(*)}}(R_1) \bowtie \Gamma\limits_{\substack{G,E\\ m=count(*)}}(R_2))$

Vykonanie GP vo vetve R_2 teda so sebou prináša dodatočnú réžiu v podobe aktualizácie m atribútu. Pri konštrukcii optimalizačných pravidiel musíme preto tento fakt zohľadniť.

V doterajších úvahách sme predpokladali, že joiny sa týkajú iba zoskupujúcich atribútov GP. Analyzujme situáciu, keď joinovací atribút patrí medzi agregované atribúty GP:

- GP je súčasťou výrazu za SQL operátorom HAVING. Poradie vyhodnotenia naznačuje aj naivný prepis do relačnej algebry - join nasleduje až **po** vyhodnotení GP.

- V opačnom prípade je nutné vykonať join **pred** GP, nakoľko dotyčný agregovaný atribút vystupuje v SQL dotaze vo WHERE podmienke.

V oboch prípadoch je teda vzájomné poradie joinu a GP určené jednoznačne sémantickými obmedzeniami.

3.3 Lazy transformácie

Pozdržať vyhodnotenie agregácie až po vyhodnotení joinu je vhodné vtedy, ak je join dostatočne selektívny. Táto operácia je určitým spôsobom reverzná voči eager transformácii a označuje sa **neskorá** agregácia(Lazy aggregation). V tomto prípade joinom zredukovaný počet riadkov urýchli výpočet agregácie. Využitie nájde táto metóda hlavne pri joinoch, ktoré simulujú selekcie so silnými selekčnými podmienkami. Naivný prepis relačného dotazu do relačnej algebry obsahuje výpočet agregácie až po všetkých joinoch. Pre použitie tejto techniky teda neexistujú špeciálne obmedzenia, keďže je príklonom ku klasickému výpočtu. V našej stratégii optimalizácie ju chápeme ako neaplikovanie skorej agregácie.

3.4 Ostatné transformácie

Algebraické pravidlá pre vyhodnocovanie dotazov známe z klasickej relačnej algebry[01] teraz popíšeme na jazyku modifikovanej algebry.

Algebraické pravidlá modifikovanej algebry:

Transformácia 3.4.1. Komutativita joinu

$$R_1 \bowtie R_2 = R_2 \bowtie R_1$$

Vďaka zovšeobecneniu operácií táto transformácia pokrýva komutativitu joinu a kartézskeho súčinu so selekciou a medzi sebou.

Transformácia 3.4.2. Asociativita joinu

$$R_1 \bowtie (R_2 \bowtie R_3) = (R_1 \bowtie R_2) \bowtie R_3$$

pokrýva zároveň asociativitu kartézskeho súčinu.

Transformácia 3.4.3. Distributívnosť joinu vzhľadom na join
V prípade, že joinovacie atribúty z R_3 sú obsiahnuté v R_1 aj R_2, platí:

$$(R_1 \bowtie R_2) \bowtie R_3 = (R_1 \bowtie R_2) \bowtie (R_1 \bowtie R_3)$$

Uvedené pravidlo je analógiou komutativity selekcie a joinu z klasickej algebry. Selekčnú podmienku vyjadruje relácia R_3.

Transformácia 3.4.4. Distributívnosť joinu vzhľadom na množinové operácie $\cup$ a -

$$R_1 \bowtie (R_2 \cup R_3) = (R_1 \bowtie R_2) \cup (R_1 \bowtie R_3)$$
$$R_1 \bowtie (R_2 - R_3) = (R_1 \bowtie R_2) - (R_1 \bowtie R_3)$$

Transformácia 3.4.5. Kaskáda joinov

$$(R_1 \cap R_2) \bowtie R_3 = R_2 \bowtie (R_1 \bowtie R_3)$$

Toto pravidlo sa používa hlavne v situáciách, keď simulujeme selekciu joinom. R_1, R_2 v tomto prípade reprezentujú dve časti konjunkcie selekčnej podmienky.

Uvedené transformácie nám umožnia modifikovať poradie vyhodnotenia joinov pri konštrukcii optimálneho výrazu vyhodnotenia.

4 Výsledky

V záverečnej kapitole uvedieme niekoľko príkladov vhodného použitia optimalizačných techník a poukážeme aj na riziká spojené s ich neuváženou aplikáciou. Opíšeme zároveň integráciu pravidiel odvodených z týchto techník do existujúcich optimizérov. Menší počet operácií modifikovanej relačnej algebry má za následok aj menší počet potrebných pravidiel, čo výrazne zjednoduší návrh optimalizačného algoritmu.

4.1 Optimalizačná stratégia

V kapitole o technikách optimalizácie sme uviedli niekoľko platných transformácií plánu vyhodnotenia. Na tomto mieste preskúmame možnosti použitia týchto techník z hľadiska zlepšenia časovej a priestorovej realizácie plánu. Keďže špecifiká praktických databázových systémov nie sú predmetom nášho skúmania, nezaoberáme sa konkrétnou optimálnou realizáciou operácií joinu resp. agregácie. Pri konštrukcii pravidiel optimalizácie si vystačíme so všeobecnými informáciami o reláciách v databáze, ktoré sú k dispozícii každému systému. Konkrétne sa jedná o:

- Prítomnosť primárneho/cudzieho kľúča
- Fyzická velkosť relácie
- Funkčné závislosti

Hlavným cieľom optimalizácie je čo najviac zmenšiť veľkosť dočasných relácií, ktoré používame na vyhodnotenie dotazu. Táto stratégia odráža všeobecný princíp optimalizačných techník - vykonávať najskôr najúspornejšie operácie.

Zo základných vlastností relácií vyplýva všeobecná selektivita joinov a cena agregácií zúčastnených na dotaze. Hlavným princípom našej stratégie je práve porovnanie týchto dvoch veličín vrámci nasledujúcich úvah:

- Joiny dvoch relácií, z ktorých aspoň jedna je veľmi malá(toto číslo nie je presne ohraničené, vo všeobecnosti ide o počty nepresahujúce jednu desiatku) sa dajú väčšinou realizovať veľmi efektívne. Pre potreby optimalizačného algoritmu im prisúdime odhadovanú selektivitu 0.1.
- Joiny konečných relácií s matematickými reláciami sú zvyčajne vysoko selektívne. Ich selektivitu aproximujeme nulou.

- Možno očakávať, že selektivita joinov na cudzí kľúč je lepšia ako bez prítomnosti cudzieho kľúča. Keďže riadok z relácie obsahujúcej cudzí kľúč sa joinuje s maximálne jedným riadkom druhej relácie, selektivita daného joinu je určite menšia ako 1. Algoritmus predpokladá selektivitu numericky 0.5.

- Ak existuje funkčná závislosť medzi atribútmi joinujúcich relácií, potom tieto relácie pravdepodobne vznikli dekompozíciou inej relácie pomocou Join projecting mapping [01]. Pri takýchto joinoch je selektivita lepšia než pri klasických joinoch a možno ju odhadnúť podobne ako v predchádzajúcom prípade(bližšie v [06]).

- Klasické joiny bez dodatočných vlastností majú veľmi zlú očakávanú selektivitu. Táto selektivita bude navyše tým horšia, čím menej joinovacích atribútov obsahujú. Algoritmus optimalizácie ju bude považovať za 1.

- Úspora GP je zmenšená o cenu výpočtu agregátov. Táto úspora bude teda s rastúcim počtom agregovaných atribútov klesať.

- Možno predpokladať, že čím viac zoskupujúcich atribútov GP obsahuje, tým je jej úspora menšia. Extrémnym prípadom je potom GP s vyžadovanými atribútmi zahŕňajúcimi všetky atribúty relácie. Tento prípad je spomenutý aj v kapitole o zovšeobecnení operácií.

Uvedené skutočnosti transformujeme na heuristiky, ktoré využijeme pri konštrukcii algoritmu optimalizácie:

Pravidlo 4.1.1. Joiny veľmi malých relácií vykonávať pred GP s agregovanými atribútmi.

Pravidlo 4.1.2. Joiny relácií, z ktorých aspoň jedna je matematická vykonávať čo najskôr(pred GP). Toto pravidlo je zhodné s pravidlom o včasnej realizácii selekcie z klasickej algebry.

Pravidlo 4.1.3. GP bez agregovaných atribútov vykonávať hneď ako je to možné, ale neaplikovať na ňu skorú agregáciu. Pravidlo odráža heuristiku klasickej algebry o skorej aplikácii projekcie.

Pravidlo 4.1.4. Joiny na cudzí kľúč a joiny relácií vzniknutých dekompozíciou vykonávať pred klasickými joinami. Pokiaľ je na výsledok joinu aplikovaná nejaká GP, uskutočníme zjednodušenú skorú agregáciu. Konkrétne vezmime situáciu pre Eager transformácie 3 a 4. Ak má daný join uvedené vlastnosti, GP dekomponujeme len na vetvu vrámci relácie s agregovaným

atribútom a dodatočnú GP. Zjednodušené verzie transformácií 3 a 4 potom vyzerajú nasledovne:

$$\Gamma_{\substack{C,E \\ agg(D)}}(R_1 \bowtie R_2) \equiv \Gamma_{\substack{C,E \\ f(D)}}(\Gamma_{\substack{B,C \\ agg(D)}}(R_1) \bowtie R_2)$$

resp.

$$\Gamma_{\substack{E,B,C \\ agg(D)}}(R_1 \bowtie R_2) \equiv \Gamma_{\substack{E,B,C \\ f(D)}}(\Gamma_{\substack{B,C \\ agg(D)}}(R_1) \bowtie R_2)$$

Pomocou tohto prístupu sa v prípade GP citlivých na duplicitu vyhneme aktualizácii m atribútu v relácii R_2, ktorá vzhľadom na dobrú očakávanú selektivitu joinu výpočet len spomalí.

Pravidlo 4.1.5. Klasické joiny dostatočne veľkých relácií vykonávať čo najneskôr, aplikovať na zúčastnené relácie skorú agregáciu podľa Transformácií 3.2.1 - 3.2.5. vždy, keď je to možné.

4.2 Algoritmus optimalizácie

Predstavíme realizáciu optimalizačných pravidiel získaných pomocou teoretických úvah v podobe rule-based optimalizátora. V súčasnej praxi je použitie tohto typu optimalizátorov obmedzené, nakoľko pre výrobcu databázy je výhodnejšie hľadať optimálne vykonanie operácií na základe špecifických, platformovo závislých informácií, ako sú napr. rôzne štatistiky, ktoré si o reláciách databázový systém zaznamenáva. Tento prístup má však jedno negatívum: pri hľadaní optimálneho plánu vyhodnotenia treba navzájom porovnávať mnoho potenciálnych plánov, ktorých počet je vzhľadom na počet relácií vstupujúcich do dotazu vo všeobecnosti exponenciálny.

Na odstránenie tohto nežiadúceho faktu slúži práve navrhovaný optimalizátor. Ten zmenší počet potenciálnych plánov vstupujúcich do cost-based optimalizátora a značným spôsobom tak urýchli celkový proces. Optimalizátor pritom berie do úvahy agregáciu a joiny, teda operácie, ktoré sa na celkovom čase vyhodnotenia podieľajú najväčšou časťou. Výstup rule-based optimalizátora bude okrem dodatočných informácií o prostredí databázy jediným vstupom pre cost-based fázu. Z tejto množiny je potom cost-based procesom s ohľadom na špecifické prostredie databázy vybratý najoptimálnejší plán, ktorý bude realizovaný.

Pracovný graf dotazu - vstup algoritmu skonštruujeme spôsobom uvedeným v predošlej kapitole. Na výstupe je množina výrazov v modifikovanej relačnej algebre s horeuvedenými vlastnosťami.

Algoritmus(Rule-based optimalizácia)

```
INPUT  : Naivný prepis SQL dotazu do relačnej algebry.
OUTPUT : Množina výrazov v modifikovanej algebre.

(1) Transformuj zadaný výraz do modifikovanej algebry
(2) Zostroj hypergraf tohto výrazu
(3) Kým sa v pracovnom grafe niečo mení, opakuj pre všetky grafy:
(4)   Hľadaj výskyty GP a aplikuj na ne transformačné pravidlá
      4.1.1 - 4.1.5
(5) Tranformáciou hypergrafov vyrob množinu výrazov v algebre.
(6) Na výrazy aplikuj transformácie 3.1.1 - 3.1.3
(7) Vráť množinu odvodených výrazov.
```

Pri rule-based optimalizácii môže v kroku (4) eventuelne nastať konflikt pravidiel. Príkladom sú situácie výpočtu GP na joine malých relácií bez dodatočných vlastností alebo výpočtu GP s veľkým počtom agregujúcich atribútov na joine s prítomnosťou cudzieho kľúča/funkčných závislostí. V takomto prípade algoritmus vygeneruje dva grafy, každý reprezentujúci výsledok aplikácie iného pravidla a pridá ich do množiny pracovných grafov. Po aplikovaní transformácií v kroku (6) algoritmus pracuje stále s rovnakým počtom výrazov. Transformácie 3.1.1 - 3.1.3 totiž vždy znamenajú zrýchlenie výpočtu, pretože eliminujú zbytočné operácie. Nie je potrebné teda generovať alternatívne plány.

Pri konštrukcii výrazu pre dotaz narážame na klasický problém optimálneho vyhodnotenia joinov. Táto oblasť je pomerne dobre preskúmaná a existuje niekoľko kvalitných algoritmov na určenie poradia joinov. Mnohé z nich však pracujú s reálnymi selektivitami a navyše nezohľadňujú prítomnosť matematických relácií. Pre naše špecifické prostredie teda zvolíme inú metódu - **pivotizáciu**. Princíp tejto techniky je nasledovný:
Na základe určitých pravidiel vybrať spomedzi sekvencie joinov jeden, nazvime ho **pivot**, ktorý sa vyhodnotí nakoniec. Pivot zároveň rozdelí množinu joinov na dve podmnožiny, ktoré rekurzívne spracujeme. Ak sa v sekvencii nachádza len jediný join, je automaticky pivotom. Takýmto spôsobom pridávame do sekvencie zátvorky a určujeme poradie vyhodnotenia jednotlivých joinov.

Príklad 4.1. Daný je výraz $R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \bowtie R_5$. Za pivota zvoľme $R_2 \bowtie R_3$. Výraz sa transformuje na $(R_1 \bowtie R_2) \bowtie (R_3 \bowtie R_4 \bowtie R_5)$. V ľavej vetve sme narazili na triviálny prípad a algoritmus tu končí. Vrámci pravej vetvy vyberme $R_3 \bowtie R_4$. Tým algoritmus končí aj v tejto vetve a spracovaný výraz má tvar $(R_1 \bowtie R_2) \bowtie (R_3 \bowtie (R_4 \bowtie R_5))$.

Algoritmus je určený pre nerekurzívne dotazy, ktoré sú reprezentované acyklickým grafom. Použitie matematických relácií však situáciu mení. Na

zabezpečenie korektnosti z pohľadu dostupnosti vstupnej množiny v čase joinovania s matematickou reláciou je potrebné špeciálne poradie vyhodnotenia. Jedná sa o prípad, keď vstupná množina nejakej matematickej relácie obsahuje dva atribúty vrámci dvoch rôznych relácií. V prípade, že takáto matematická relácia nie je súčasťou nejakého cyklu, je potrebné ho vyrobiť pridaním fiktívneho JP medzi relácie, ktoré obsahujú atribúty vstupnej množiny matematickej relácie. Ako bolo už spomenuté, všetky fiktívne JP vyhodnocujeme metódou nested-loop. Okrem takto umelo vytvorených cyklov sa v grafe môžu nachádzať aj cykly, ktoré vyplývajú z charakteru dotazu(Príklad 4.2). Na každom cykle v grafe sa teda podieľa matematická relácia a to práve dvomi svojimi atribútmi. Algoritmus zabezpečí sprístupnenie vstupnej množiny pre nekonečné relácie v cykle vyhodnotením joinov relácií obsahujúcich potrebné atribúty pred joinom s matematickou reláciou. Kvôli prehľadnosti je časť algoritmu venovaná detekcii cyklov oddelená od pivotizácie.

Algoritmus(Transformácia hypergrafu)

```
INPUT  : Hypergraf dotazu v modifikovanej relačnej algebre.
OUTPUT : Výraz modifikovanej algebry.
GLOBAL : Množina inštancovaných atribútov

(1) Nájdi matematickú reláciu s atribútmi vstupnej množiny
    vrámci rôznych relácií R1, R2 mimo cyklu
(2) Medzi R1, R2 pridaj fiktívny JP
(3) Ak existuje v grafe cyklus
(4)  Identifikuj v ňom nekonečnú reláciu R obsahujúcu 2 JP
(5)  Prstenec = transformuj zvyšok cyklu
(6)  Rest = transformuj zvyšok grafu
(7)  Vráť (R join Prstenec) join Rest
(8) Inak pivotizáciou transformuj graf na výraz
```

Každý JP je reprezentovaný dvojicou (selektivita, zoznam joinujúcich relácií). V algoritme pivotizácie predpokladáme, že každému JP prislúchajú práve dve relácie. V praxi sa však môže vyskytnúť situácia s viacerými reláciami na jeden JP. Tento prípad redukujeme na predpokladaný nasledovne: Vykonáme pivotizáciu "v malom"na joinoch daného JP, pričom takto vznikne z pôvodného JP niekoľko nových JP, ktoré už obsahujú len dve joinujúce relácie.

Pre algoritmus pivotizácie sú kľúčové dva kroky:

Výber pivota je v súlade s optimalizačnou stratégiou. To znamená, že za pivota vyberieme JP s najhoršou odhadovanou selektivitou. V prípade nejednoznačnosti vezmeme JP zahŕňajúci väčšie relácie. Táto heuristika

predpokladá, že vstupné relácie joinu so zlou selektivitou sa podarí aplikáciou selektívnejších joinov zmenšiť. V prípade nejednoznačnosti uprednostňujeme pri výbere pivota JP reprezentujúci join väčších relácií a join nezahŕňajúci relácie obsahujúce vyžadované atribúty nejakej GP.

Realizácia GP: Aby mohla byť GP vykonaná, musia byť k dispozícii všetky vyžadované atribúty. Vrámci algoritmu je táto podmienka overovaná porovnaním množiny vyžadovaných atribútov GP s množinou inštancovaných atribútov. Ak sú vyžadované atribúty už inštancované, je sémanticky korektné GP vykonať. O optimálnosti tejto operácie však rozhodnú až pravidlá 4.1.1 - 4.1.5.

Algoritmus(Pivotizácia)

```
INPUT  : Acyklický hypergraf dotazu.
OUTPUT : Výraz modifikovanej algebry.

(01)  Ak v grafe neexistuje JP
(02)   Result = jediná reláciu grafu R
(03)   Medzi inštancované atribúty pridaj atribúty R
(04)  Inak
(05)   Nájdi pivotujúci JP J a označ ho
(06)   R1 = prvá joinujúca relácia vrámci J
(07)   R2 = druhá joinujúca relácia vrámci J
(08)   Ak sa v grafe nachádza neoznačený JP
(09)     Result_R1 = transformuj podstrom zo strany R1
(10)     Result_R2 = transformuj podstrom zo strany R2
(11)   Inak
(12)     Medzi inštancované atribúty pridaj atribúty R1, R2
(13)     Ak na R1 existuje GP1, ktorú možno vykonať
(14)      Result_R1 = GP1(R1), inak Result_R1 = R1
(15)     Ak na R2 existuje GP2, ktorú možno vykonať
(16)      Result_R2 = GP2(R2), inak Result_R2 = R2
(17)   Result = (Result_R1) join (Result_R2)
(18)  Ak na Result existuje GP, ktorú možno vykonať
(19)   Result = GP(Result)
(20)  Vráť Result
```

Tvrdenie: Množina plánov generovaná rule-based optimalizáciou má nasledovné vlastnosti:

1. je konzervatívne ohraničená, t.j. počet vygenerovaných plánov je asymptoticky menší ako exponenciálny

2. obsahuje plán, ktorý nie je horší ako naivný plán získaný priamym prepisom SQL dotazu do relačnej algebry

Počet výstupných plánov je určený množstvom situácií, na ktorých vznikne pri rule-based optimalizácii v kroku (4) nejednoznačnosť. Tieto nastávajú pri aplikácii Transformačných pravidiel, ktoré sa týkajú GP v grafe výrazu. V prevažnej väčšine praktických dotazov je počet GP s agregovanými atribútmi zhora ohraničený číslom $log_2(N)$, kde N je počet zúčastnených relácií. Rádovo na toľkých miestach teda algoritmus vygeneruje alternatívny plán. Tento fakt limituje mohutnosť množiny výstupných výrazov na lineárnu funkciu N.

Optimalizačná stratégia kladie dôraz na vyhodnotenie agregovaných atribútov, ktoré pri skúmaných dotazoch tvorí najvýznamnejšiu časť výpočtu. Úspešnosť prístupu do značnej miery závisí od veľkostí zúčastnených tabuliek. V prípade, že dodatočná réžia skorej agregácie preváži jej úsporu, prikláňa sa optimalizátor ku klasickej(neskorej) variante výpočtu. Aplikácia skorej agregácie mení poradie joinov oproti klasickému plánu. Je preto potrebné zároveň optimalizovať aj túto časť dotazu. Heuristika pivotizácie zohľadňuje dôležité vlastnosti jednotlivých joinov, ako bolo spomenuté vyššie. Algoritmus pracujúci na základe týchto princípov má preto dobré predpoklady generovať plány, ktoré sú minimálne tak dobré, ako klasické dotazy.

4.3 Príklady

V nasledujúcich príkladoch ukážeme fungovanie optimalizačného algoritmu na niekoľkých typických prípadoch dotazov.

Príklad 4.2. Cieľom príkladu je demonštrovať prácu s matematickými reláciami. Daná je relačná schéma $R_1(A, B); R_2(B, C, D, E); R_3(E, F, G); R_4(H, I)$ kde B je cudzí kľúč v R_2 a SQL dotaz

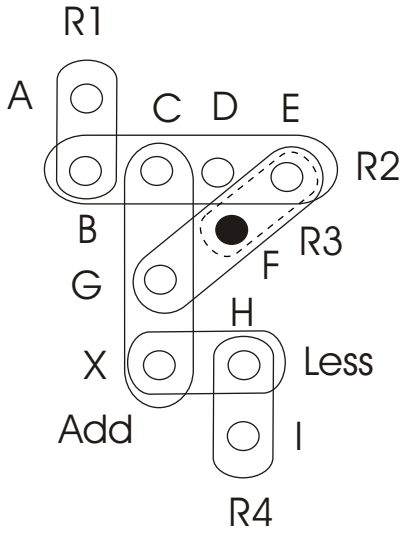
SELECT E, max(F)
FROM R_1, R_2, R_3, R_4
WHERE $(R_1.B = R_2.B)$ AND $(R_1.C + R_2.G < R_4.H)$
GROUP BY E

Priamou transformáciou z naivného prepisu dotazu do klasickej relačnej algebry zapíšeme tento dotaz v modifikovanej algebre ako

$$\Gamma_{\substack{E \\ max(F)}}(R_1 \bowtie R_2 \bowtie R_3 \bowtie Add \bowtie Less \bowtie R_4)$$

Okrem konečných relácií sa vo výraze vyskytujú aj matematické relácie *Add* a *Less*. Prvá z nich je súčasťou cyklu, podľa toho teda algoritmus postupuje pri vyhodnotení. Kvôli *Less* musíme takýto cyklus vyrobiť pridaním fiktívneho JP $Y(0, \{Add, R_4\})$. Ostatné JP sú:

$B(0.5, \{R_1, R_2\})$,
$C(0, \{Add, R_2\})$,
$E(1, \{R_2, R_3\})$,
$G(0, \{Add, R_3\})$,
$H(0, \{Less, I\})$,
$X(0, \{Less, Add\})$



Obrázok 6: Hypergraf dotazu k príkladu 4.2

Simulujme teraz Rule-based optimalizáciu na takomto grafe:
Podľa Pravidla 4.1.5 vykonáme skorú agregáciu Transformáciou 3.2.1. Do grafu pribudne ďalšia GP - dodatočné určenie celkového maxima z parciálnych maxím jednotlivých zoskupení a výraz bude mať tvar

$$\Gamma_{\substack{E \\ max(F)}}(R_1 \bowtie R_2 \bowtie (\Gamma_{\substack{E \\ max(F)}}(R_3)) \bowtie Add \bowtie Less \bowtie R_4)$$

JP E nereprezentuje join na cudzí kľúč, pôvodnú GP teda nemožno odstrániť. Ďalším krokom je vyhodnotenie cyklov:

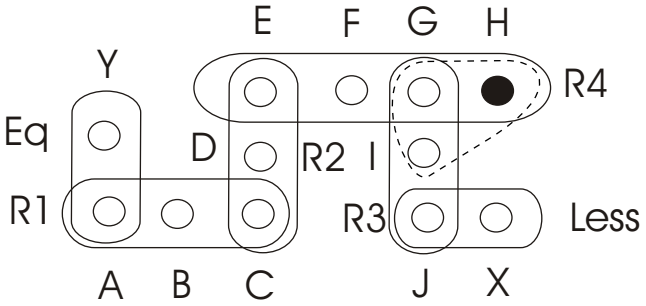
$$\Gamma_{\substack{E \\ max(F)}}(R_1 \bowtie (((R_2 \bowtie (\Gamma_{\substack{E \\ max(F)}}(R_3))) \bowtie Add) \bowtie R_4) \bowtie Less)$$

Nakoniec procesom pivotizácie určíme skoré vykonanie $R_1 \bowtie R_2$, nakoľko tento join má lepšiu selektivitu ako $R_2 \bowtie R_3$. Poradie ostatných joinov je už určené sémantickými ohraničeniami, výsledný výraz má teda tvar:

$$\Gamma_{\substack{E \\ max(F)}}(((((R_1 \bowtie R_2) \bowtie (\Gamma_{\substack{E \\ max(F)}}(R_3))) \bowtie Add) \bowtie R_4) \bowtie Less)$$

Príklad 4.3. V tomto príklade sa zameráme na algoritmus pivotizácie. Daná je relačná schéma $R_1(A, B, C); R_2(C, D, E); R_3(E, F, G, H); R_4(G, I, J)$ kde E je cudzí kľúč v R_3 a SQL dotaz

SELECT G,I, sum(H)
FROM R_1, R_2, R_3, R_4
WHERE $(R_2.C = R_1.C)$ AND $(R_2.E = R_3.E)$ AND $(R_4.J < X)$ AND $(R_1.A = Y)$
GROUP BY G,I



Obrázok 7: Hypergraf dotazu k príkladu 4.3

Počiatočným výrazom dotazu v modifikovanej algebre je

$$\Gamma_{\substack{G,I \\ sum(H)}}((Eq \bowtie R_1) \bowtie R_2 \bowtie R_4 \bowtie (Less \bowtie R_3))$$

Eq a $Less$ sú pomocné matematické relácie testujúce $R_1.A$ na rovnosť a $R_4.J$ na nerovnosť s konštantou. V ďaľšom kroku algoritmus aplikuje na graf Transformáciu 3.2.4. Po nej má zodpovedajúci výraz tvar

$$\Gamma_{\substack{G,I \\ sum(H')}}((Eq \bowtie R_1) \bowtie R_2 \bowtie \Gamma_{\substack{E,G \\ sum(H)}}(R_4) \bowtie \Gamma_{\substack{G,I \\ m=count(*)}}(Less \bowtie R_3))$$

$H' = m * H$ kvôli kompenzácii duplicitných riadkov, ktoré vznikli aplikáciou $\Gamma_{\substack{G,I \\ G,I}}(Less \bowtie R_3)$. Alternatívny výraz, v ktorom nie je potrebné zahrnúť multiplicitu riadkov:

$$\Gamma_{\substack{G,I \\ sum(H)}}((Eq \bowtie R_1) \bowtie R_2 \bowtie \Gamma_{\substack{E,G \\ sum(H)}}(R_4) \bowtie (Less \bowtie R_3))$$

Nakoniec procesom pivotizácie určíme poradie vyhodnotenia joinov. Vzhľadom na charakter jednotlivých joinov volíme za pivotov JP v nasledovnom poradí: Prvým pivotom je C, vo vetve R_1 je to potom automaticky A. Vo vetve R_2 vyberieme za pivota G, výber E a J je ďalej jednoznačný. Po vyhodnotení poradia joinov potom získame nasledovné výstupné výrazy, z ktorých cost-based optimalizátor vyberie lepší

$$\Gamma_{\substack{G,I \\ sum(m*H)}}((Eq \bowtie R_1) \bowtie ((R_2 \bowtie \Gamma_{\substack{E,G \\ sum(H)}}(R_4)) \bowtie \Gamma_{\substack{G,I \\ m=count(*)}}(Less \bowtie R_3)))$$

$$\Gamma_{\substack{G,I \\ sum(H)}}((Eq \bowtie R_1) \bowtie ((R_2 \bowtie \Gamma_{\substack{E,G \\ sum(H)}}(R_4)) \bowtie (Less \bowtie R_3)))$$

Reálny prínos optimalizácie teraz demonštrujeme zjednodušenou analýzou niektorých dotazov z praxe.

Príklad 4.4. Daná je relačná schéma
Objednavka(idProd,idPred,mnozstvo);
Predajca(id,mesto);
Produkt(id,oddelenie);
kde idProd a idPred sú cudzími kľúčmi v relácii Objednavka a dotaz

SELECT idProd,idPred,sum(mnozstvo)
FROM Objdenavka, Predajca, Produkt
WHERE (Predajca.mesto='Bratislava') AND (Produkt.oddelenie='Doprava')
AND (Objednávka.idProd=Produkt.id) AND (Objednávka.idPred=Predajca.id)
GRUP BY idProd,idPred

Klasický spôsob vyhodnotenia je

$$\Gamma_{\substack{sum(mnozstvo) \\ idProd,idPred}}((Predajca \bowtie Produkt) \bowtie Objednavka)$$

s príslušnými selekciami realizovanými pred jednotlivými joinami. Vezmime praktickú situáciu, že relácia Objednavka je omnoho väčšia ako relácie Produkt a Predajca (veľkosti sú typicky rádovo postupne $10^5, 10^2, 10^1$). Keďže sú na to splnené všetky podmienky, môžeme aplikovať Transformáciu 3.2.1

a výpočet grúp vykonať na relácii Objednavka pred joinami s reláciami Predajca a Produkt

$$(\Gamma_{\substack{sum(mnozstvo)\\idProd,idPred}}(Objednavka) \bowtie Predajca) \bowtie Produkt$$

Nakoľko relácia Predajca je relatívne malá, dostávame pomocou pivotizácie a Pravidla 4.1.1 alternatívny plán

$$\Gamma_{\substack{sum(mnozstvo)\\idProd,idPred}}(Objednavka) \bowtie (Predajca \bowtie Produkt)$$

Pretože relácie Predajca a Produkt sú oproti relácii Objednavka o niekoľko rádov menšie, skorá agregácia prinesie zlepšenie času výpočtu. Konkrétne, v relácii Objednavka sa môže nachádzať rádovo maximálne $10^2 * 10^1 = 10^3$ záznamov s rôznou hodnotou atribútov idProd alebo idPred. To podľa Dirichletovho princípu znamená, že aplikovaním skorej agregácie sa táto relácia zmenší minimálne 100-násobne. Takéto zmenšenie s najväčšou pravdepodobnosťou preváži selektivitu joinov s reláciami Predajca resp. Produkt. Optimálnosť jednotlivých plánov potom záleží od reálnych selektivít daných joinov.

Príklad 4.5. Majme teraz nasledovnú relačnú schému:
Objednavka(idProd,idPred,mnozstvo);
Produkt(id,idOdd);
Oddelenie(id,idSek);
s veľkosťami relácií rádovo $10^5, 10^2, 10^1$ záznamov a nad ňou dotaz

```
SELECT idProd,idSek,idOdd,sum(mnozstvo)
FROM Oddelenie,Objednavka,Produkt
WHERE (Objednavka.idProd=Produkt.id) AND (Produkt.idOdd=Oddelenie.id)
AND(Objednavka.idPred='3')
GROUP BY idProd,idSek,idOdd
```

Klasický plán vyhodnotenia bez ohľadu na selekcie(kvôli jednoduchosti) je

$$\Gamma_{\substack{sum(mnozstvo)\\idProd,idSek,idOdd}}((Objednavka \bowtie (Produkt \bowtie Oddelenie))$$

Ak chceme použiť skorú agregáciu v tomto prípade, musíme ju najskôr rozdeliť do kaskády podľa Tranasfomácie 3.1.4. Po vykonaní rule-based optimalizácie budeme mať potom k dispozícii nasledovné plány:

$$\Gamma_{\substack{sum(mnozstvo)\\idProd,idSek,idOdd}}(\Gamma_{\substack{sum(mnozstvo)\\idProd,idOdd}}(\Gamma_{\substack{sum(mnozstvo)\\idProd}}(Objednavka) \bowtie Produkt) \bowtie Oddelenie)$$

$$\Gamma_{\substack{sum(mnozstvo)\\idProd,idSek,idOdd}}(\Gamma_{\substack{sum(mnozstvo)\\idProd}}(Objednavka) \bowtie (Produkt \bowtie Oddelenie))$$

V oboch prípadoch je relácia Objednavka zmenšená minimáne o dva rády skorým vykonaním GP. Následné joiny do ktorých vstupuje už operujú z menším počtom riadkov, čo značne urýchli výpočet. Ktorý z nich bude nakoniec použitý na vyhodnotenie dotazu, určí cost-based optimalizátor.

Viac praktických príkladov možno nájsť v [02].

5 Použitá literatúra

- [01] J.D. Ullman: Principles of database and knowledge systems
- [02] A. Gupta, V. Harinarayan, D. Quass: Generalized projections: A powerful approach to aggregation
- [03] S. Chaudhuri, K. Shim: Including group-by in query optimization
- [04] W. P. Yan, P. Larson: Eager aggregation and lazy aggregation
- [05] W. P. Yan, P. Larson: Interchanging the order of grouping and join
- [06] G. N. Paulley: Exploiting functional dependence in query optimization