Ročníkový projekt - report - zimný semester

Úvod

Pôvodným zadaním ročníkového projektu bolo implementovať operátory relačnej algebry s materializáciou výsledkov v pamäti. V priebehu semestra sa však ukázalo zaujímavejšie najskôr zjednodušiť sadu operátorov tak, aby bola stále úplná (dokázala robiť všetko to, čo predošlá implementácia relačnej algebry) a zároveň umožnila flexibilnejšie rozhranie operátorov (n-árne operátory miesto iba binárnych, zbavenie sa nutnosti špecifikovať joinovaciu podmienku v operátoroch Join a Antijoin). Toto sa teda na zvyšok semestra stalo cieľom ročníkového projektu.

Motivácia

Nedostatkom pôvodnej implementácie relačnej algebry bolo, že operátory Join a Antijoin dostávali joinovaciu podmienku v konštruktore (tupleTrans), t.j. fungovali ako theta-join, pričom z pohľadu implementácie (a pravdepodobne aj použitia) je jednoduchšie obmedziť ich na prirodzený Join resp. Antijoin. Okrem toho boli všetky operátory implementované ako binárne, čo zbytočne zvyšuje zložitosť a komplikuje optimalizáciu výpočtov.

Priebeh práce počas semestra

Práca spočívala najmä v návrhu redukovanej sady operátorov relačnej algebry a ich vstupných argumentov:

1. operátory Join a Antijoin majú byť prirodzené (bez explicitne zadanej podmienky na rovnosť atribútov)
2. operátory Join, Antijoin a Union majú implicitne zahŕňať funkčnosť operátorov Atov, Vtoa (Atov a Vtoa operátory už nie sú potrebné)
3. všetky operátory (Join, Antijoin a Union) potrebujú dostávať vstupné a výstupné “vzory“ - vektory termov, podľa ktorých budú robiť Atov, Vtoa matching (a operátory Join a AntiJoin z nich zistia joinovaciu podmienku).
4. operátory majú byť zovšeobecnené z binárnych na n-árne

Príklad 1

Majme relácie $r_1(A, B), r_2(B, C), r_3(C, D)$ zadané tabuľkou:

r_1		r_2		r_3	
1	2	2	4	4	7
2	3	3	5	2	6

Majme dotaz *q*, ktorý je v relačnej algebre (s pôvodnou sadou operátorov) zadaný ako:

$s := (r_1 \bowtie r_2) \bowtie (r_2 \bowtie r_3)$
$q := vtoa(q(1, X, 4, Y), s)$

Ide teda o dotaz na všetky dvojice (X, Y) ktoré sa po prirodzenom joine všetkých 3 relácií nachádzajú vo výstupnej štvorici (1, X, 4, Y). V Datalogu by sme dotaz vyjadrili ako:

$\mathrm{q(A, B, C, D)} \leftarrow \mathrm{r1(A, B)}, \mathrm{r2(B, C)}, \mathrm{r3(C, D)}.$
? q(1, X, 4, Y)

S použitím pôvodnej implementácie relačnej algebry v Jave by pseudokód programu realizujúceho daný dotaz vyzeral nasledovne:

```
s = Join(
        Join(
                r1, r2,
                (t1, t2) -> t1[1] == t2[0] ?  {t1[0], t1[1], t2[1]} : null
        ),
        Join(
                r2, r3,
                (t1, t2) -> t1[1] == t2[0] ?  {t1[0], t1[1], t2[1]} : null
        ),
        (t1, t2) -> t1[1] == t2[0] && t1[2] == t2[1] ?  {t1[0], t1[1], t1[2], t2[2]} : null
)
q = Vtoa(
        [X, Y],             // 'head' := termy, ktoré sa majú nachádzať vo výstupe
        s,                  // 'inner' := vnútorná relácia
        [1, X, 4, Y]        // 'variables' := poradie premenných vo vnútornej relácii
)
```

Ten istý dotaz sa dá v novej implementácii relačnej algebry vyjadriť ako:

```
Join(
        [r1, r2, r3],
        [[A, B], [B, C], [C, D]],
        [1, B, 4, D]
)
```

Je vidieť podstatné zjednodušenie, ktoré je možné vďaka tomu, že:

1. Join je implementovaný ako všeobecný (n-árny)
2. výstupná selekcia a projekcia sa nerobí explicitne pomocou operátora Vtoa, ale implicitne priamo v Joine podľa premenných vo vstupnom a výstupnom vektore
3. nemusíme explicitne špecifikovať joinovaciu podmienku, operátor urobí prirodzený join podľa premenných špecifikovaných vo vstupnom vektore

Operátor Join v príklade 1 bude pri volaní metódy next() fungovať nasledovne:

1. Nájde nasledujúci prvok kartézskeho súčinu relácií r1, r2, r3, ktorý spĺňa matching podľa vzoru vo vstupných vektoroch ([A, B, B, C, C, D]).
2. Ak taký nájde, aplikuje vtoa podľa výstupného vektora (v tomto prípade [1, B, 4, D]); inak vráti null.

Príklad 2

Majme binárnu reláciu r, unárnu reláciu s a datalogový program s dotazom:

$p(X) \leftarrow r(\text{‘a’}, Y).$
$p(X) \leftarrow r(X, Y), \neg s(X).$
? p(X)

Tento program by sa v pôvodnej implementácii vyjadril nasledovne:

```
p1 = Atov(
        [‘a’, Y],
        r,
        [Y]
)
p2 = Vtoa(
        [X],
        AntiJoin(
                r, s,
                (t1, t2) -> t1[0] == t2[0] ? null : t1
        ),
        [X, _ ]
)
p = Union(p1, p2)
```

Teraz ho vyjadríme ako:

```
p2 = AntiJoin(
        [r, s],
        [[X, Y], [X]],
        [X]
)
p = Union(
        [r, p2],
        [[‘a’, X], [X]],
        [X]
)
```

S pôvodnou sadou operátorov Atov selektuje z relácie r tie riadky, ktoré majú atribút X rovný konštante ‘a’ (funkčný symbol s aritou 0), a následne urobí projekciu na atribút Y. S novou sadou sa táto selekcia a projekcia vykoná implicitne v operátore Union.

Operátor Union v príklade 2 bude fungovať nasledovne:

Postupne pri volaní metódy next():
1. Kým sa v relácii r nachádzajú ďalšie záznamy (tuples) spĺňajúce vzor (‘a’, X) nájde “najbližší“ taký záznam (tak, ako by to urobil pôvodný operator atov)
2. Na výstup dá tento záznam, na ktorom bola aplikovaná substitúcia podľa výstupného vektora (tak, ako by to urobil pôvodný operátor vtoa)
3. Keď sa spracuje celá relácia p1, postupne dá na výstup všetky záznamy z p2, pretože všetky spĺňajú vstupný vzor pre p2 (X) a výstupný vzor je identický (X)

Operátor AntiJoin zjednodušene bude fungovať analogicky (pre každý riadok sa vykoná atov > antijoin > vtoa)

Záver

Práca počas semestra pozostávala z veľkej časti v hľadaní “správnej” sady operátorov a reprezentácie ich argumentov. Metódou “overenia správnosti” boli myšlienkové experimenty, t.j. úvahy o realizácii výpočtov vybraných dotazov v rôznych algebrách. Zdrojové kódy k implementácii zvolenej algebry boli až sekundárnym výstupom, preto sa ich dokončením budeme zaoberať v budúcom semestri. Ďalší postup môže tiež spočívať v ešte silnejšej redukcii operátorovej sady relačnej algebry (v extréme možno až na jeden všeobecný operátor).