Ročníkový projekt Datalog->RA

popis vývoja projektu v prvom semestri

Cielom projektu v tomto semestri bolo sfunkčniť program Ramková databáza a začať v ňom manuálne prekladať niektoré jednoduché programy. Takto som chcel získať prehľad v tom, čo bude potrebné na automatické prekladanie datalogu do relačnej algebry.

- Program Ramková databáza bol vytvorený v jednom z bývalých ročníkových projektov. Jeho účelom bolo realizovať vybrané operátory relačnej algebry bez použitia pamäte na ukladanie medzivýsledkov výpočtu.

Nanešťastie som nemal prístup k poslednej verzii projektu Ramkovej Databázy a tak som sa ju rozhodol naprogramovať nanovo s obmenami, ktoré budú užitočné pri neskoršom preklade. Na funkcionalitu programu boli nasledovné požiadavky:

- Program pracuje nad reláciami databázy

  Elementy relácií sú riadky (tuple) a elementy tuplov sú Atribúty, ktoré reprezentujú vždy string charakterov.

  Relácie neobsahujú duplikáty (2 rovnaké tuple)

  Prvky relácie sa nedajú meniť alebo mazať

  Tuple ani Atribúty nemôžu byť null
- Program realizuje oprátory relačnej algebry Join, Selekcia, Projekcia, Union a Antijoin

  Operátory sú triedami, z ktorých sa dajú robiť inštancie „iterátorov“, podľa toho nad akými reláciami a podmienkami pracujú, majú verejnú metódu next(), ktorá vždy vráti nasledujúci riadok výslednej relácie
- Dve inštancie toho istého operátora alebo relácie sú na sebe nezávyslé

  Za týmto účelom majú Operátory aj Relácie metódu instance(), ktorá vráti ich kópiu a táto kópia vracia metódou next() výsledky odznova, nezávysle od pôvodného Operátora/Relácie
- Program si neukladá medzivýsledkzy

  Pamäť, ktorú program využiva je (až na potreby inicializácie Operátorov) iba tak veľká ako tabuľky Relácií, s ktorými pracuje. Ak je potrebné zistiť nejaký výsledok Operátora znovu, program odsimuluje Operátor nanovo.
- Operátory nevytvárajú duplikáty

  Nakoľko jazyk datalogu nepočíta s prítomnosťou duplikátov, ani môj program vôbec nevyrába duplikáty. Pôvodná myšlienka projektu bola odstrániť duplikáty v každom Operátore tak, že metóda next() odsimuluje priebeh programu po momentálny stav a ak nájde duplikát, tak výsledok pokladá za nesprávny a hľadá ďalší. Namiesto toho sa Relácie zaviedli bez duplikátov a odstraňovanie duplikátov prebieha iba v Operátoroch, ktoré sú schopné ich vytvoriť-Union a Projekcia

Odstraňnovanie objemovej složitosti na zásobníku:

- Pôvodný projekt môjho predchodcu realizoval metódu next() rekurzívne a to tak, že pri každom Tuple, ktorý nevyhovoval podmienkam Operátora sa zavolala metóda next() nanovo

  Namiesto tohto prístupu som sa rozhodol prerobiť metódu next() na cyklickú aby som zabránil stack overflow na väčších vstupoch s malým množstvom výsledkov
- Pôvodnou ideou odstraňovania duplikátov bolo odsimulovať doterajší priebeh Operátora presne tak ako prebiehal prvý krát. To by však znamenalo, že počet súčasných spustení odstraňovania duplikátov by bol priamo úmerný dĺžke tabuľky s ktorou Operátor pracuje. Toto by pri väčších vstupoch spôsobovalo stack overflow.

  Za účelom predísť tomuto problému vznikla metóda nonDistinctNext(), ktorá nekontroluje prítomnosť duplikátov v Operátoroch, ktoré sú simulované vyššie uvedeným spôsobom. Odstraňovanie duplikátov teda volá vždy nonDistinctNext() a nie next() a metóda nonDistinctNext() tiež nikdy nevolá next(), čím sa odstáni aj časť časovej zložitosti.

TupleTransformation:

- Pre väčšiu univerzalitu Operátorov sme sa s vedúcim projektu rozhodli správnosť výsledku rozhodovať pomocou triedy TupleTransformation.
- Táto trieda má public metódu transform(Tuple), ktorá vráti Tuple ak má ísť na výsledok Operátoru alebo null v opačnom prípade.
- Toto nám umožnilo spojiť projekciu a selekciu do jedného operátora a ak sa tak v budúcnosti rozhodneme, je možné pridať projekciu ku každému inému operátoru.

Testovanie:

- Nakoniec som vytvoril niekoľko krátkych testov, ktoré odhalili malé chyby v operátoroch. Tieto chyby sa podarilo veľmi rýchlo odstrániť.
- Testovanie mi pomohlo nahlidnuť na to, akú funkcionalitu budú musieť mať výsledné triedy TupleTranformation aby bolo možné urobiť automatický preklad.