

Ročníkový projekt Datalog→RA

popis vývoja projektu v prvom semestri

Cieľom projektu v tomto semestri bolo sfunkčniť program Ramková databáza (RAMD), ktorý urobil P. Becza v ročníkovom projekte v 2014/2015, a začať v ňom manuálne prekladať niektoré jednoduché Datalogové programy. Takto som chcel získať prehľad v tom, čo bude potrebné na automatické prekladanie Datalogu do relačnej algebry.

Účelom programu RAMD bolo realizovať vybrané operátory relačnej algebry nad reláciami uloženými v RAM, bez použitia dodatočnej pamäte na ukladanie medzivýsledkov výpočtu.

Keďže som spočiatku nemal prístup k poslednej verzii RAMD, rozhodol som sa implementovať RAMD nanovo s obmenami, ktoré budú užitočné pri neskoršom preklade Datalogových programov.

Požiadavky na funkcionálnosť programu

- **Program pracuje nad reláciami databázy, ktoré sú uložené v RAM.**

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

Relácie neobsahujú duplikáty (2 rovnaké tuple).

Prvky relácie nie je potrebné meniť alebo mazať.

Tuple ani Atribúty nikdy nemajú hodnotu null.

Relácie ponúkajú iterátor s verejnou metódou next(), ktorá umožňuje postupne vymenovať všetky Tuples relácie.

- **Program realizuje operátory relačnej algebry Join, Selection, Projection, Union a Antijoin.**

Tieto operátory sú implementované ako triedy (class), z ktorých sa dajú robiť inštancie „iterátorov“, pričom všetky ponúkajú verejnú metódu next(), ktorá vracia nasledujúci Tuple výslednej relácie.

- **Dve inštancie toho istého operátora alebo relácie sú na sebe nezávislé.**

Za týmto účelom majú Operátory aj Relácie metódu instance(), ktorá vráti ich kópiu

a táto kópia vracia metódou next() výsledky odznova, nezávisle od pôvodného Operátora/Relácie.

- **Program si neukladá medzivýsledky.**

Pamäť, ktorú program využíva je (až na potreby inicializácie Operátorov) iba taká veľká ako databázové relácie, s ktorými pracuje. Ak je potrebné zistiť nejaký výsledok Operátora znovu, program odsimuluje Operátor nanovo.

- **Operátory nevytvárajú duplikáty.**

Nakoľko Datalog nepočíta s prítomnosťou duplikátov, ani môj program 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. Relácie neobsahujú duplikáty.

Odstraňovanie duplikátov prebieha iba v Operátoroch, ktoré sú schopné ich vytvoriť (Union a Projection).

Redukcia pamäťovej zložitosti na zásobníku

- Pôvodný projekt RAMD môjho predchodcu realizoval metódu next() rekurzívne tak, že pri každom Tuple, ktorý nevyhovoval podmienkam Operátora sa rekurzívne zavolala metóda next(). Túto rekurziu som nahradil cyklom, 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 veľkosti relácie, 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(). Týmto sa dosiahne, že veľkosť zásobníka nie je úmerná veľkosti databázy. Taktiež sa tým niekedy zredukuje aj časová zložitosť.

TupleTransformation

- Pre väčšiu univerzalitu Operátorov sme sa rozhodli podmienky v selekciách a projekciách testovať uniformným spôsobom, pomocou triedy TupleTransformation.
- Trieda TupleTransformation 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

- Vytvoril som niekoľko krátkych testov, ktoré odhalili jednoduché chyby v operátoroch. Tieto chyby sa podarilo veľmi rýchlo odstrániť.
- Testovanie mi tiež pomohlo nahliadnuť na to, akú funkcionálnosť bude musieť mať trieda TupleTransformation, aby bolo možné urobiť automatický preklad z Datalogu.