Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.9
DOI: https://doi.org/10.24867/29BE10Gravara

MiniOptimizer: ПОЈЕДНОСТАВЉЕНИ РЕЛАЦИОНИ ОПТИМИЗАТОР УПИТА

MiniOptimizer: A SIMPLIFIED RELATIONAL QUERY OPTIMIZER

Милош Гравара, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У овом раду је представљен MiniOptimizer, поједностављени релациони оптимизатор упита, са циљем да практично прикаже теоријске основе оптимизације упита. MiniOptimizer, развијен у програмском језику C#, подржава основне SQL операције и користи MiniQL, специјално дизајнирани упитни језик. Рад обухвата структуру алата MiniOptimizer, укључујући парсер, логички и физички план извршења, проценитељ кардиналности и каталог података. Процес оптимизације укључује парсирање MiniQL упита, креирање и трансформацију логичког плана, оптимизацију редоследа спајања и избор најповољнијег физичког плана извршења. Један од доприноса овог рада јесте разумевање техника оптимизације упита, кроз развој MiniOptimizer-а као едукативног алата који може да служи као основа за даља истраживања и развој напреднијих система за управљање базама података.*

Кључне речи: *Релациони оптимизатор упита, SQL, процена кардиналности, системи за управљање базама података*

Abstract – *This paper presents MiniOptimizer, a simplified relational query optimizer, with the aim of practically demonstrating the theoretical foundations of query optimization. Developed in the C# programming language, MiniOptimizer supports basic SQL operations and uses MiniQL, a specially designed query language. The paper covers the structure of the MiniOptimizer tool, including the parser, logical and physical execution plan, cardinality estimator, and data catalog. The optimization process includes parsing MiniQL queries, creating and transforming a logical plan, optimizing the join order, and choosing the most favorable physical execution plan. One of the contributions of this work is enhancing the understanding of query optimization techniques through the development of MiniOptimizer as an educational tool, which can serve as a foundation for further research and the development of more advanced database management systems.*

Keywords: *Relational query optimizer, SQL, cardinality estimation, database management systems*

1. УВОД

Током седамдесетих година прошлог века, дошло је до значајног искорака у свету управљања над подацима развојем декларативних система. Претходно, процес претраживања и преузимања података претежно је био вођен процедуралним системима, који су захтевали од корисника да пажљиво специфицирају сваки корак у процесу добијања потребних података. Овакав приступ се често показао као непрактичан и неефикасан, што је представљало значајне изазове у руковању сложеним скуповима података. Са друге стране, декларативни системи омогућили су корисницима да, постављањем упита, специфицирају само шта желе да постигну, без детаљног описа како то да постигну. Овако постављени, декларативни системи су понудили интуитивнији и ефикаснији начин интеракције са базама података, превазилазећи ограничења својих процедуралних претходника, и омогућили корисницима да се фокусирају на пословну логику уместо на имплементационе детаље.

Ипак, како би се могле развијати апликације које користе велике количине података са циљем опслуживања корисничких захтева, декларативност није довољна, већ је неопходно да системи за управљање базама података (СУБП) буду ефикасни, ради смањења времена чекања на одговор захтева и пружања пријатног корисничког искуства. С тим на уму, оптимизација перформанси базе података постала је централна активност која омогућава кратко време одзива на корисничке захтеве за подацима. Главну улогу у овом процесу оптимизације игра оптимизатор упита, компонента задужена за анализирање постављених упита и одређивање оптималног начина њиховог извршавања. Узимајући у обзир различите факторе као што су структура упита, шема базе података и системски ресурси, оптимизатор упита смишља план извршења, којим употребу системских ресурса своди на минимум, истовремено пружајући најбоље перформансе.

Циљ овог рада јесте да се, на практичном примеру имплементације оптимизатора упита, пружи детаљан увид у улоге и механизме процеса оптимизације упита у релационим СУБП. Како би се то постигло, креиран је *MiniOptimizer*, алат написан у програмском језику C# који илуструје основне концепте оптимизације упита у релационим системима за управљање базама података. Пажња је усмерена на логички аспект оптимизације - генерисање ефикасних планова извршења применом техника оптимизације засноване на правилима и оптимизације засноване на трошку са

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Владимир Димитриески, ванр. проф.

избором оптималног редоследа спајања коришћењем алгоритма динамичког програмирања.

2. АНАТОМИЈА ИЗВРШЕЊА УПИТА

Улазна тачка сваког упита представљеног СУБП-у је парсер. Задатак парсера је да, најпре, изврши лексичку и синтаксну анализу упита, како би проверио да упит подлеже правилима граматике упитног језика, а затим да га трансформише у апстрактно синтаксно стабло које представља хијерархијску структуру упита.

Након тога, на основу апстрактног синтаксног стабла се формира иницијални план упита, типично представљен усмереним ацикличним графом оператора релационе алгебре. Граф креиран на основу оператора релационе алгебре се назива логички план упита. Овакав логички план се, применом одређеног скупа правила, трансформише у еквивалентне логичке планове за које се очекује да ће захтевати мање времена за извршење [1].

Међутим, логички план је апстрахован од детаља физичког приступа подацима, као и алгоритама којима се реализују релациони оператори. Из тог разлога неопходно га је превести у физички план упита који СУБП може да изврши [1]. Физички план извршења упита дефинише избор алгоритама релационих оператора на нивоу приступа подацима и одређује редослед њиховог извршења. Поред тога, физички план садржи детаље о томе како треба да се приступа релацијама базе података, као и о томе да ли и када их је потребно сортирати. Као и логички план извршења, представљен је као усмерени ациклични граф.

У оквиру избора оптималног логичког и физичког плана, оптимизатор упита мора да донесе неколико одлука. Најпре, неопходно је одредити које алгебарски еквивалентне форме упита доводе до најефикаснијег избора алгоритама извршења упита [1]. Затим, за сваки релациони оператор потребно је изабрати најповољнији алгоритам имплементације. На крају, неопходно је донети одлуку о томе на који начин оператори треба да прослеђују податке једни другима, у виду проточне обраде, употребом бафера радне меморије или путем диска.

Након што се одабере најповољнији физички план упита, СУБП га извршава и враћа резултате кориснику који је упит специфицирао.

3. ПРЕГЛЕД СТАЊА У ОБЛАСТИ

Историја релационих оптимизатора упита обухвата развој неколико кључних система који су значајно унапредили технике оптимизације и поставили основе за савремене алате.

Селинџеров *R* оптимизатор [2], развијен у оквиру Систем *R* пројекта од стране *ИБМ*-а током 1970-их, уводи динамичко програмирање за генерисање оптималног плана извршења упита користећи статистичке информације о подацима.

Volcano оптимизатор [3], развијен крајем 1980-их од стране Геца Грефеа, користи генерализовани приступ оптимизацији, укључујући стратегију сепарације и евалуације (енгл. *branch and bound*) и подршку за паралелизам и процену трошкова на више нивоа.

Балса оптимизатор [4], заснован на моделу подржаног учења (енгл. *reinforcement learning*), представља значајан помак од традиционалних метода. Балса континуирано ажурира своју базу знања из стварних повратних информација о извршењу упита и на тај начин се ефикасно прилагођава променама у дистрибуцији података и различитим шаблонима упита, стварајући тако ефикасније планове извршења.

4. ИМПЛЕМЕНТАЦИЈА АЛАТА *MiniOptimizer*

MiniOptimizer је поједностављени релациони оптимизатор упита чија је основна намена да илуструје основне теоријске концепте који стоје иза процеса оптимизације упита. Алат је написан у програмском језику C#, који је изабран због својих напредних механизама апстракције, што омогућава лакше моделовање концепата кључних за правилно функционисање оптимизатора упита, који је инхерентно логички оријентисан.

MiniOptimizer подржава основне SQL операције као што су филтрирање, спајање, пројекција и дистинкција. Ови конструкти су представљени путем *MiniQL* упитног језика развијеног за потребе овог пројекта.

Због сложености пројекта, *MiniOptimizer* не укључује компоненту СУБП-а која имплементира физичке операторе и директно извршава упите. Уместо тога, пажња је усмерена на логичком аспекту оптимзације, где се истражују различите стратегије и технике за генерисање ефикасних планова извршења упита.

4.1. Архитектура алата *MiniOptimizer*

Архитектура *MiniOptimizer* алата обухвата неколико компоненти које омогућавају избор најефикаснијих планова извршења упита и приказана је на слици 4.1.

Након што корисник представи упит *MiniOptimizer*-у, прва станица на коју стиже је парсер упита. Задатак парсера је да, најпре, изврши лексичку и синтаксну анализу, затим користећи каталог података да провери семантичку валидност упита и, на крају, да представљени упит написан у *MiniQL* језику претвори у иницијални логички план.

Иницијални логички план се прослеђује оптимизатору заснованом на правилима, који примењује скуп алгебарских закона чиме се логички план трансформише у ефикаснији план и затим прослеђује компоненти за оптимизацију операција спајања.

Компонента за оптимизацију операција спајања, уз употребу модела трошка, бира најповољнији редослед извршења операција спајања и такав преуређени логички план прослеђује оптимизатору заснованом на трошку.

Оптимизатор заснован на трошку, такође уз употребу модела трошка, који ће бити детаљно објашњен у поглављу 4.5, генерише оптималан физички план који у себи садржи детаље о начину приступа подацима, редоследу извршења операција и конкретним алгоритмима имплементације релационих оператора. Овако конструисан физички план се враћа кориснику који је алату представио *MiniQL* упит.

4.1. Каталог података

Каталог података у *MiniOptimizer* алату је централно складиште метаподатака о бази података.

Слика 4.1 – Архитектура алата *MiniOptimizer*

Он садржи податке о табелама, колонама, индексима и статистикама. Ови подаци укључују структуру табела, типове података колона, доступне индексе и статистичке податке као што су број редова и дистрибуција вредности.

4.2. Парсер и упитни језик *MiniQL*

Парсер *MiniOptimizer*-а развијен је употребом алата *ANTLR4* и његов основни задатак је анализа и обрада улазних упита писаних у *MiniQL* језику. Лексичка и синтаксна анализа се врши у оквиру алата *ANTLR4* спрам граматике *MiniQL* језика представљене на листингу 4.1.

```
query ::= SELECT [DISTINCT] attrList FROM relList [WHERE condition]
attrList ::= attribute [,attribute]*
relList ::= relation [,relation]*
condition ::= condition AND condition | attribute EQ attribute
attribute ::= identifier | constant
relation ::= identifier
```

Листинг 4.1 – Део граматике *MiniQL* упитног језика

Са друге стране, семантичка анализа се врши уз употребу каталога података и основни задатак овог корака је да се обезбеди да све табеле и колоне специфициране у упиту постоје у каталогу података, чиме се умањује потреба за провером на грешке у даљим корацима оптимизације упита.
Након што упит прође семантичку анализу, парсер генерише иницијални логички план рекурзивним обиласком апстрактног синтаксног стабла које је генерисао алат *ANTLR4*.

4.3. Логички план

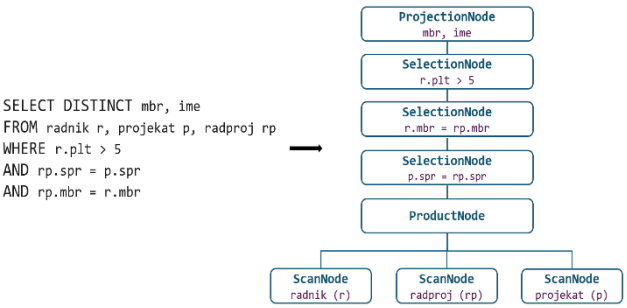
Логички план представља алгебарски израз који описује начин извршења упита над подацима. У *MiniOptimizer*-у, представља структуру усмереног ацикличног графа моделовану кроз скуп међусобно повезаних логичких чворова, где сваки чвор поседује листу потомака и референцу на родитељски чвор.

4.5. Модел трошка

Основни циљ модела трошка у *MiniOptimizer* алату је да пружи брзе и прецизне процене кардиналности и процене трошка извршења одређених оператора без њиховог физичког извршавања. Ова компонента садржи два основна задатка: Процена кардиналности и процена трошка. Оба поступка се заснивању на статистичким подацима садржаним у каталогу.
Значај овакве структуре лежи у могућности за ефикасно премештање чворова са једног дела графа на други, чиме се подржавају хеуристике за побољшање логичких планова. Поред тога, логички чворови поседују и информације о кардиналности резултата оператора које представљају, што је значајно током процеса процене кардиналности међурелација приликом оптимизације редоследа операције спајања.

4.4. Оптимизатор заснован на правилима

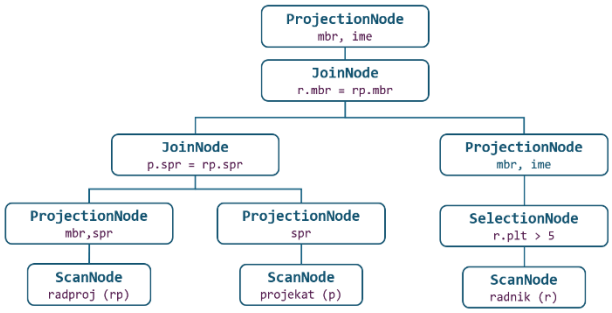
Иницијални логички план се састоји од релационог оператора пројекције који је настао на основу `SELECT` клаузуле и који као потомке садржи ланац оператора селекције који се формирају уколико је *MiniQL* упит садржао `WHERE` клаузулу. Ланац оператора селекције као потомка садржи оператор декартовог производа који се односи на све релације поменуте након `FROM` клаузуле упита. Пример иницијалног логичког плана приказан је на слици 4.2.



Слика 4.2 – Пример инцијалног логичког плана

Задатак оптимизатора заснованог на правилима је да трансформише иницијални логички план у еквивалентан али ефикаснији облик. Ово постиже применом неколико правила у одређеном редоследу. Најпре се врши креирање операција спајања где се за сваке две релације креира одговарајући релациони оператор споја, чиме се замењује мање ефикасни декартов производ. Затим се врши пропагација селекције, при чему се оператори селекције померају ближе изворним релацијама. На крају, долази до креирања операција дистинкције и репликације оператора пројекције како би се смањио број података који пролази кроз систем. Резултат оптимизације засноване на правилима је ефикаснији логички план извршења који је спреман за фазу уређивања редоследа спајања. Слика 4.3 приказује логички план након оптимизације засноване на правилима.

Под проценом кардиналности се сматра поступак процене величине резултујуће релације након примене одређеног оператора релационе алгебре. Поступак процене кардиналности је неопходан приликом одређивања оптималног редоследа спајања.



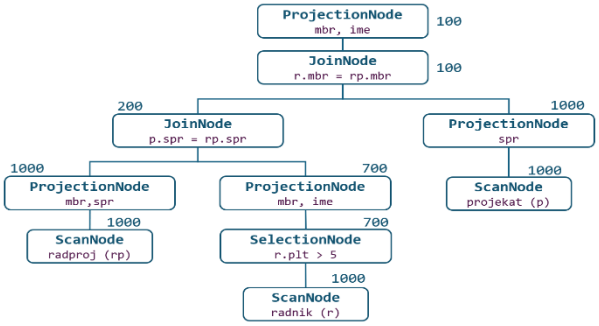
Слика 4.3 – Оптимизован логички план

Са друге стране, процена трошка представља поступак процене количине улазно-излазних операција ка диск јединици СУБП-а. Поступак процене трошка је неопходан приликом избора најповољнијих физичких оператора.

4.6. Оптимизатор операција спајања

Задатак оптимизатора операција спајања је да пронађе оптималан редослед спајања релација у упиту. Проблем проналаска оптималног редоследа у *MiniOptimizer* алату је решен применом алгоритма динамичког програмирања.

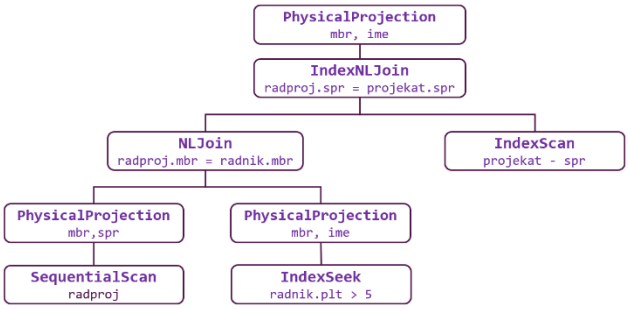
Алгоритам динамичког програмирања се састоји из неколико корака. Први корак представља иницијализацију табеле динамичког програмирања где се за сваке две релације процени кардиналност њиховог спајања. Затим се, итеративно, за сваки скуп релација који садржи више од две релације проналази њихов оптималан редослед спајања тако што се бира онај редослед чија процена даје најмању кардиналност. Овај процес се врши док се не дође до величине плана који одговара величини улазног плана, када се долази до оптималног редоследа. Слика 4.4 приказује оптимизован логички план са слике 4.3 уз процењене кардиналности.



Слика 4.4 – Оптимизован редослед операција спајања

4.7. Оптимизатор заснован на трошку

Задатак оптимизатора заснованог на трошку је да за сваки релациони оператор, изабере најповољнији алгоритам његове имплементације. Поред тога, за сваку изворну релацију специфицирану у логичком плану, потребно је да изабере најповољнији приступни пут. На основу процене трошка, постојања индексних структура и структуре логичког плана, оптимизатор, најпре бира, најповољнији приступни пут, који може бити секвенцијално или индексно скенирање, а затим бира најповољнији алгоритам имплементације операција спајања, који може бити угњеждено спајање, индексно спајање или хеш спајање. Овим кораком се комплетира физички план и враћа кориснику. Пример физичког плана приказан је на слици 4.5.



Слика 4.5 – Пример физичког плана

5. ЗАКЉУЧАК

MiniOptimizer алат је способан да генерише ефикасне планове извршења упита користећи основне поступке оптимизације упита попут скупа алгебарских закона за побољшање логичких планова, алгоритма динамичког програмирања за одређивање оптималног редоследа операција спајања и модела трошка за избор најповољнијих физичких оператора. Основна намена је да представи практичну имплементацију теоријских концепата које стоје иза поступка оптимизације упита и самим тим представља едукативни алат који служи као основа за даља истраживања и развој напреднијих система за управљање базама података.

Иако пружа солидну основу за оптимизацију упита, постоји низ праваца за даљи развој и унапређење алата. У будућим истраживањима, планира се додавање подршке за угњеждене упите, скуповне операције и ДМЛ (енгл. *Data Modification Language*) наредбе, што ће значајно проширити функционалности алата. Такође, интеграција извршиоца упита би омогућила валидацију перформанси алата и пружила детаљнију анализу ефикасности генерисаних планова извршења

6. ЛИТЕРАТУРА

[1] Hector Garcia-Molina, Jeffrey D. Ullman. Database Systems: The Complete Book. Prentice Hall Press, USA, 2 edition, 2008.

[2] Graefe, Goetz. “The Volcano optimizer generator: extensibility and efficient search.” Proceedings of IEEE 9th International Conference on Data Engineering (1993): 209-218.

[3] Selinger, Patricia G. et al. “Access path selection in a relational database management system.” ACM SIGMOD Conference (1979).

[4] Zongheng Yang et al. Balsa: Learning a query optimizer without expert demonstrations. In Proceedings of the 2022 International Conference on Management of Data, SIGMOD/PODS ’22. ACM, June 2022.

Кратка биографија:

Милош Гравара рођен је 20. јануара 2000. године у Новом Саду. Завршио је Гимназију „Исидора Секулић“ 2019. године након чега уписује Факултет техничких наука, смер Рачунарство и аутоматика. Дипломски рад је одбранио 2023. године када уписује мастер студије на студијском програму Рачунарство и аутоматика.