УНИВЕРЗИТЕТ У БЕОГРАДУ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

НЕМАЊА М. КОЈИЋ

ОПТИМИЗАЦИЈА ПРИСТУПА ПОДАЦИМА У ОБЈЕКТНО-РЕЛАЦИОНОМ МАПИРАЊУ ЗАСНОВАНА НА АУТОМАТСКОЈ ДЕНОРМАЛИЗАЦИЈИ

ДОКТОРСКА ДИСЕРТАЦИЈА

БЕОГРАД, 2019.

UNIVERSITY OF BELGRADE
SCHOOL OF ELECTRICAL ENGINEERING

NEMANJA M. KOJIĆ

OPTIMIZATION OF DATA ACCESS IN OBJECT-RELATIONAL MAPPING BASED ON AUTOMATIC DENORMALIZATION

DOCTORAL DISSERTATION

BELGRADE, 2019.

Ментор:

др Драган Милићев, редовни професор

Универзитет у Београду – Електротехнички факултет

Чланови комисије:

др Драган Милићев, редовни професор

Универзитет у Београду – Електротехнички факултет

др Драган Бојић, ванредни професор

Универзитет у Београду – Електротехнички факултет

др Саша Малков, ванредни професор

Универзитет у Београду – Математички факултет

др Јелица Протић, редовни професор

Универзитет у Београду – Електротехнички факултет

др Милош Цветановић, ванредни професор

Универзитет у Београду – Електротехнички факултет

Датум одбране:

_____ године

Захвалнице

Ова докторска дисертација представља резултат мог шестогодишњег рада на докторским академским студијама на Електротехничком факултету Универзитета у Београду који је започет почетком 2014. године и завршен крајем 2019. године. У њој су постављене почетне теоријске основе проблема побољшања транспарентности и ефикасности технологије објектно-релационог мапирања.

Током свог научно-истраживачког рада имао сам част и задовољство да сарађујем са многим људима који су имали значајан утицај на мој научни развој.

Прво бих се захвалио свом ментору, проф. др Драгану Милићеву, који ми је предложио ову тему научног истраживања, активно учествовао у постављању хипотеза и основа истраживања, пажљиво и детаљно анализирао резултате мог рада и својим ауторитетом и стручним коментарима значајно утицао на квалитет саме докторске дисертације. Посебно ми је задовољство и част што сам имао прилику да упијам од њега велико стручно знање, пословне вештине и професионалну етику.

Захваљујем проф. др Јелици Протић на драгоценим саветима и неизмерној подршци и помоћи коју ми је пружила током мог научно-истраживачког рада. Захвалност дугујем и проф. др Вељку Милутиновићу који ме је увео у воде научног истраживања и са којим сам објавио неколико научних радова. Захваљујем проф. др Драгану Бојићу са којим сам радио на научном истраживању из области аутоматског тестирања софтвера. Хвала и колегама проф. др Милошу Цветановићу, проф. др Захарију Радивојевићу и др Саши Стојановићу на инспиришућим дискусијама и вредним саветима и коментарима. Захвалио бих се свим професорима код којих сам полагао испите на докторским студијама, као и члановима Комисије за преглед, оцену и одбрану ове докторске дисертације на стручним и детаљним коментарима који су значајно допринели побољшању њеног квалитета. Хвала и свим наставницима и сарадницима Катедре за рачунарску технику и информатику са којима сам имао част и задовољство да сарађујем у настави током осам година рада на Електротехничком факултету.

Захваљујем администраторима Катедре за рачунарску технику и информатику, Драгану Миладиновићу и Драгиши Миладиновићу, на техничкој подршци у току припреме окружења за експерименталну анализу. Хвала и др Ненаду Королији на помоћи у току припреме документације за предају завршне верзије ове докторске дисертације.

Посебну захвалност дугујем мом пријатељу, колеги и „кумашину“ др Дражену Драшковићу са којим сам успешно сарађивао, објављивао научне радове, одлазио на антологијска факултетска путовања и проживљавао и добре и лоше тренутке. Захваљујем му на безрезервној подршци и пожртвовању, као и на кључној улози коју је одиграо у току завршних процедуралних активности припреме ове дисертације за оцену и преглед.

Неизмерну захвалност дугујем својој породици, родитељима Милки и Милану и брату Срђану, који су ми пружали безусловну љубав и подршку кроз живот и школовање. Захваљујем се и баки Радмили и деди Ђорђу уз које сам одрастао и чије искуство ми је увек било драгоцено у животу. Хвала породици Николић – Миланки и Милану, породици Игњатовић – Љиљани и Драгану, као и породици Барац, на подршци сваке врсте коју су пружили мени и мојој породици током мог рада на овој тези.

Посебну и највећу захвалност дугујем супрузи Ивани и деци, Елени и Александру, који сваки мој дан испуњавају срећом и љубављу, од којих свакодневно учим и са којима сазревам у сваком смислу. Хвала им на великом разумевању, стрпљењу, одрицању и несебичној подршци коју су ми пружили да овај посао успешно приведем крају, па зато ову дисертацију посвећујем њима.

У Београду, Немања Којић

3. 10. 2019. године

Резиме

Наслов: Оптимизација приступа подацима у објектно-релационом мапирању заснована на аутоматској денормализацији

Предмет научног истраживања описаног у овој докторској дисертацији јесте оптимизација објектно-релационог мапирања у објектно оријентисаном (ОО) апликативном софтверу за управљање подацима у великим релационим базама података заснована на денормализацији релационог модела. Чест проблем који се јавља у ОО апликативном софтверу у овом контексту јесте тај да се његове перформансе приступа подацима значајно погоршавају са повећањем количине података у бази података, као и порастом броја конкурентних приступа тим подацима.

Објектни модел програмирања, који почива на строгој објектној (семантичкој) декомпозицији проблема, елиминисању сваког вида редундансе, подели одговорности и колаборацији више типова објеката у обављању сложених апликативних операција, не даје увек добре перформансе приступа подацима у великим релационим базама података. Несофистицираност постојећих решења за објектно-релационо мапирање, која пресликавају нередундантни концептуални модел и навигациони начин приступа подацима у ОО моделу директно на нормализовани релациони модел података, доводи до појаве неефикасних образаца приступа бази података, као што су учестало обраћање бази података за дохватање релативно малих пакета података или извршавање сложених упита са доста операција спајања табела. Такви обрасци приступа великим релационим базама података најчешће су главни узрок лоших перформанси ОО апликативног софтвера за управљање подацима.

У пракси, као и у доступној литератури, постоје различите технике оптимизације приступа подацима за које је заједничко то да одлуку о оптимизацији доносе на основу анализе радног оптерећења апликативног софтвера (енгл. *workload*) сачињеног од операција приступа подацима издатих релационој бази у току његовог извршавања. Због сложености комбинаторног оптимизационог проблема одлучивања о избору „идеалне“ оптимизације приступа подацима за потребе задатог радног оптерећења, постојеће технике су често ограничене или на

решавање конкретних проблема у специфичним доменима, или на коришћење грубих експертских хеуристика за убрзавање претраживања огромног простора могућих оптимизација. Поред тога, предложена решења долазе углавном из домена аналитичких система (енгл. *online analytical processing – OLAP*). Међутим, не постоји решење које на систематичан и ефикасан начин третира проблем оптимизације приступа подацима у релационим базама података коришћењем објектно-релационог мапирања у домену трансакционих система (енгл. *online transaction processing – OLTP*).

У овој дисертацији предложена је метода аутоматске денормализације заснована на увођењу редундансе у релациони модел за оптимизацију приступа великим релационим базама података коришћењем објектно-релационог мапирања. Редундантни подаци, названи још и оптимизацијама, уводе се у релациони модел у циљу убрзавања (чешћих) операција читања уз свесно и циљано жртвовање ефикасности (ређих) операција уписа које треба да обезбеде конзистентност редундантних података. За потребе аналитичког моделовања постизања компромиса између позитивног утицаја таквих оптимизација на операције читања и њиховог негативног утицаја на операције уписа, предложен је формалан модел цене и добити (енгл. *cost-benefit*). У циљу проналажења праве мере увођења таквих оптимизација, којом се постиже најбољи однос између добити при читању и цене при упису, узимајући у обзир анализиране начине приступа подацима у радном оптерећењу, формулисан је оптимизациони проблем за постизање равнотеже редундансе у релационом моделу (енгл. *data redundancy equilibrium – DRE*) на један потпуно аутоматски и генерички начин. У тези је доказано да предложени оптимизациони проблем припада класи NP-комплетних проблема и дат је детаљан опис његовог свођења на NP-комплетан проблем бинарног линеарног програмирања (енгл. *binary linear programming – BLP*) за чије решавање данас постоје ефикасне хеуристике.

Предложени теоретски модел аутоматске денормализације релационог модела података експериментално је анализиран коришћењем *de facto* стандардног теста перформанси за трансакционе системе и базе података, TPC-E, (енгл. *The Transaction Processing Performance Council – TPC*), који приближно одсликава сложеност (нормализованог) релационог модела, количину података, као и

карактеристике радног оптерећења реалних трансакционих система. Применом предложене методе денормализације на референтну TPC-E релациону базу података (са нормализованом шемом), време извршавања прописаног TPC-E радног оптерећења побољшано је за 65% у односу на време потребно за извршавање истог тог оптерећења над референтном нормализованом базом података, уз потребу за 25% додатног простора за редундантне податке. Важност коришћења предложене аутоматске денормализације анализирана је и њеним поређењем са методом неселективне примене денормализације којом се занемарује негативан утицај оптимизација на операције уписа. Неселективном („похлепном“, енгл. *greedy*) денормализацијом референтне TPC-E релационе шеме добијено је побољшање времена извршавања коришћеног TPC-E радног оптерећења, али је оно за 19% било лошије у односу на време добијено предложеном аутоматском денормализацијом, при чему је величина базе података порасла три пута. Експерименталном анализом показано је још и то да предложени модел цене и добити за процену утицаја оптимизација даје употребљиве предикције односа цене и добити примењених оптимизација које су биле у линеарној корелацији са измереним показатељима перформанси.

На основу свих добијених резултата у току овог истраживања може се закључити да предложена метода аутоматске денормализације релационог модела може значајно да побољша ефикасност приступа подацима у великим релационим базама података, да даје добру полазну основу за унапређење транспарентности технологије објектно-релационог мапирања, као и то да остварена скалабилност предложене методе за аутоматску денормализацију отвара нове могућности и ствара реалан потенцијал за њену примену у пракси.

Кључне речи: релациона база података, денормализација, редунданса података, објектно-релационо мапирање, комбинаторна оптимизација, бинарно линеарно програмирање, NP-комплетан оптимизациони проблем

Научна област: Електротехника и рачунарство

Ужа научна област: Софтверско инжењерство

УДК број: 621.3:004

ABSTRACT

Title: Optimization of Data Access in Object-Relational Mapping based on Automatic Denormalization

This doctoral dissertation explores the problem of optimization of data access in object-relational mapping in large relational databases based on denormalization of relational data model. A common issue in everyday exploitation of object-oriented (OO) applications is that their performance decays significantly with the increase of data in the databases as well as of the number of concurrent data access operations.

The object programming model, which rests on the principles of strong object decomposition, elimination of redundancy of any kind, separation of concerns and collaboration of multiple types of objects in executing complex application operations and rules, does not always lead to acceptable performance of data access in large relational databases. The problem lies in lack of sophistication of existing object-relational mapping solutions which translate the normalization and the navigational way of data access at the object level directly to a normalized relational data model. Therefore, the described way of mapping leads to a proliferation of small and inefficient queries issued to the database and complex queries that require multiple joins. Such types of data access patterns are deemed the root cause of poor performance of object-oriented applications in most cases.

The application of data access optimization techniques, which can be found both in practice and in the available literature, is driven by analysis of data access operations issued to a relational database that comprise the so-called application software workload. Due to the complexity of the combinatorial optimization problem of finding the "ideal" relational model optimization based on the workload parameters, the existing solutions very often resort to solving specific or partial problems or utilize coarse approximations for optimizing search of the exponential space of potential solutions. In addition, the proposed solutions are mostly aimed at online analytical transaction processing systems (OLAP). However, there does not exist a solution that solves the problem of the optimization of data access in relational databases in online transaction processing system (OLTP) in a generic and systematic way.

Therefore, a novel method for optimization of data access in object-relational mapping, based on automatic denormalization of the relational data model in large databases, is proposed in this thesis. It is based on adding redundant data, also called optimizations, to the relational data model, that speed up (more frequent) read operations at the expense of the lower efficiency of (preferably rare) write operations that maintain consistency of the redundant data. For the sake of modeling the trade-off between the positive impact of the optimizations on read operations and the negative impact on write operations, a cost-benefit analytical model has been defined. Also, a formal optimization problem, called *Data Redundancy Equilibrium*, has been proposed for finding a deployment of redundancy in the relational data model that yields the best possible ratio between the positive and the negative impact of the optimizations given the workload characteristics. It has been proven that the proposed optimization problem belongs to the class of NP-Complete problems. A detailed polynomial reduction of the proposed optimization problem to the NP-complete problem of binary linear programming, which can be solved efficiently using existing heuristic algorithms, is described in the thesis.

The proposed theoretical model of the automatic denormalization has been experimentally evaluated by using the *de facto* standard TPC-E benchmark for transactional systems and databases, defined by The Transaction Processing Performance Council (TPC) organization, which approximately reflects the complexity of the (normalized) relational model, the amount of data and the workload characteristics of real transactional systems. By applying the proposed optimization method to the normalized TPC-E relational database, the execution time of the standard TPC-E workload was improved by 65%, compared to its execution time on the normalized TPC-E database, at the expense of the increase in the space for the redundant data of 25%. The importance of the proposed automatic denormalization was evaluated by comparing it with the unselective (greedy) denormalization which disregards the negative impact of the optimization on write operations. The unselective denormalization also improved performance of the TPC-E workload significantly, compared to its execution time on the normalized database, but the proposed automatic denormalization still gave better results than the unselective one by 19%. However, the size of the greedily denormalized TPC-E database was increased three times compared to the normalized database. It has been also shown that the

obtained predictions of the proposed cost-benefit model were in a linear correlation with the measured performance parameters.

Based on all the given results produced in this research, it can be concluded that the proposed method of automatic denormalization of the relational data model can significantly improve efficiency of data access in large relational databases and it lays firm foundations for further advances toward better transparency of the object-relational mapping technology. The achieved scalability of the proposed method of automatic denormalization opens new possibilities for its application in practice.

Keywords: relational database, denormalization, data redundancy, object-relational mapping, combinatorial optimization, binary linear programming, NP-Complete problem

Scientific field: Electrical Engineering and Computing

Scientific subfield: Software Engineering

UDC Number: 621.3:004

Кратак садржај

САДРЖАЈ

1. Увод

Капитални ресурс пословних система (енгл. *enterprise systems*) јесу подаци. Поуздано складиштење и руковање подацима захтев је највишег приоритета у пословним система који се неретко регулише правним и финансијским инструментима (посебно када су у питању системи из финансијског или неког другог правно осетљивог домена). И док се подаци сматрају најважнијим ресурсом у пословним системима, софтвер се сматра најважнијим „алатом“ који је од суштинског значаја за управљање тим подацима и њихово стављање у службу пословних процеса. Смисао постојања софтвера и података јесте искључиво у њиховој симбиози. Продором дигиталне трансформације у нове пословне домене драстично и прогресивно се повећава разноврсност и количина података којима софтверски системи данас управљају (нпр. *Internet of Things*, *IoT*) [1], [2]. У потрази за одговорима на новонастале изазове у индустрији софтвера, традиционалне и савремене методе и алати за развој софтвера за рад са подацима непрестано се унапређују и еволуирају [1].

Софтверски системи за рад са подацима сачињени су од различитих типова апликативног софтвера за руковање подацима (у даљем тексту „апликације“, енгл. *data-centric applications*). Радно оптерећење (енгл. *workload*) апликација за рад са подацима, у општем случају, подразумева комбиновано извршавање и операција читања/претраживања података, као и операција уписа/ажурирања података у вишекорисничком режиму рада. У зависности од намене и природе операција за приступ подацима, софтверски системи за рад са подацима деле се на највишем нивоу на софтверске системе за обраду трансакција (енгл. *online transaction processing systems*, *OLTP*) и софтверске системе за аналитичку обраду података (енгл. *online analytical processing*, *OLAP*). Аналитички системи типично се користе за извршавање комплексних упита над великим количинама података за претраге

и извештаје, док су претраге података у трансакционим системима типично знатно једноставније од аналитичких, њихова фреквенција може бити знатно већа и типично се извршавају упоредо са операцијама ажурирања података.

1.1. Објектно оријентисана парадигма развоја апликација

У основи развоја пословне логике апликација за рад са подацима данас преовладава објектно оријентисана (ОО) парадигма. Саздана на принципу апстракције (енгл. *abstraction*), елиминисања сваког вида редундансе и „пројектовања за промене“ (енгл. *“design for change”* [3]), имплементираних кроз концепте енкапсулације (енгл. *encapsulation*), наслеђивања (енгл. *inheritance*) и полиморфизма (енгл. *polymorphism*), допринела је не само ефектнијем и ефикаснијем пројектовању и развоју софтвера, већ и његовом одржавању и еволуцији као одговору на промене функционалних захтева [4].

Развој објектно оријентисаних апликација започиње концептуалним моделовањем (енгл. *conceptual modeling*) – идентификацијом кључних концепата / ентитета, кључних односа између ентитета (енгл. *relationships*), као и идентификацијом кључних активности из домена проблема [4]. Ентитети из домена проблема моделују се доменским објектима чија се заједничка својства апстрахују класама у апликативном програмском ОО моделу. Својства ентитета моделују се атрибутима доменских класа, док се функционалности моделују операцијама придруженим класама. Објекти су типично повезани структурним везама (енгл. *links*), формирајући структуре усмерених графова у којима се могу појавити и циклуси.

Структурним везама моделује се логичка и природна повезаност објеката у домену проблема која се манифестује кроз природну потребу за њиховом колаборацијом у оквиру различитих доменских активности. Другим речима, активности у домену проблема производ су деловања тачно одређених скупова (логички повезаних) ентитета. На пример, иако студент може бити повезан, кроз разне видове сарадње, са више професора, докторска дисертација може настати само у колаборацији са изабраним ментором. Однос менторства у овом случају одсликава логичку повезаност ментора и студента током научно-истраживачког рада и настанка докторске дисертације као финалног производа те колаборације.

Додатно ограничење у домену докторских студија може бити и то да професор не може бити ментор студенту, уколико студент није одслушао бар један од курсева које професор држи. Да би се проверило то доменско правило, потребно је имати на располагању (у току извршавања програма) објекте професора, студента, курсеве које је студент одслушао, као и курсеве које професор држи. Уколико постоји преклапање дата два скупа курсева, испуњен је услов за менторство. Такви доменски односи између професора, студената и курсева моделују се структурним везама у објектном моделу. Да би се разматрали одговарајући курсеви, објекат професора и студента морају да буду унапред задати, а до курсева које држе, односно слушају долази се следом структурних веза који се назива *навигација*.

1.2. Појам објектног простора

Функционалности ОО апликација за рад са подацима имплементирају се руковањем доменским објектима који настањују логички неограничену меморију под називом *објектни простор* (енгл. *object space*) [5]. Објекти се могу стварати у објектном простору, уклањати из објектног простора, може им се ажурирати стање, могу се повезивати са другим објектима или се те везе могу раскидати и могу се претраживати у објектном простору. Апликативни програмски модели ОО апликација по својој природи су навигациони [6]. Објектни модел програмирања почива на строгој објектној (семантичкој) декомпозицији проблема, елиминисању сваког вида редундансе, подели одговорности и колаборацији више типова објеката у обављању сложених апликативних операција. На исти начин, декларативни упити на објектном упитном језику (енгл. *object query language – OQL*) такође користе навигацију преко структурних веза за приступ подацима (вредностима атрибута објеката) који се користе или у испитивању критеријума претраге или се враћају као резултати у пројекцији. У оба наведена случаја, било да се операције имплементирају као императивне секвенце акција над објектним простором или као декларативни упити, навигација (преко структурних веза) преставља главни образац приступа подацима у објектном простору.

Да би се омогућило трајно и поуздано чување података, објектни простор апликација за рад са подацима мора да буде перзистентан (енгл. *persistent*). За перзистенцију објектног простора традиционално се користе релационе базе података које су специјализоване за трансакциони приступ подацима [7] у

вишекорисничком режиму и поуздано управљање подацима у секундарној меморији за трајно чување података реализованој помоћу технологије магнетних дискова (енгл. *hard disk drives*) и технологије полупроводничке трајне („флеш“) меморије (енгл. *solid state drives*). За оба типа уређаја за трајно чување података у наставку текста биће коришћен заједнички термин дискови, ради концизности. Иако би објектно оријентисане базе података биле природно окружење за трајно складиштење објектних модела, оне до данас нису успеле да потисну релационе базе података [8], како због сложености технологије ОО база података, тако и због теоријских и практичних предности технологије релационих база података [9].

1.3. Релационе базе података

Релационе базе података одржале су се до данас као доминантна технологија у индустрији софтверских система за руковање подацима, који се одликују великим и сложеним концептуалним моделима, захваљујући чврстим математичким основама на којима је постављена теорија релационог модела, постојању општеприхваћених стандарда и бројном заједницом, као и зрелости и поузданости технологије која се развијала деценијама [8].

1.3.1. Релациони модел података

Релациони модел података је од посебног значаја за даљу дискусију у дисертацији, тако да се у овој секцији даје кратак преглед његових најважнији структурних концепата и дефиниција.

Домен (енгл. *domain*) представља скуп вредности истог типа. Домен је прост уколико су вредности у њему простог типа података (у том смислу да систем за управљање базама података не може да их разложи на мање делове). Нека се посматра n домена, $D_1, D_2, \dots, D_n$, $n > 0$, који не морају нужно бити различити. *Шема релације* R дефинише се као скуп атрибута $\{A_1, A_2, \dots, A_n\}$, где сваки атрибут A_i има придружен домен $D_i = dom(A_i)$ [10], [11]. Релација r задате шеме R, у ознаци $r(R)$, дефинише се као коначан скупа пресликавања $r = \{t_1, t_2, \dots, t_m\}$, где свако t_i пресликава шему релације R на унију домена њених атрибута, у ознаци $t_i: R \rightarrow dom(A_1) \cup dom(A_2) \cup \dots \cup dom(A_m)$, при чему мора да важи услов $t_i[A_j] \in dom(A_j)$, $1 \leq j \leq n$. Свако такво пресликавање t у релацији r назива се торка, док се нотацијом $t_i[A_j]$ представља вредност атрибута A_j шеме релације R у

посматраној торки. У изворном математичком смислу, релација се дефинише формално као подскуп Декартовог производа домена $D_1 \times D_2 \times \ldots \times D_n$ који представља скуп уређених n-торки $(d_1, d_2, \ldots, d_n)$, тако да важи $d_i \in D_i,\ 1 \leq i \leq n$. Каже се да скуп n-торки у релацији представља *тренутно стање релације* или *тренутни модел дела реалног света* који се моделује посматраном релацијом. [11]

Елегантност релационог модела проистиче из могућности да се истим концептом, релацијом, могу униформно моделовати, како чињенице о ентитетима из реалног света, тако и чињенице о њиховим међусобним односима. Свака шема релације може да се посматра као апстрактна спецификација одређене тврдње (енгл. *assertion*) о ентитету или односу из реалног света. Свака торка у једној таквој релацији може да се посматра као инстанца дате тврдње или чињеница из реалног света (прецизније модел чињенице из реалног света). [12], [13]

На сличан начин, релациони модел може да се доведе у блиску везу са предикатском логиком првог реда [14]. Наиме, шема неке релације може да се посматра као спецификација једног n-арног *предиката* (енгл. *predicate*). За сваку торку која припада датој релацији каже се да њене вредности задовољавају дати предикат, као и да било која друга комбинација могућих вредности у торкама, која не постоји у стању релације, не задовољава тај предикат [11].

У математичком смислу, стање релације је подразумевано непроменљиво (константно). Међутим, у релационом моделу, свака релација се сматра (временски) променљивом (енгл. *time-varying*), уколико се не прецизира другачије [13]. Када се говори о променљивости релације, мисли се пре свега на њено стање, али се промене понекад дешавају и на нивоу саме шеме релације (додавањем или уклањањем атрибута).

У релационом моделу типови података се класификују на просте и сложене. Вредности простих типова података не могу се разложити на мање саставне делове (нпр. од стране система за управљање базом података), док се вредности сложених типова података могу разложити на мање саставне делове. У изворном релационом моделу података постоји један једини сложени тип података, а то је релација. Релациона база података представља скуп релација различитих степена. Сви доступни оператори примењују се искључиво на релације и дају релације као резултат (затворени су на скупу релација). Едгар Код тврди [13] да додавање било

ког другог сложеног типа податка само повећава сложеност релационог модела без добијања на снази и изражајности модела. За сваки тип податка морају да постоје основне операције за њихово руковање (стварање, ажурирање, брисање, читање). За сваки додатни сложени тип података у релационом моделу било би потребно дефинисати и имплементирати посебне верзије наведене четири основне операције [14].

Структурно повезивање података (торки) у релационом моделу заснива се искључиво на поређењу вредности, било да су те вредности идентификатори ентитета или нека друга њихова својства. Улога и постојање домена од суштинског је значаја за успостављање веза између ентитета јер две вредности има смисла упоредити ако и само ако припадају истом домену (не само типу података) [13]. На пример, идентификатор особе и године старости могу бити истог (целобројног) типа, али њихово поређење нема логичког смисла.

Уколико су сви домени релације прости, релација може да има табеларну меморијску репрезентацију са својствима да је свака торка у њој јединствена, да поредак торки није битан, да поредак вредности атрибута у торкама није битан, и да су све вредности атрибута у торкама атомичне [14]. У релационом базама података релације су имплементиране као табеле, док су торке представљене њиховим редовима. Из овог разлога, када се у даљем тексту буду наглашавали аспекти физичке организације релационог модела, термини релација и торка биће замењени терминима табела и ред, респективно.

1.3.2. Нормализација релационог модела

У пракси шема релационог модела се типично пројектује и одржава поштовањем принципа нормализације [10], [11]. Нормализована релациона шема је семантички стабилна, лако разумљива и минимизује или потпуно елиминише аномалије у релационом моделу, као што су понављање истих података, немогућност представљања одређених информација и односа, или чак и губитак информација [15], [11], [16]. Откако је Едгар Код иницирао истраживање нормализације релационог модела [10], она је била предмет проучавања и многих других истраживача [16]. Нормализација релационог модела формализована је у виду критеријума под називом *нормалних форми* које дефинишу правила груписања и распоређивања атрибута у шемама релација у циљу добијања модела

без поменутих аномалија, а уз истовремено очување свих *функционалних зависности* података из домена проблема.

Иницијално је Едгар Код предложио прве три нормалне форме и назвао их редом прва (*1НФ*), друга (*2НФ*) и трећа (*3НФ*) нормална форма. Касније је добијена стриктнија варијанта треће нормалне форме под називом Бојс-Кодова нормална форма (енгл. *Boyce-Codd Normal Form, BCNF*). Дате нормалне форме засноване су на концепту *функционалне зависности* података. Касније је предложена четврта нормална форма (*4НФ*), заснована на концепту *вишевредносних зависности* (енгл. *multivalued dependency*) података, као и пета нормална форма (*5НФ*) заснована на концепту зависности спајања (енгл. *join dependency*) [11]. Поред наведених традиционалних нормалних форми, у доступној литератури могу се пронаћи и други видови нормалних форми (нпр. објектна нормална форма, енгл. *object normal form* [17]).

1.3.3. Кратак преглед дефиниција нормалних форми

У наставку ове секције наводи се кратак преглед основних концепата нормализације и самих нормалних форми.

За шему релације R, представљену скупом атрибута $\{A_1, A_2, \dots, A_n\}$, *наткључ* (енгл. *superkey*) представља подскуп атрибута $S \subseteq R$ такав да не постоје две торке t_1 и t_2 у валидном стању дате релације за које важи $t_1[S] = t_2[S]$. *Кључ* K представља наткључ за додатном особином да уклањање било ког атрибута из K доводи до тога да K губи особину наткључа. Другим речима, кључ представља минималан наткључ. У општем случају, једна релација може да има више *кандидата за кључ*, од којих се један усваја као *примарни кључ*. Атрибут релације R је примаран уколико припада било ком од њених кључева кандидата.

Прва нормална форма (1НФ) дефинише ограничење по ком сваки атрибут шеме релације може да садржи највише једну вредност неког од простих типова података (домена).

Друга нормална форма (2НФ) заснована је на концепту *потпуне функционалне зависности* (енгл. *full functional dependency*) атрибута шеме релације. За скупове атрибута X и Y важи да је функционална зависност $X \rightarrow Y$ потпуна уколико за било који атрибут A из скупа X важи да скуп атрибута $\{X - \{A\}\}$ не одређује

функционално скуп атрибута Y. У супротном функционална зависност је *парцијална*. Уколико R представља скуп свих атрибута шеме једне релације, при чему је скуп атрибута X ($X \subseteq R$) њен примарни кључ, дата релација задовољава другу нормалну форму ако је сваки њен атрибут из скупа $R - X$ потпуно функционално зависан од њеног примарног кључа.

Трећа нормална форма (3НФ) заснована је на концепту транзитивне функционалне зависности (енгл. *transitive functional dependency*). Функционална зависност $X \rightarrow Y$ у шеми релације R је транзитивна, уколико постоји неки подскуп атрибута Z, $Z \subset R$, који није кандидат за кључ нити подскуп било ког другог кључа шеме релације R, такав да важе функционалне зависности $X \rightarrow Z$ и $Z \rightarrow Y$. Шема релације R задовољава *3НФ* уколико задовољава *2НФ* и не постоји атрибут у шеми R који није примаран и истовремено транзитивно функционално зависан од њеног примарног кључа [10]. Трећа нормална форма може да се формулише и алтернативно на следећи начин: шема релације R задовољава *3НФ*, уколико за сваку нетривијалну функционалну зависност $X \rightarrow A$ у њој важи следеће: а) X је наткључ у шеми R или б) A је примарни атрибут у шеми релације R. Другим речима, шема релације R задовољава *3НФ* уколико је сваки њен непримарни атрибут потпуно функционално и нетранзитивно зависан од сваког кључа у R [11].

Бојс-Кодова нормална форма (BCNF) представља стриктнију варијанту *3НФ*. Шема релације R задовољава нормалну форму *BCNF* уколико у свакој нетривијалној функционалној зависности $X \rightarrow A$, која у њој важи, X представља наткључ.

Четврта нормална форма (4NF) заснована је на концепту *вишевредносне зависности* (енгл. *multivalued dependency*). Нарушавање четврте нормалне форме често се дешава као последица трансформације шеме релације са вишевредносним атрибутима у *1НФ* тако да се за сваку вредност таквог атрибута додаје нова торка у релацију (уз понављање вредности осталих атрибута који нису вишевредносни). Нека се посматра шема релације R са два дисјунктна подскупа атрибута X и Y и подскупом атрибута $Z = R - (X \cup Y)$. Релација $r(R)$ задовољава мултивредносну зависност $X \twoheadrightarrow Y$, ако за било које две торке t_1 и t_2 у њој са идентичним вредностима скупа атрибута X, $t_1[X] = t_2[X]$, постоји торка t_3 таква да су

задовољени следећи услови: $t_1[X] = t_3[X]$, $t_1[Y] = t_3[Y]$ и $t_2[Z] = t_3[Z]$. [17] Пошто постоји симетрија у третирању скупова атрибута Y и Z према претходној дефиницији, посматрана вишевредносна зависност може се записати нотацијом $X \twoheadrightarrow Y|Z$. Вишевредносна зависност $X \twoheadrightarrow Y$ је *тривијална* ако је а) $Y \subseteq X$, или б) $X \cup Y = R$. Вишевредносна зависност $X \twoheadrightarrow Y|Z$ назива се *правом вишевредносном зависношћу* ако није тривијална и ако ниједан од њених делова $X \twoheadrightarrow Y$ и $X \twoheadrightarrow Z$ не представља функционалну зависност [17]. Да би се избегла нежељена редунданса (торки) у релацији $r(R)$ узрокована постојањем вишевредносне зависности, посматрана шема релације се може трансформисати на два фрагмента без губитка информација, $R_1(X, Y)$ и $R_2(X, Z)$. На основу свега наведеног, дефинише се да једна шема релације R задовољава *4НФ* уколико истовремено задовољава *BCNF* и не садржи праве вишевредносне зависности.

Пета нормална форма (5НФ) заснована је на концепту *зависности спајања* (енгл. *join dependency*). У случају постојања вишевредносних зависности, трансформација шеме релације у *4НФ* може да се спроведе итеративном бинарном декомпозицијом без губитка информација или добијања сувишних торки приликом спајања. Међутим, у неким случајевима шему релације није могуће итеративно бинарно декомпоновати на описани начин (без нарушавања стања оригиналне релације), већ је то могуће учинити искључиво њеним декомпоновањем на више од две пројекције. Постојање зависности спајања, у ознаци $JD(R_1, R_2, \ldots, R_n)$, којом се дефинише сложено ограничење над стањем неке релације $r(R)$, подразумева да се њена шема може једино декомпоновати на пројекције $R_1, R_2, \ldots, R_n$, тако да се првобитно стање релације $r(R)$ може поново реконструисати њиховим природним спајањем, $*(\pi_{R1}(R), \pi_{R2}(R), \ldots, \pi_{Rn}(R)) = r(R)$. Зависност спајања је тривијална уколико је бар једна од њених пројекција једнака шеми релације R. Зависност спајања $JD(R_1, R_2, \ldots, R_n)$ је нередундантна уколико ниједан подскуп пројекција у њој не дефинише зависност спајања у стању релације $r(R)$. Коначно, шема R релације $r(R)$ задовољава *5НФ*, уколико за сваку нередундантну зависност спајања $JD(R_1, R_2, \ldots, R_p)$ у релацији $r(R)$ свака пројекција R_i, $1 \leq i \leq p$, представља наткључ у шеми R релације $r(R)$ [17].

1.3.4. Објектно-релационе базе података

Произвођачи релационих база података временом су препознали нарастајуће потребе у домену ОО концептуалног моделовања, као и недостатке релационог модела за ефикасну имплементацију комплексних ОО доменских модела, тако да су неки од концепата ОО база података уграђени у постојеће системе за управљање релационим базама података. Примери таквих објектних концепата који се могу срести у традиционалним комерцијалним релационим базама података јесу:

а) објектни идентификатори (енгл. *object identifier – OID*),

б) кориснички дефинисани типови података (енгл. *user-defined data types*),

в) енкапсулација операција у кориснички дефинисаним типовима података,

г) наслеђивање кориснички дефинисаних типова (укључујући наслеђивање и атрибута и операција), као и

д) полиморфизам операција корисничких типова [11].

Формално гледано, релационе базе података са придруженим ОО концептима у литератури се третирају као хибридна решења и називају објектно-релационим системима за управљање базама података (енгл. *object-relational database management systems*) [11]. Неки од поменутих објектних концепата могу се пронаћи у популарним системима за управљање базама података, као што су, примера ради, *IBM DB2*, *Postgre SQL*, *Oracle*, *SQL Server*.

1.4. Објектно-релационо мапирање

Доменска логика ОО апликативног софтвера за управљање подацима у релационим базама података типично се имплементира руковањем објектима, док се истовремено тежи апстраховању приступа бази података и „сакривању“ елемената и детаља релационог модела. За подршку таквом типичном приступу развоја софтвера неопходно је да постоји пресликавање елемената ОО модела на елементе релационог модела које се назива *објектно-релационо мапирање* или скраћено ОРМ (енгл. *object-relational mapping – ORM*).

Објектно-релационо мапирање у пракси се користи у два различита контекста, који се везују за две фазе животног циклуса ОО софтвера, а то су:

а) *Статичко пресликавање* између елемената ОО и релационог модела података које се спроводи у фази имплементације софтвера (енгл. *design-time*) и којим се омогућава трансформација једног модела података у други, као и

б) *Динамичко пресликавање* акција на објектном нивоу на операције (упите) на нивоу релационог модела за време извршавања програма (енгл. *run-time*).

У теорији и пракси постоје и користе се различите стратегије објектно-релационог мапирања и свака од њих има и предности и недостатке који су анализирани до сада у доступној литератури [18], [19], [20]. Квалитет стратегије мапирања типично се квантитативно процењује бројем потребних релација у релационом моделу, сложеношћу операција приступа подацима, као и отпорношћу модела на аномалије при упису [18].

Када је у питању статичко објектно-релационо мапирање, посебно је значајно нагласити то да се у овој дисертацији не предочава ниједан конкретан смер нити редослед корака трансформације између ОО и релационог модела. У развоју ОО апликативног софтвера за руковање подацима постоје три основна приступа објектно-релационог мапирања [21], која су класификована на основу типа централног концептуалног модела од ког се полази у развоју, а то су:

а) **Приступ *„model-first“*** – објектни модел, представљен у језику за моделовање са визуелном нотацијом (нпр. Unified Modeling Language – UML [22]), представља централни модел на основу којег се одговарајућом (аутоматском) трансформацијом може добити ОО изворни код дефиниција класа на изабраном програмском језику имплементације и шема релационог модела података;

б) **Приступ *„code-first“*** – пројектовање концептуалног модела започиње дефинисањем класа на изабраном ОО програмском језику имплементације, а затим се одговарајућом (аутоматском) трансформацијом може добити шема релационог модела података, као и (визуелни) објектни модел;

в) **Приступ *„database-first“*** – примењује се онда када је релациони модел података унапред дефинисан и представља централни модел из којег се

одговарајућом (аутоматском) трансформацијом могу добити дефиниције класа на ОО језику имплементације, као и објектни модел.

Без обзира на усвојени приступ развоја ОО апликативног софтвера за управљање подацима, полазни концептуални модел података по правилу се пројектује поштовањем описаног принципа „нормализације“, као важног принципа „добре праксе“.

1.4.1. Усвојена стратегија објектно-релационог мапирања

У пракси, статичко објектно-релационо мапирање у фази пројектовања и развоја софтвера, спроводи се, уз мање или веће варијације, коришћењем једне стратегије која је широко распрострањена у развоју ОО апликативног софтвера за управљање подацима у релационој бази података. Она је укратко описана у наставку стављањем акцента искључиво на оне детаље пресликавања који су од интереса за наставак дискусије у тексту ове дисертације.

Објектно-релационо мапирање између ОО концептуалног модела и шеме релационе базе података спроводи се помоћу следећих шест описаних трансформација. У свакој од трансформација прво се наводи објектни концепт, а затим и њему одговарајући релациони концепт, али без намере да се на тај начин фаворизује смер саме трансформације.

T1) За сваку класу (апстрактну или конкретну) у ОО моделу мора да постоји одговарајућа шема релације у релационом моделу.

T2) За сваки атрибут класе у ОО моделу мора да постоји одговарајући атрибут у шеми релације која одговара класи у којој је атрибут дефинисан.

T3) За сваки прости тип података у ОО моделу мора да постоји одговарајући прости домен (тип) података у релационом моделу (нпр. ОО тип података *String* пресликава се у тип података *Varchar(N)* у релационом моделу).

T4) За сваку асоцијацију типа *1:N* између две класе у ОО моделу мора да постоји одговарајући атрибут (страни кључ) у шеми релације која одговара класи на страни *N* посматране асоцијације.

T5) За сваку асоцијацију типа *M:N* у ОО моделу мора да постоји одговарајућа шема релације са два атрибута (страна кључа) релационом моделу, од

којих је један везан за шему релације за класу на страни M, а други је везан за шему релације за класу на страни N.

Т6) За сваки однос генерализације/специјализације у ОО моделу мора да постоји пар атрибута, од којих је један примарни кључ у шеми релације наткласе, а други је примарни кључ у шеми релације изведене класе. Преклопљено наслеђивање (енгл. *overlapped inheritance*) није подржано. За сваки пар релација $r(R_1)$ и $r(R_2)$ у релационом моделу, са идентичним примарним кључем K, чије шеме R_1 и R_2 одговарају класама изведеним из исте наткласе у ОО моделу, мора да буде задовољен услов да не постоји ниједан пар торки, $t_1 \in r(R_1)$ и $t_2 \in r(R_2)$, са особином да имају идентичну вредност примарног кључа, односно $t_1[K] = t_2[K]$.

Наведене трансформације обезбеђују релативно једноставно и праволинијско пресликавање између ОО и релационог модела података. Без обзира на то да ли релациони модел података представља централни модел који се пројектује поштовањем принципа нормализације, или се добија објектно-релационим мапирањем на основу „нормализованог“ објектног модела, по правилу је шема релационог модела података нормализована (типично у смислу задовољења свих нормалних форми од *1НФ* до *5НФ*).

Нормализована шема релационог модела минимизује сложеност операција уписа, па самим тим поједностављује и пресликавање операција уписа на објектном нивоу на операције/упите на релационом нивоу. На пример, додавање објекта у објектни простор једноставно се пресликава на операцију додавања новог реда у табелу класе тог објекта. Уколико објекат припада класи која је у хијерархији извођења, додавање новог објекта пресликава се, према усвојеној стратегији мапирања, на неколико релационих операција које додају нове торке у све релације класа на стази наслеђивања од релације (динамичке) класе објекта који се ствара до релације корене класе, што представља пример мало сложенијег пресликавања. Ажурирање вредности атрибута објекта своди се на ажурирање вредности атрибута у одговарајућој торки релације чија шема одговара класи датог објекта ОО моделу. Ако је атрибут наслеђен и концептуално дефинисан у некој од натклаcа, пресликавање операције ажурирања вредности наслеђеног атрибута захтева прво

одређивање шеме релације у којој постоји дати атрибут, а затим се ажурира вредност атрибута у одговарајућој торки те релације.

1.4.2. Реализација објектно-релационог мапирања у пракси

У пракси се објектно-релационо мапирање најчешће спроводи или а) писањем наменске имплементације за потребе конкретног концептуалног модела, или б) коришћењем неког од популарних генеричких радних оквира за ОРМ (*Entity Framework* [23], *Enterprise Java Beans* [24], *Hibernate* [25] и других).

Наменска имплементација подразумева мануелно пројектовање шеме релационог модела и дефиниција доменских класа, као и имплементацију мапирања писањем SQL упита. Овај приступ реализацији објектно-релационог мапирања даје потпуну контролу над мапирањем, уз додатну цену напора у развоју софтвера, његовом тестирању и отклањању софтверских грешака. Цена додатног напора уложеног у наменску имплементацију објектно-релационог мапирања посебно је изражена уколико су промене концептуалног модела честе, што је неретка пракса у окружењу са агилном методологијом развоја софтвера (енгл. *agile development methodology* [1]).

Са друге стране, готова решења за објектно-релационо мапирање драстично повећавају продуктивност развоја, посебно у раним фазама, омогућавајући аутоматско пресликавање између објектног и релационог модела података, што је посебно важно при изменама концептуалног модела. Готова решења дају мању контролу над самим мапирањем за разлику од наменског мапирања и врло често у пракси намећу ограничења мапирања која чине пресликавање сложених доменских операција и софистицираних концептуалних решења веома компликованим, па се неретко критичне и сложене доменске функционалности имплементирају наменски, заобилажењем радног оквира за ОРМ. С тим у вези, популарни радни оквири за ОРМ типично нуде и могућност додавања таквих специфичних наменских мапирања уколико постојећа генеричка мапирања не дају добре перформансе или су сувише комплексна.

1.5. О ЕФИКАСНОСТИ ПРИСТУПА ПОДАЦИМА У ОБЈЕКТНО-РЕЛАЦИОНОМ МАПИРАЊУ

У експлоатацији ОО апликација за управљање подацима у релационим базама података често се јавља проблем лоших перформанси њихових функционалности када количина података којима се приступа, као и број конкурентних приступа подацима у радном оптерећењу, постану сувише *велики*. У циљу појашњења дате тврдње, најпре се даје јаснија дефиниција квалификације *велико* (нпр. велике базе података, велики број трансакција, велике табеле).

1.5.1. ПОЈАМ ВЕЛИКИХ ПОДАТАКА

Појам великих података појављивао се периодично у историји рачунарства [26]. У наставку се дају две комплементарне дефиниције великих података.

„Скупови података, који по својој количини и сложености превазилазе могућности ефикасне обраде традиционалним методама, могу се сматрати великим подацима“ [26]. Технички гледано, подаци су велики уколико потреба за рачунарским ресурсима за њихову обраду на традиционалан начин превазилази доступне капацитете расположивих (ограничених) рачунарских ресурса (оперативне меморије, процесора, дискова итд.) [26].

Слична дефиниција великих података формулисана је на следећи начин: „Великим подацима могу се сматрати сви скупови података које није могуће сагледати (енгл. *perceive*), прибавити (енгл. *acquire*), организовати (енгл. *manage*) и обрадити (енгл. *process*) традиционалним софтверским и хардверским техникама и алатима у прихватљивом времену (енгл. *tolerable time*)“ [27].

Од сада па надаље у тексту дисертације, квалификација *велико* у контексту података сматра се дефинисаним у смислу наведених дефиниција [27], [26].

1.5.2. ТЕХНИКЕ ОПТИМИЗОВАЊА ПРИСТУПА ПОДАЦИМА У РЕЛАЦИОНИМ БАЗАМА ПОДАТАКА

Један од главних узрока лоших перформанси ОО апликација лежи у њиховом начину приступа подацима који је у својој суштини најчешће неефикасан за технологију релационих база података ускладиштених на дисковима. Негативни ефекти таквог неефикасног начина приступа подацима постају приметни и

прогресивно се погоршавају са порастом величине базе података, као и навале захтева у радном оптерећењу.

У пракси, као и у теорији, постоје различите технике побољшања ефикасности приступа релационим базама података, које се типично примењују у оптимизовању извршавања упита претраживања података у аналитичким системима на основу информација о начинима приступа подацима из радног оптерећења. Такве технике су типично засноване на увођењу секундарних приступних структура (индекса, материјализованих погледа), као и на елиминацији скупих операција спајања табела у упитима претраживања коришћењем техника денормализације релационог модела. Технике денормализације су посебно ефикасне и моћне, али је њихова примена доминантно сведена на домен аналитичких система [15].

Када је у питању оптимизација извршавања императивног објектног кода, постојећа решења се базирају или на поправљању ефикасности апликативног ORM кода, или на техникама дохватања података унапред коришћењем предикције приступа подацима на нивоу објектно-релационог мапирања. Међутим, једна од хипотеза у овој дисертацији јесте та да се поменутим техникама оптимизације приступа бази података само „лече“ последице, док узрок и даље остаје присутан на свом извориштy – на нивоу релационог модела података, као и да се на нивоу релационог модела података крије вредан и недовољно истражен потенцијал за оптимизацију приступа подацима у *OLTP* системима. Стога, предмет научног истраживања у овој дисертацији јесте теоријско разматрање могућности и потенцијала примене техника денормализације релационог модела за оптимизацију приступа подацима у објектно-релационом мапирању у реалним трансакционим софтверским системима.

1.6. Структура докторске дисертације

Након уводног поглавља које дефинише контекст и оквире дисертације, у глави 2 говори се више о дефиницији проблема и мотивацији датог научног истраживања, а потом следи детаљан преглед постојећих решења у глави 3. У глави 4 дат је неформални опис проблема оптимизације приступа подацима употребом техника денормализације, као и предлог решења, а затим у глави 5 следи детаљан опис формалног поступка примене предложене методе оптимизације приступа подацима засноване на денормализацији. Предложена метода оптимизације подвргнута је експерименталној анализи и најважнији резултати презентовани су и дискутовани у глави 6. Коначни закључци, као и разматрање даљих праваца развоја идеје и датог истраживања наведени су у глави 7.

2. Дефиниција проблема и циљеви истраживања

2.1. Проблеми неефикасног приступа релационој бази података

Функционалности пословне логике у ОО трансакционим апликацијама веома често су сложене и софистициране, тако да је за њихову реализацију, у складу са правилима ОО парадигме, неопходна колаборација већег броја логички повезаних (енгл. *related*) објеката (различитих типова). Логичка повезаност објеката имплементира се структурним везама (инстанцама асоцијација), па се логички релевантни објекти неопходни за имплементацију апликативних функционалност типично прибављају навигацијом преко структурних веза.

Имајући у виду усвојену стратегију директног (праволинијског) статичког пресликавања између ОО и релационог модела може се закључити да се „нормализација“ дизајна у једном моделу (нпр. објектном) директно преноси на други (нпр. релациони). Било да се шема релационог модела пројектује као полазни / централни концептуални модел података, или се добија статичким објектно-релационим мапирањем из нередундантног ОО концептуалног модела, она је типично нормализована.

Неефикасан приступ великим релационим базама података у ОО трансакционим апликацијама узрокован је несофистицираним и праволинијским начином објектно-релационог мапирања којим се објектни начин руковања и приступа подацима, пре свега навигација, преноси директно на релациони модел података.

Навигациони модел приступања подацима погодан је за меморије са добрим перформансама случајног приступа подацима (енгл. *random data access*). Објекти у

ОО језицима, и начин руковања њима, дизајнирани су баш за меморију са директним приступом [28].

Међутим, када се навигација на објектном нивоу пресели и примени директно као начин приступа подацима у релационим базама података ускладиштеним на диску, она доводи до приступања већем броју различитих табела. Учитавање сваког (перзистентног) објекта из базе података захтева типично приступ једном реду у одговарајућој табели. За сваки захтевани ред са диска се преноси цео блок података (осим ако није већ кеширан).

Учестало приступање размештеним (случајним) блоковима у случају магнетних дискова, има убедљиво најлошије перформансе. Са друге стране, технологија дискова са полупроводничком („флеш“) меморијом за трајно чување података даје знатно боље перформансе случајног приступа [29], [30], али време одзива случајног приступа блоку на диску није једини проблем ни узрок лоших перформанси приступа подацима.

Случајан приступ редовима у великим и различитим табелама доводи до повећаног преноса блокова са диска у меморију, што може утицати, у складу са расположивим системским ресурсима, на повећање режијских операција на нивоу самог оперативног система (честих „промашаја“ и замена страница у кеш меморији, честих замена страница виртуелне меморије, уског грла на улазу-излазу итд.). Чак и са добрим перформансама случајног приступа које технологија дискова са полупроводничком меморијом даје, није могуће ублажити све остале наведене негативне ефекте на перформансе система као целине.

2.2. Недостаци постојећих принципа оптимизације

У пракси, као и у доступној литератури, оптимизације објектно-релационог мапирања углавном се спроводе или на објектном нивоу [31] или на нивоу објектно-релационог мапирања [6], [32]. Међутим, применом ових техника не решава се суштинска сложеност приступа подацима у бази података са нормализованом релационом шемом.

Оптимизације приступа подацима засноване на проналажењу и поправљању неефикасних образаца програмирања апликативног кода (тзв. антиобразаца) ефикасне су уколико је неефикасан код једини узрок проблема лоших

перформанси [33], [34], [35], [31]. Међутим, у пракси се често дешава да чак и изворни код који је оптимално имплементиран има лоше перформансе због описаног неефикасног приступа на нивоу релационог модела података. Осим тога, циљ и јесте то да програмер свој концептуални модел и апликативни програм прави на начин који је њему најлакши и најпримеренији домену проблема и његовом добром логичком представљању, односно према принципима апстракције, енкапсулације, добре расподеле одговорности, локализације и избегавања редундансе сваке врсте, а без оптерећења питањима перформанси приступа подацима и начина њиховог смештања у бази података. [33], [34], [35].

Технике оптимизације приступа подацима које се предлажу на нивоу објектно-релационог мапирања засноване су суштински на смањењу броја захтева упућених бази података помоћу техника кеширања и дохватања унапред (енгл. *prefetching*) коришћењем предикције приступа подацима [6], [36]. Иако, у општем случају, смањење броја обраћања бази података има позитиван ефекат на перформансе, проблем приступања различитим (великим) табелама у релационом моделу са нормализованом шемом остаје и даље присутан.

2.3. Предности техника денормализације

Технике денормализације шеме релационог модела су ефикасне, али и „инвазивне“ технике оптимизације и захтевају измене шеме полазног (нормализованог) релационог модела у циљу елиминисања скупих релационих операција, као што је спајање табела [15] или смањења удела података који се не користе у трансакцијама спровођењем вертикалног партиционисања шема релација [37], [12]. Њиховом применом, подаци релевантни за операције које се оптимизују, компактније се групишу по јединици меморије, тачније по јединици преноса података, чиме се може драстично смањити потребан број преноса блокова са диска. Међутим, овим оптимизацијама последично се у релациони модел уносе аномалије при упису података и смањује ниво интегритета базе података. Због потребе за додатним операцијама одржавања конзистентности података, технике денормализације се скоро искључиво примењују у домену аналитичких система (где операције уписа не постоје у радном оптерећењу или су веома ретке, па додатне режије при упису могу да се толеришу) [15]. Посебно је значајна техника денормализације заснована на додавању редундантних података

(у даљем тексту кратко „денормализација“) јер представља елементарну трансформацију шеме релационог модела на најфинијем нивоу гранулације, на нивоу појединачних атрибута шема релације, чиме се обезбеђује максимална флексибилност поступка [15].

Још једна техника просторног груписања података којима се често приступа заједно употребљена је у оквиру две дистрибуиране базе података развијене у компанији Гугл, Google Cloud Spanner [38] и F1 [39], за потребе оптимизације приступа подацима у дистрибуираном окружењу и елиминацију кашњења при комуникацији између сервера. Обе базе података комбинују скалабилност нерелационих база података [40] (нпр. *BigTable* [41]) са поузданим механизмима за одржавање конзистентности података и моћним декларативним упитним језиком релационих база података. Због потребе за синхроном репликацијом, у циљу обезбеђивања конзистентности података, кашњење у извршавању дистрибуираних трансакција ублажава се коришћењем специјалног хијерархијског модела података, структурираних типова података (нпр. *Protocol Buffer*) и „паметним“ пројектовањем апликација према реалним потребама радног оптерећења [39].

Дистрибуирана база података Spanner [38] обезбеђује скалабилно складиштење података, синхрону репликацију, висок степен конзистентности података и поуздану обраду конкурентних дистрибуираних трансакција. Она захтева да се модел података подели на партиције, где свака партиција садржи подскуп табела модела података које су семантички и структурно повезане у односу „родитељ-дете“ и којима се често или увек приступа заједно [42]. Различите партиције се смештају на различите сервере, док се подаци у оквиру једне партиције смештају на исти сервер, при чему се структурно повезани редови просторно групишу заједно (учешљавањем, енгл. *interleaving*) у виду тзв. *директоријума* (енгл. *directory*) у циљу убрзавања приступа редовима при навигацији кроз хијерархију. Могуће је успоставити аналогију између наведене технике организације модела података у виду просторног груписања повезаних редова из различитих табела, која је од суштинске важности за добре перформансе приступа подацима у овој дистрибуираној бази података, и класичне денормализације релационе шеме која на сличан начин смањује кашњење операција, у овом случају, издатих блоковским уређајима за трајно чување

података. Ово је још један пример из праксе који недвосмислено указује на потенцијал и ефикасност техника оптимизације заснованих на просторном груписању повезаних података којима се често приступа заједно.

2.4. Основне хипотезе истраживања

У оквиру овог истраживања постављене су следеће хипотезе.

Х1: Начини приступа подацима у реалним ОО трансакционим апликацијама за управљање подацима у великим релационим базама података погодни су за оптимизацију применом технике денормализације релационог модела засноване на увођењу редундантних атрибута у шемама релација.

Х2: Селективним увођењем редундансе у релациони модел података, заснованим на анализи радног оптерећења и постизању компромиса између њеног позитивног утицаја на операције читања и негативног утицаја на операције уписа, може да се постигне ефикаснији приступ подацима у релационој бази података него применом постојећих техника кеширања, дохватања података унапред, као и оптимизовања апликативног изворног кода.

Х3: Селективно увођење редундансе у релациони модел података доприноси ефикаснијем приступу подацима у трансакционим системима у односу на неселективно увођење редундансе којим се занемарују њени негативни утицаји.

Х4: Поступак селективног увођења редундансе у релациони модел података може да се формално моделује и аутоматизује.

Х5: Предложена метода аутоматске денормализације, засноване на селективном увођењу редундансе у релациони модел података, може да се примени на реалне релационе моделе и велике базе података у пракси.

2.5. Шири контекст истраживања

У методологији развоја ОО софтвера вођеној моделима (енгл. *model-driven development*) концептуални ОО модели представљају централне артефакте, док се изворни код, као и релациони модел изводе из њега (нпр. аутоматским трансформацијама модела) [5]. Таква потпуно аутоматизована трансформација модела отвара нове могућности за организовање релационог модела на један „егзотичнији“ начин, наизглед необичан и неприродан за инжењере, али далеко ефикаснији за базу података. Такав начин организовања релационог модела могао би да обезбеди да изворни код, написан чак и неефикасно, не мора да пати од лоших перформанси, већ сам радни оквира за ОРМ у том контексту треба да омогући интелигентно пресликавање објектних акција на ефикасне упите релационог модела, уз подршку редундантно припремљених података за најчешће и/или најсложеније („најскупље“) трансакције које читају податке.

Ово истраживање представља само један у низу корака ка побољшању транспарентности технологије објектно-релационог мапирања и њене ефикасности у погледу приступа подацима у великим релационим базама података.

3. Преглед постојећих решења

У овој глави даје се детаљан преглед доступне литературе која је релевантна и уско повезана са проблемом који је предмет ове дисертације. Постојећа решења су класификована према начину оптимизовања приступа подацима у релационој бази података. Нека од решења се директно баве проблемом оптимизације објектно-релационог мапирања и у том смислу су сродна, док, са друге стране, постоје решења која су првенствено везана за проблем оптимизација упита на релационом нивоу, па су у овој тези наведена из тог разлога што садрже важне теоретске и практичне закључке који су од интереса за ово истраживање.

3.1. Класификација постојећих решења за оптимизацију приступа подацима

У овом поглављу даје се једна класификација анализираних постојећих решења за потребе јаснијег сагледавања принципа оптимизације приступа подацима у релационим базама података. Решења су према начину решавања поменутог проблема класификована у 6 категорија. За сваку од категорија прво се наводи симболичко име (за потребе концизнијег референцирања), као и кратак опис, а потом се прелази на опис решења. Категорије су уређене према нивоу апстракције на ком се спроводе, идући од апликативног ка релационом. Поред конкретних решења наведени су и прегледни радови који се сврставају у посебну секцију на почетку поглавља.

П: Прегледни радови. Радови из ове категорије не доносе специфична решења проблема, већ се искључиво баве прегледом постојећих решења и њиховом компаративном анализом.

К1: „Денормализација“ објектног концептуалног модела. Принцип решавања проблема базира се на прилагођавању концептуалног модела потребама

приступа подацима у радном оптерећењу. На бази профилизације радног оптерећења изводе се оптимизације базиране на реструктурирању класа и односа у објектном моделу које су сличне техникама денормализације на релационом нивоу.

К2: Елиминација антиобразаца у објектном коду за ОРМ. Принцип решавања проблема своди се на анализу изворног кода објектно оријентисаног софтвера за приступ подацима и откривање образаца неефикасног кодирања (тзв. антиобразаца), као и откривање погрешног подешавања радног оквира за ОРМ.

К3: Редукција приступа бази података дохватањем података унапред. Проблем се решава на нивоу објектно-релационог мапирања. Анализом приступа подацима формирају се предикције на основу којих се смањује број обраћања бази података коришћењем технике дохватања података унапред (енгл. *prefetching*).

К4: Додавање редундантних (секундарних) приступних структура. Решења из ове категорије доминантно долазе из домена оптимизација упита претраживања за потребе побољшања перформанси радног оптерећења. Проблем решавају увођењем индекса и материјализованих погледа, али се баве и проблемом њиховог идеалног распореда у релационом моделу према карактеристикама радног оптерећења.

К5: Додавање редундантних података у релациони модел. Принцип решавања проблема своди се на увођење редундантних података (колона) у релациону шему ради елиминисања операција спајања у упитима претраживања.

К6: Партиционисање и кластеровање релација. Принцип оптимизације приступа подацима своди се на вертикално и хоризонтално партиционисање полазне релационе шеме, као и на кластеровање табела, на основу анализе приступа подацима у радном оптерећењу.

3.2. Детаљан преглед релевантне литературе

За сваки научни рад који се презентује у овом поглављу у заглављу се наводи ознака категорије којој припада, према уведеној класификацији, година објављивања, референца и његов оригинални наслов на енглеском језику, а затим и извод најважнијих закључака који су од значаја за ову анализу.

П (2001, [18]): Performance Evaluation of the Object-Oriented Transformation Methodology

---

У овом раду [18] презентована је упоредна анализа перформанси SQL операција на релационим моделима добијеним а) традиционалним релационим моделовањем и б) мапирањем објектних модела на релационе помоћу унапређене објектно-релационе трансформације која се предлаже у том раду.

Аутори стављају фокус на унапређене традиционалне технике објектно-релационог мапирања релација генерализације-специјализације, агрегације, композиције и асоцијације на релационе моделе и поређење перформанси функционално еквивалентног дизајна до ког се долази традиционалним релационим моделовањем.

У раду је идентификовано пет различитих приступа објектно-релационом мапирању:

- **Објектно-релациона трансформација**: ОО парадигма користи се за моделовање проблема из реалног света, а затим се добијени ОО модел трансформише на релациони модел. Приступ објектно-релационог мапирања, описан у том раду, сврстава се управо у ову категорију. На овај начин се користе предности оба света – изражајност ОО парадигме за моделовање и поузданост и зрелост технологије релационих база података за складиштење и обраду података;
- **Објектни омотачи релационих система** (енгл. *object wrappers*): подразумевају постојање слоја софтвера изнад релационог модела који изграђује ОО функционалности. Један пример таквог софтвера јесу преводиоци упита написаних на објектном упитном језику (енгл. *Object Query Language*, *OQL*) на релационе SQL упите. Објектно-базирани погледи (енгл. *object views*) представљају још један пример објектних омотача. Подацима у релационој бази података приступа се кроз објектне погледе, док се објектни погледи у позадини своде на сложене операције спајања, селекције и пројекције над основним релационим табелама;
- **Објектно-релациони системи**: представљају еволутивни искорак класичних релационих система проширених подршком за објектне

концепте, као што су објектни идентификатори (OID), кориснички дефинисани типови, хијерархије типова и наслеђивање, енкапсулација и сакривање информација (енгл. *information hiding*), као и интеграција (позива) кориснички дефинисаних операција у релационим упитима;

- **Коегзистенција ОО и релационог система**: подразумева постојање интерфејса кроз који ОО систем приступа подацима у релационом систему. Такав интерфејс је објектни, а табеле и упити из релационог модела су енкапсулирани у виду објеката. Пример је *Borland Database Engine API* за објектне програме писане на језику C++;
- **Кооперација ОО и релационог система**: овај приступ је у употреби посебно у хетерогеним системима са више база података. У том контексту, обично постоји посебан систем који контролише све базе података, врши превођење и дистрибуцију извршавања упита. Тако је могуће да се за одређени упит део података учитава из ОО, а други део из релационе базе података.

П (2006, [15]): Denormalization Strategies for data retrieval from data warehouses

---

У овом раду презентоване су технике денормализације релационих модела за складишта података које се користе за побољшање перформанси сложених претрага података. Нормализација релационог модела представља фундаментални принцип дизајна база података који је важан за трансформацију софтверских захтева на семантички недвосмислен и робустан релациони модел података [15]. Међутим, уколико нормализован релациони модел не обезбеђује очекиване перформансе, пројектанти база података свесно одустају од нормалних форми и праве компромис између побољшања перформанси критичних претрага и одржавања конзистентности података у бази података. Денормализација релационог модел не мора нужно да представља лошу одлуку уколико се примењује паметно и у контролисаним условима [15].

У поменутом раду су идентификоване следеће стратегије денормализације релационог модела, које су прикупљене анализом ранијих „пионирских“ приступа [43], [44], [45]:

1) **Елиминисање релација** (енгл. *collapsing relations*) – трајно физичко спајање релација које се у упитима над нормализованим моделом често спајају; трансформација се може применити на све типове односа међу релацијама (релације у односу 1:1, 1:N, као и M:N);
2) **Партиционисање релација** (енгл. *partitioning relations*),
    a. вертикално партиционисање (енгл. *vertical partitioning*),
    b. хоризонтално партиционисање (енгл. *horizontal partitioning*)
3) **Додавање редундантних атрибута** (енгл. *adding redundant attributes*),
4) **Додавање изведених атрибута** (енгл. *adding derived attributes*).

Ефикасност сваке од наведених техника денормализације анализира се коришћењем модела добити и цене заснованог на процени броја приступа блоковима података у секундарној меморији (диску) и релационој алгебри као основи за извођење израза за цену и добит датих техника.

K1 (2013, [31]): Optimising Conceptual Data Models through Profiling in Object Databases

У овом раду аутори су се бавили проблемом оптимизације приступа подацима на нивоу концептуалног ОО модела [31]. Проучавали су проблем а) навигационог приступа подацима у релационој бази података, б) проблем дохватања сувишних података (атрибута, везаних објеката, енгл. excessive data), као и в) проблем увођења редундантних асоцијација у ОО модел, чијим се пресликавањем на релациони модел може убрзати навигациони приступ подацима. Међутим, пошто се трансформације раде над ОО моделом, промене нису транспарентне за постојећи изворни код. Да би се искористиле примењене оптимизације на нивоу ОО модела и пресликале на релациони модел, неопходне су измене изворног кода.

K2 (2014, [34]): Detecting Performance Anti-patterns for Applications Developed using Object-Relational Mapping

Предмет истраживања у овом раду јесте проблем неадекватне употребе радних оквира за ОРМ у развоју ОО апликација као узрок њихових лоших

перформанси. Такви идиоматски софтверски обрасци, који представљају један од главних узрока неефикасног приступа релационим базама података, називају се антиобрасци [34].

Тврди се да је неретка пракса у индустрији софтвера да се изворни код за ОРМ пише без потпуног увида у његов утицај на перформансе приступа подацима, а да је истовремено приметно и одсуство подршке у виду помоћних софтверских алата који би програмерима помогли да детектују критичне делове имплементације за ОРМ и делују проактивно у правцу отклањања таквих недостатака у ОО апликацијама. Показано је да се само финим подешавањем конфигурације радног оквира за ОРМ може побољшати време приступа подацима у релационој бази података једне комерцијалне ОО апликације, која је анализирана у том раду, за 98% у просеку.

Као скалабилно решење аутори овог рада предложили су аутоматизовани радни оквир за а) откривање антиобразаца за ОРМ у изворном коду и б) процену њиховог утицаја на перформансе у целини. Откривање антиобразаца у изворном коду имплементације заснива се на статичкој анализи изворног кода. Са друге стране, предложена процена утицаја сваког уоченог недостатка на перформансе апликације у целини, као подршка дефинисању приоритета отклањања недостатака у изворног коду, почива на ригорозним статистичким методама.

У поменутом раду, идентификована су два главна антиобрасца у изворном коду за ОРМ:

- дохватање сувишних података (енгл. *excessive data*),
- итеративно дохватање података (енгл. *one-by-one processing*).

Оба антиобрасца појављују се у контексту приступа објектима класа које су у релацији асоцијације и реч је о два антагонистичка ефекта.

Проблем дохватања сувишних података изражен је у случају када слој за ОРМ, поред објекта који је експлицитно захтеван у трансакцији, дохвати и све његове везане објекте. Такво понашање проистиче из чињенице да се асоцијациони крајеви у класама (које су у релацији асоцијације) имплементирају као поља/референце на објекте класа на супротним крајевима асоцијације. Због тога, приликом учитавања објекта у меморију, у циљу иницијализације свих поља

објекта (због семантике класа дефинисаних у програмском језику имплементације), радни оквири често подразумевано учитају и везане објекте како би се иницијализовала и поља-референце асоцијационих крајева. Ово је наравно нежељени ефекат у посматраном случају и због тога популарни комерцијални оквири за ОРМ имају механизме одложеног учитавања везаних објеката (енгл. *lazy loading*).

Са друге стране, проблем итеративног дохватања података јавља се у изворном коду за ОРМ који обично врши некакву обраду у циклусу за сваки објекат у одређеној колекцији, што резултује извршавањем великог броја ситних и мање ефикасних упита. У овом случају погодније је имати мање упита којима се учитава већи број потребних објеката. Због тога популарни радни оквири за ОРМ имају и механизам дохватања података унапред (енгл. *eager loading*). Да би се итеративна обрада учинила ефикаснијом, слоју за ОРМ може се дати „инструкција“ да при првом учитавању везаног објекта истовремено учита и додатне (везане) објекте које апликација, са великом вероватноћом, може захтевати у будућности (нпр. дохвати се првих N везаних објеката при првом обраћању бази података и тиме се уштеди N-1 упита за приступ осталим објектима). У неким случајевима, решавање проблема N+1 упита захтева и измене у самом објектном моделу (нпр. увођењем посебне класе за имплементацију асоцијација уместо коришћења простих референци за асоцијационе крајеве у изворном објектном моделу), уколико радни оквир за ОРМ не нуди потребна фина подешавања [46].

K2 (2016, [35]): Finding and evaluating the performance impact of redundant data access for applications that are developed using object-relational mapping frameworks

---

Претходно описано научно истраживање, презентовано у конференцијском раду [34], настављено је разрадом проблема непотребног приступа подацима (енгл. *redundant data access*) у ОО апликацијама развијеним коришћењем радних оквира за ОРМ. И у овом раду [35] је поновљена хипотеза да се радни оквири за ОРМ често користе недовољно оптимално и да њихово подразумевано функционисање најчешће не обезбеђује задовољавајуће перформансе приступа подацима, пошто они функционишу на ниском нивоу приступа подацима и мапирања врше

искључиво статички и праволинијски. Оптимизација приступа подацима типично захтева додатни инжењерски напор и време, често познавање имплементационих детаља технологије за ОРМ, као и познавање природе и динамике извршваања апликације [35].

У том раду је описано решење за откривање проблема непотребног приступа подацима (енгл. *redundant data access*) комбиновањем статичке анализе изворног кода и динамичке анализе трагова извршавања апликације и динамички генерисаних SQL упита у време извршавања. Оцена резултата до којих су аутори дошли применом ригорозних статистичких метода показала је да се елиминисањем учитавања непотребних података време одзива неколико анализираних комерцијалних ОО апликација може поправити за 17% у просеку. Предложено решење не уводи додатни ниво софтвера између ОО апликације и релационе базе података, већ се користи као помоћни алат и подршка инжењерима у току развоја софтвера.

Следећи типови проблема непотребног приступа подацима идентификовани су у овом раду:

- **Ажурирање свих атрибута објекта** (енгл. *update all*): када ОО програм промени вредности поља објекта и захтева његову перзистенцију, софтверска компонента за ОРМ типично мапира такву акцију на ажурирање вредности свих поља у одговарајућем реду у релационој бази података (јер се понекад, због елиминације додатне сложености у време извршавања, не врши праћење животног циклуса објекта и промена стања на нивоу појединих атрибута) [35]. Оваква поједностављена стратегија перзистенције објеката посебно може да угрози перформансе приступа бази података уколико се међу ажурираним атрибутима налазе велике текстуалне/бинарне вредности, или уколико постоје индекси над пољима (колонама) које се непрестано и непотребно уписују, трошећи тако време и ресурсе за ажурирање и самих индекса;
- **Читање свих атрибута ентитета** (енгл. *select all*): учитавање објекта у меморију подразумева иницијализацију свих његових поља специфицираних концептуалним моделом, без обзира на то који ће атрибути заиста бити коришћени у текућој трансакцији. У раду се тврди

да тај проблем проистиче из чињенице да ОРМ функционише на ниском нивоу приступа подацима и да због немогућности увида у природу и структуру апликативних трансакција може само да учита све податке објекта да би се обезбедило исправно функционисање програма [35]. Међутим, корен овог проблема делом потиче и због изворне семантике класа и објеката у објектно оријентисаним програмским језицима који се користе за представљање ентитета у меморији [5], [47]. У ОО програмским језицима стање објеката чине вредности њихових поља [28], па је за такве објекте у пракси одомаћен назив „обични објекти“ (нпр. *Plain Old Java Objects – POJO* и/или *Plain Old C# Objects – POCO*). Дохватање вредности атрибута објекта са класичном ОО семантиком своди се на приступ одговарајућем пољу објекта у меморији. У поменутом раду предлаже се спровођење динамичке анализе приступа подацима и предлагање могућих конфигурација слоја за ОРМ. Алтернативно решење може бити редефинисање семантике објеката у меморији за потребе ефикаснијег приступа подацима (вредностима атрибута) у релационој бази података, као што је на пример *OOIS UML* семантика [5], по којој присуство објекта у меморији не подразумева истовремено и присуство самих вредности његових атрибута;

- **Дохватање сувишних података** (енгл. *excessive data*): овај проблем је већ детаљно описан у претходном раду [34] и јавља се приликом учитавања објеката који су везани са другим објектима. Приступ подацима се у овом случају оптимизује експлицитним дефинисањем подграфа објеката који су неопходни за извршавање текуће трансакције, што захтева измене у објектним упитима;
- **Локално кеширање на нивоу трансакције** (енгл. *per-transaction cache*): уколико различите трансакције читају исте податке извршавајући идентичне упите, глобално кеширање, уместо локалног на нивоу појединих трансакција, може драстично да поправи перформансе приступа подацима.

К3 (2009, [36]): Speeding Up Data-driven Applications with Program Summaries

У овом раду [36] аутори су проучавали оптимизације приступа подацима у базама података коришћењем техника динамичке анализе извршавања програма. Традиционална технологија развоја апликативног софтвера за (релационе) базе података, чија је тенденција апстракција релационог модела, пресликава један принцип виших програмских језика по ком се подаци у програму дефинишу/дохватају тачно у тренутку првог коришћења. Аутори износе запажање да је за апликације за приступ подацима у базама података карактеристично то да бази података приступају учесталим издавањем елементарних SQL упита/акција [36], што је последица навигационог обрасца приступа подацима [6].

Оптимизације приступа подацима, ако и постоје у таквим апликацијама, углавном су локалног типа и на нивоу појединих (критичних) трансакција. Аутори зато предлажу оптимизације глобалног типа засноване на динамичкој анализи приступа подацима употребом концепта програмских сажетака (енгл. *program summaries*). Програмски сажетак представља образац приступа подацима у бази података током извршавања апликације који се добија анализом трагова навигације кроз објектни модел и представља стабло апстрактних навигационих путања (енгл. *path expressions*). Поменути програмски сажеци предвиђени су да се користе експлицитно у имплементацији као сложене објектне инструкције које неодољиво подсећају на OQL упите, по којима ОРМ може, интерпретацијом тих објектних сажетака, да учита податке који ће се заиста користити у текућој трансакцији. У том случају се каже да је (апстрактни) програмски сажетак материјализован.

Подразумевано, без концепта програмских сажетака, навигација кроз објектни простор врши се извршавањем низа елементарних SQL акција, или краће, навигацијом, чиме се постепено „материјализује траг извршавања програма“ [36]. У поменутом раду се овакав начин приступа подацима назива „проблем N+1 упита“ и манифестује се учитавањем почетног објекта, од ког почиње навигација, а затим и појединачним учитавањем сваког од његових N везаних објеката у циклусу. Програмски сажеци решавају овај проблем, тако што унапред дефинишу коначан траг програма, па се идентичан граф објеката може добити знатно ефикасније предложеним аутоматским трансформацијама упита на бази

интерпретације програмских сажетака у време извршавања. Међутим, ово решење није потпуно транспарентно за апликативни слој јер се програмски сажеци достављају слоју за објектно-релационо мапирање управо са апликативног слоја, који има информације о контексту приступа подацима.

КЗ (2000, [6]): Context-based prefetch – an optimization for implementing objects on relations

У овом раду [6] проучаван је проблем ефикасности приступа перзистентним објектима у релационој бази података, са нагласком на смањењу броја обраћања бази података.

Једна од најважнијих карактеристика објектно-релационих система, како се наводи у том раду, јесте могућност да се перзистентни објекти (у релационој бази података) трансформишу у објекте у меморији према одговарајућем објектном моделу извршног окружења [6]. И док се на овај начин премошћава јаз између објектне и релационе парадигме, са друге стране остаје отворен проблем ефикасности комуникације са базом података [6].

Основни проблем лоших перформанси генеричких приступа ОРМ-у лежи у чињеници да су апликативни објектни модели по својој природи навигациони [6]. Навигација кроз објектни простор, вођена структурним везама између објеката, најчешћи је образац програмирања функционалности ОО апликација. Свако учитавање везаног објекта преко асоцијационог краја захтева обраћање бази података. Иако кеширање у таквим случајевима може да побољша перформансе приступа, оно само по себи нема смисла уколико се кешираним објектима у току трансакције не приступа више од једног пута. Међутим, кеширање би у таквом случају имало смисла само ако би се некаквом предикцијом учитали унапред (енгл. *prefetch*) сви објекти које ће програм сигурно користити, тако да је питање добијања квалитетне предикције основно питање у поменутом раду.

Решење које аутори предлажу своди се на дохватање података унапред (енгл. *data prefetching*) унапређено концептом *контекста* (енгл. *structure context*). *Контекст* представља образац приступа подацима, у навигационом смислу, представљен као апстрактно објектно стабло. За сваки објекат у кешу памти се контекст у ком је учитан. Уколико се у току исте сесије са базом података појави

потреба за читањем другог објекта (исте класе, мапиране на исту табелу), врши се дохватање унапред свих података према навигационим путањама у апстрактној објектној структури важећег контекста.

Иако дати модел дохватања података унапред са предикцијом поправља перформансе приступа подацима, јер драстично смањује број обраћања бази података, постоје и одређени проблеми о којима треба водити рачуна, ако се једно такво решење имплементира:

- одржавање контекста за сваки објекат, као и животни циклус структуре контекста (контекст настаје са учитавањем објекта и нестаје са крајем животног века објекта у меморији),
- елиминисање вишеструког учитавања истог објекта у различитим контекстима,
- иницијално кашњење при учитавању већих објектних структура унапред; делом се овај проблем може поправити асинхроним учитавањем унапред, али опет остаје да се реши проблем синхронизације код конкурентног приступа подацима у кешу,
- одржавање стања предиктора.

Као што је показано и у овој докторској тези, и поред драстичног смањења броја обраћања бази података које обезбеђује решење са дохватањем података унапред, и даље остаје отворено питање ефикасности самих упита, као и потреба за вишеструким спајањем табела у бази података. Иако посматрани приступ [6] смањује број захтева упућених бази података, њиме се не елиминише суштинска сложеност приступања већем броју релација у нормализованом релационом моделу.

K3 (2006, [32]): Automatic Prefetching by Traversal Profiling in Object Persistence Architectures

У посматраном раду је презентована техника дохватања унапред под називом *AutoFetch* која представља проширење слоја за објектно-релационо мапирање за оптимизацију извршавања објектног кода динамичким, аутоматским и транспарентним проширењем објектних упита на основу статистичког модела

приступа подацима добијеног анализом навигације у објектном простору. Ручно додавање спецификација дохватања унапред у објектне упите није скалабилно решење и подложно је грешкама, посебно у сложеним и модуларним програмима у којима није једноставно утврдити којим подацима приступити унапред. Анализом упита формирају се класе образаца приступа подацима у које се класификују упити са сличним навигационим обрасцима чиме се поједностављује модел за предикцију приступа подацима. Недостатак овог приступа јесте тај што захтева да се сваки упит први пут изврши без оптимизација док се не иницијализује предиктор приступа.

K4 (1997, [48]): An efficient, Cost-Driven Index Selection Tool for Microsoft SQL Server

---

У посматраном раду разматра се проблем аутоматске селекције индекса вођен информацијама и статистикама операција за приступ подацима добијених из задатог радног оптерећења.

За изводљивост идеје о аутоматској селекцији индекса неопходна је тесна спрега за оптимизатором упита јер су изабрани индекси онолико добри колико оптимизатор упита може да их употреби [48]. Због тога модел употребе индекса у описаном решењу за селекцију индекса мора да буде конзистентан са самим моделом употребе индекса који користи оптимизатор упита [48]. Као мера квалитета изабраних индекса користи се укупна цена упита у задатом радном оптерећењу. Скуп индекса у релационог моделу назива се *конфигурација*. *Оптимална конфигурација* карактерише се минималном укупном ценом упита у задатом оптерећењу.

Ефикасност методе за одабир индекса квантификује се помоћу неколико показатеља квалитета: а) бројем индекса који су разматрани у току претраге за оптималном конфигурацијом, б) бројем конфигурација које су предложене и в) бројем захтева упућених оптимизатору упита [48].

Посебна пажња је посвећена хеуристикама претраге за минимизовање позива оптимизатору упита увођењем концепта *атомичних конфигурација*. Конфигурација је атомична за задато радно оптерећење, ако постоји упит за чије извршавање могу бити искоришћени сви индекси у конфигурацији [48]. Одабир

атомичних конфигурација поједностављује се тиме што се за сваки упит разматра само ограничен и мали број индекса по табели у упиту јер у пракси оптимизатори упита не користе више од највише неколико индекса по табели [48]. Иза ове одлуке стоји емпиријска чињеница да на цену упита највећи утицај имају индекси који се користе у првих неколико операција спајања табела (енгл. *joins*) [48]. Предложени алгоритам за избор оптималне конфигурације подразумева прво одабир најбољих конфигурација за сваки упит у радном оптерећењу појединачно, а затим се коначна конфигурација формира као унија индекса у добијеним локално оптималним конфигурацијама.

K4 (1998, [49]): AutoAdmin "What-if" Index Analysis Utility

Фино подешавање перформанси база података у продукцији је важан, али и комплексан задатак, који је практично немогуће ефикасно спровести без одговарајуће подршке у виду помоћних алата [49]. Посебно је неопходно имати могућност испробавања различитих хипотетичких индекса (енгл. "*what*-if") без директног утицаја на постојећи дизајн и трансакције у продукцији [49]. Хипотетичка конфигурација индекса може да укључује постојеће (већ имплементиране) индексе, као и хипотетичке индексе [49]. Предложена метода подразумева симулацију хипотетичке конфигурације кроз а) анализу утицаја на цену упита у радном оптерећењу и б) анализу искоришћења индекса. Употребна вредност индекса искључиво зависи од тога да ли ће га сам оптимизатор упита узети у обзир у време извршавања [49]. Подршка за хипотетичке индексе, као и пратећа проширења оптимизатора упита уграђена су у верзију SQL Server 7.0 [49], и од тада су доступни у свим наредним верзијама. Након десет година рада и истраживања овог проблема и побољшања у систему за управљање базама података SQL Server аутори су објавили и ретроспективни рад [50] у ком су закључили да се ради о крупном проблему који се не може једноставно решити. Као посебну забрињавајућу чињеницу и ограничавајући фактор добијања софистицираног и аутоматског подешавања база података навели су то да постојећи системи за управљање базама података нису иницијално пројектовани са идејом аутономног подешавања, па су временом постали веома комплексни кроз непрестано унапређење компонената за обраду упита на једном таквом моћном језику као што је SQL [41].

K4 (2000, [51]): Automated Selection of Materialized Views and Indexes for SQL Databases

Оптимизације приступа подацима у релационим базама података обично се своде на додавање редундантних приступних структура за убрзавање извршавања упита, као што су материјализовани погледи и индекси [51]. У овом раду детаљно је обрађен проблем аутоматизације избора материјализованих погледа и индекса који обезбеђују оптимално (или скоро оптимално) извршавање унапред задатог радног оптерећења (енгл. *workload*) у смислу минимизације времена одзива, као и укупне цене свих упита.

Скуп или селекција материјализованих погледа и индекса у релационом моделу у посматраном раду назива се *конфигурација* [51]. Иако су индекси и материјализовани погледи концептуално слични, материјализовани погледи пружају већи степен слободе и могућности за оптимизације јер: а) могу бити дефинисани над више табела и б) подржавају комплекснију синтаксу упитног језика SQL [51]. Са већим могућностима које ове структуре пружају експоненцијално се повећава простор претраге оптималних (или субоптималних) конфигурација. У поменутом раду се посебно истиче то да је неопходно анализирати и ефекте интеракције између индекса и материјализованих погледа у релационом моделу јер они значајно утичу на квалитет решења [51]. Другим речима, избор одређеног материјализованог погледа утиче на избор релевантних индекса, и обрнуто.

Поменути рад настао је у склопу пројекта *AutoAdmin* компаније Мајкрософт који је обухватао истраживачки рад на пољу аутоматизације пројектовања релационих база података за SQL Server. Претходило му је истраживање на пољу аутоматизације избора индекса у релационим базама којима управља SQL Server [48], као и рад на тему анализе хипотетичких индекса [49], као основа и подршка пројектовању база података за SQL Server.

K4 (2001, [52]): A Formal Perspective on the View Selection Problem

У овом раду [52] разматран је проблем избора погледа за оптимизацију навигационих упита (енгл. *conjuctive queries*) који садрже операције спајања, пројекције и селекције по једнакости (енгл. *equi-joins*). Такви упити садрже серију

линеарног спајања релација, при чему је број редова у коначном резултату знатно мањи него у релацијама које се спајају [52]. У раду је разматрано неколико питања проблема избора упита за материјализацију: а) које погледе укључити у оптималну конфигурацију у зависности од упита у трансакцијама у радном оптерећењу, б) која је кардиналност оптималне конфигурације погледа у односу на величину базе података, као и в) која је сложеност проблема избора погледа.

Линеарно спајање релација само је специјалан случај обрасца приступа подацима који се у овој докторској дисертацији назива и *линеарна навигација*. Исто тако, као што ће бити илустровано касније у овој дисертацији, индексирање не мора увек да обезбеди оптимално време одзива; биће показано и то да технике оптимизације, предложене у овој докторској дисертацији, могу значајно да надмаше технике базиране искључиво на индексирању.

K4 (2011, [53]): CoPhy: A Scalable, Portable, and Interactive Index Advisor for Large Workloads

---

Ово је први рад, како аутори тврде (2011. година), који је показао да проблем избора скупа индекса (*конфигурација*) на основу карактеристика задатог радног оптерећења има структуриран простор решења који се може ефикасно претраживати применом техника линеарних оптимизација [53].

Претходна решења [51], [37], [49], [48], почивају на бројним експертским претпоставкама и ограничењима за скраћивање експоненцијалног простора претраге, што доводи често до губитка оптималних конфигурација [53]. Такве конфигурације често нису уочљиве експертима на први поглед. Са друге стране, претходна решења [51], [37], [49], [48] зависе од оптимизатора упита на основу хипотетичких конфигурација (енгл. *what-if optimizer*). Позив датом оптимизатору упита сматра се „скупом“ операцијом, па је минимизација позива оптимизатору још једно практично ограничење које утиче на квалитет добијених конфигурација зарад ефикаснијег алгоритма претраге простора решења.

Аутори посматраног рада тврде да се било који проблем избора индекса, који зависи од брзог „what-if“ оптимизатора, своди на бинарно целобројно програмирање (енгл. *binary integer programming*) у којем број променљивих расте линеарно са бројем упита и ограничења. Неколико ограничења је уведено у циљу

постизања линеарног модела ограничења: а) не разматрају се ефекти вишеструких индекса над истим табелама, б) ограничења која се не могу записати линеарним једначинама нису подржана.

К4 (2013, [54]): Incremental mapping compilation in an object-to-relational mapping system

---

У овом раду се разматра метода за анализу и превођење концептуалних ОО модела на интерну релациону репрезентацију. Аутори овог рада разматрали су материјализоване погледе са редундантним подацима који убрзавају упите читања. Они су се посебно осврнули на дисконтинуитете у мапирању (енгл. *lossy mappings*), који се манифестују неконзистентним ажурирањем података у основним табелама и њихових редундантних копија у материјализованим погледима. Радни оквири за ОРМ најчешће уопште не одржавају конзистентност копија података, већ је то по правилу искључиво одговорност програмера да пише и одржава гломазан код за одржавање конзистентности података, уместо да се фокусира на решавање захтева из домена проблема [54].

К5 (2013, [55]): On Redundant Data for Faster Recursive Querying Via ORM Systems,

---

Увођењем редундансе у релациони модел може се убрзати извршавање претрага података у рекурзивним структурама података као што су хијерархије и мреже, које се неретко појављују у моделима пословних организација [55].

Наивна имплементација навигације кроз такве рекурзивне структуре података обично се своди на итеративну неефикасну имплементацију на неком језику треће генерације [55]. Примери приступа подацима у рекурзивним структурама јесу проналажење (кореног) надређеног елемента за сваки елемент у хијерархијској структури података или дохватање свих елемената са истим заједничким надређеним елементом. Два су начина да се такви приступи подацима убрзају: а) трансформацијом („размотавањем“) упита и б) увођењем редундансе у релациони модел [55].

Размотавање упита (енгл. *query unrolling*) може бити а) хоризонтално, вишеструким спајањем исте релације неколико нивоа унапред, или б) вертикално,

издавањем неколико упита у једном обраћању бази података (енгл. *batch*) којима се поступно дохватају парцијални резултати [55]. Међутим, са порастом броја елемената хијерархије долази до изражаја сложеност операције спајања, као и негативни ефекти учесталог извршавања малих и неефикасних упита код вертикалног размотавања, при чему су експериментални резултати у том раду показали да хоризонтално размотавање даје боље време одзива упита од вертикалног [55].

Уколико се навигација кроз хијерархијску или мрежну структуру врши учестало, технике материјализације редундантних података обезбеђују далеко боље перформансе приступа подацима у односу на технике размотања упита [55]. Четири илустроване технике у раду [55], засноване на редундантним подацима, деле се на две групе по две.

У прву групу спадају две технике које захтевају додавање колона у постојеће табеле. То су а) угнеждени скупови (енгл. *nested sets*) и б) размотане стазе (енгл. *materialized paths*) [55]. Угнеждени скупови имплементирају се коришћењем интервала позиција у стаблу хијерархије. Сваки елемент у хијерархији добија јединствен индекс одговарајућим обиласком стабла (*pre-order*) и додатна два индекса: почетак и крај интервала. Индекси свих елемената који припадају подстаблу посматраног елемента припадају његовом придруженом интервалу. Исто тако, придружени интервал посматраног елемента представља подскуп интервала њему надређеног елемента. Код размотаних/материјализованих стаза, за сваки елемент хијерархије чува се (у додатној колони) путања до кореног елемента (као секвенца идентификатора), али се на тај начин крши прва нормална форма [10].

У другу групу техника заснованих на редундантним подацима сврставају се две технике које се имплементирају помоћу додатних табела у релационом моделу, а то су: а) комплетне стазе (енгл. *full paths*) и б) логаритамске стазе (енгл. *logarithmic paths*) [55]. Комплетне стазе су сличне материјализованим стазама, једино што се путање до кореног чвора чувају у посебној нормализованој табели уместо у једној колони (која нарушава прву нормалну форму). У случају дубоких хијерархија, број редундантних пуних путања расте са квадратом дубине хијерархије. Код логаритамских стаза, у посебној табели се чувају само путање чија

је дужина степен броја 2. На тај начин се постиже компромис између потребног простора за смештање редундантних веза и броја спајања у односу на комплетне стазе. Све у свему, у поменутом раду је показано да технике увођења редундантних путања у хијерархијским структурама значајно убрзавају приступ подацима, чак и у односу на технике размотавања упита, као и то да је свака од техника погодна за одређене обрасце приступа подацима [55].

К6 (2004, [37]): Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design

Поред избора индекса и материјализованих погледа, хоризонтално и вертикално партиционисање релација су важни аспекти пројектовања релационих база података који значајно поправљају перформансе упита, према наводима у посматраном раду [37]. Партиционисање је посебно важно у пројектовању скалабилних решења за претрагу података у великим базама података и дистрибуцију оптерећења [37].

Код хоризонталног партиционисања табела (енгл. *horizontal partitioning*) веома је важно водити рачуна о „поравнању“ (енгл. *alignment*) партиција и индекса над њима јер то поједностављује рутинске операције над базом података, као што су чување резервних копија (енгл. *backup*) и враћање базе података у претходно конзистентно стање (енгл. *restore*) [37]. Вођењем рачуна о поравнању, операције спајања се значајно убрзавају уколико су структуре (нпр. табеле и индекси), које се могу семантички спајати, партиционисане идентично (над истим заједничким колонама, енгл. *co-located*). Избор оптималног партиционисања табела, индекса и материјализованих погледа су доказано NP-тешки проблеми [37]. За разлику од симулације хипотетичких индекса [51], симулација партиционисања је значајно комплекснија (има већу цену) јер захтева а) стварање физичких партиција у релационом моделу и б) прераду постојећих упита да користе нове хипотетичке фрагменте уместо оригиналних табела [37].

Поред свега, формално моделовање међусобне интеракције датих структура од посебног је значаја за добијање оптималних конфигурација, као и за елиминацију невалидних/немогућих конфигурација, узимајући у обзир и физичка ограничења која систем за управљање базом података и оптимизатор упита намећу

[37]. Предложено решење за налажење оптималне конфигурације полази од прикупљених статистика радног оптерећења (енгл. *workload statistics*) и састоји се од неколико алгоритама (нпр. проналажење корелисаних група колона за вертикално партиционисање, спајање вертикалних кандидат-партиција), који раде на принципу минимизације цена упита у узорку радног оптерећења (добијених од оптимизатора) за задате хипотетичке конфигурације. Експериментални резултати су потврдили ефективност експертских хеуристика за ограничавање простора претраге оптималних конфигурација и указали на значај моделовања интеракција између оптимизационих структура.

K6 (1984, [12]): Vertical Partitioning Algorithms for Database Design

---

Вертикално партиционисање релација подразумева деобу шеме релације на „уже“ релације (фрагменте) које обухватају (недисјунктне) подскупове атрибута полазне шеме релације, са идејом да се створе релације које су скројене према захтевима трансакција [12]. Вертикални фрагменти смањују обим улазно-излазног преноса података са секундарном меморијом јер резултати упита заузимају (знатно) мање страница [12], уз повећање густине релевантних података за трансакције по страници. У поменутом раду [12] презентоване су две класе алгоритама за аутоматско вертикално партиционисање:

- **Алгоритми засновани на емпиријским функцијама циља**: груписање колона релација врши се према афинитету, као мери заједничког коришћења (енгл. *usage*) у трансакцијама. Колоне које се често заједно користе у трансакцијама имају већу вероватноћу да се заједно нађу у вертикалним фрагментима. Описано је неколико емпиријских начина партиционисања као што су: а) партиционисање на дисјунктне скупове атрибута, б) партиционисање са преклапањем атрибута, ц) партиционисање на примарне и секундарне фрагменте, д) понављајуће (енгл. *repetitive*) бинарно партиционисање;
- **Алгоритми засновани на оптимизацији цене**: решавају проблеме емпиријских алгоритама који нису често довољно прецизни јер предложени линеарни алгебарски изрази за афинитет колона нису довољно софистицирани, већ се користе само као брзе и једноставне

хеуристике у процесу планирања вертикалног партиционисања. Цена дизајна, добијена вертикалним партиционисањем, је вишедимензионална функција која укључује следеће компоненте: а) цену приступа ирелевантним колонама у фрагменту (по бајту), б) цену приступа фрагментима при читању и упису (по приступу), в) цену алокације (по бајту), г) цену преноса података (по бајту).

Углавном, технике денормализације су посебно ефикасне када су примењене у аналитичким системима (OLAP) јер се такви системи карактеришу малим бројем операција ажурирања података, у поређењу са операцијама читања и претрага, па одржавање конзистентности редундантних структура има прихватљиву цену. Међутим, у трансакционим (OLTP) системима, описане технике денормализације не могу се примењивати неограничено, без систематског разматрања њиховог (негативног) утицаја на операције ажурирања података и одржавање података конзистентним у реалном времену.

K6 (1989, [56]): Integrating Knowledge-Based Component Into a Physical Database Design

Важност знања о природи и извршавању програма за моделовање ефикасног релационог модела истакнута је и детаљно обрађена у овом раду [56]. Аутори тог рада формулисали су проблем пројектовања шеме релационог модела на основу информација из радног оптерећења као проблем одлучивања за чије решавање су предложили један експертски систем. Експертски систем је заснован на скупу правила одлучивања која су допуњена статистикама приступа подацима, као што су: фреквенције читања, фреквенције уписа, фреквенције појединачних упита, фреквенције појединачних спајања итд. Полазећи од нормализованог концептуалног модела ентитета и односа, датим приступом релациони модел се трансформише на скуп кластера ентитета на основу корелације приступа у оквиру анализираних трансакција. Насупрот томе, приступ кластеровања података, који је предложен у овој тези, ради на финијем нивоу гранулације података, разматрајући свако својство типа ентитета посебно у циљу боље контроле повећања потребног меморијског простора.

4. Предлог и опис решења

У овој глави дат је неформалан опис предложеног решења за проблем дефинисан у глави 2. Принципи датог решења илустровани су на упрошћеном практичном примеру. Поред описа решења, неформално се уводе и концепти који су неопходни за даљу анализу, а који ће бити и формално дефинисани у глави 5.

4.1. Опис радног примера

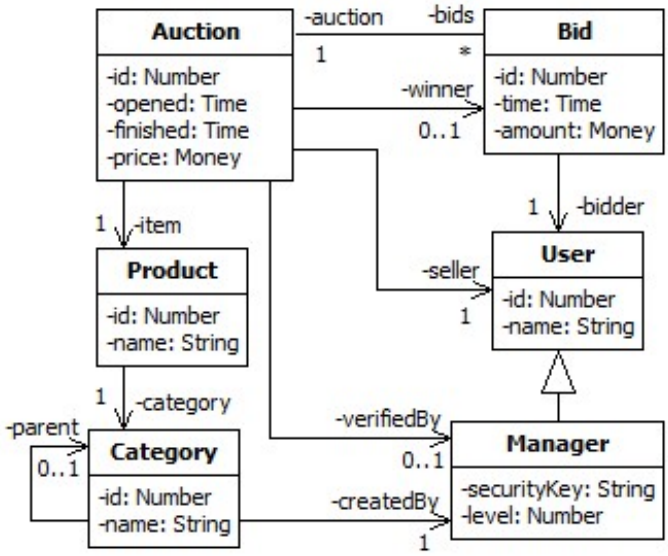
За потребе илустрације предложене методе оптимизације објектно-релационог мапирања користиће се поједностављен ОО концептуални модел једног хипотетичког веб портала за аукцијску продају робе на Интернету (у даљем тексту „аукцијска апликација“) који је приказан на слици 1. Изабрана апликација погодна је за разматрање пре свега као типичан пример трансакционих апликација за које је предложена метода осмишљена и на коју се може применити. Апликација се карактерише великим бројем корисника и великим бројем трансакција у јединици времена које се обрађују.

Иако је сам концептуални модел, приказан на слици 1 коришћењем језика за моделовање UML, довољно једноставан и интуитивно разумљив, у следећем поглављу се наводи само сажетак најважнијих детаља из изабраног домена проблема.

4.1.1. Концептуални опис домена

Када корисник (енгл. *User*) треба да огласи производ за продају, отвара нову аукцију (енгл. *Auction*) и попуњава захтеване детаље производа (енгл. *Product*). Сваки производ класификује се у тачно једну категорију (енгл. *Category*), док свака категорија има надређену категорију (енгл. *parent*). Једино категорија на врху хијерархије, корена категорија, нема надређену категорију, што се моделује

усмереном асоцијацијом са асоцијационим крајем *parent* чија је мултипликативност 0..1 (слика 1). Када корисник комплетира нову аукцију, додељује јој се администратор (енгл. *manager*) чији је задатак да изврши ауторизацију аукције са циљем да је одобри или да је врати на дораду кориснику (енгл. *seller*). Након што аукција отпочне, други корисници (купци, енгл. *bidder*) могу да дају понуде (енгл. *Bid*) све док се она не заврши. Последња понуда у тренутку затварања аукције проглашава се победничком (енгл. *winner*) и експлицитно се евидентира у објекту аукције.



Слика 1: Фрагмент концептуалног модела аукцијске апликације

4.1.2. Опис случајева коришћења веб портала за аукцијску продају

Поред концептуалног модела и статичких детаља, датих у виду класа и асоцијација, неопходно је размотрити и релевантне начине приступа подацима од стране апликације. За потребе радног примера издвојено је неколико поједностављених случајева коришћења из концептуалног модела и они су приказани у табели 1.

Сваки случај коришћења (енгл. *use case*) означен је кратким симболичким именом (нпр. *UC1*), има кратак опис, број појављивања (добијен анализом извршавања апликације) и детаљну спецификацију приступа подацима / атрибутима класификованих по припадности класама у концептуалном моделу.

Случајеви коришћења *UC1* и *UC2* предвиђени су за генерисање извештаја и садрже искључиво операције читања података. У случају коришћења *UC1* апликација враћа резултат издавањем упита приказаног у листингу 1. Датим упитом спаја се неколико табела да би се приступило свим захтеваним подацима због тога што је релациони модел нормализован и изведен из објектног модела на слици 1 применом усвојене стратегије објектно-релационог мапирања (описане у поглављу 1.4.1).

ТАБЕЛА 1:
ЛИСТА НАЈВАЖНИЈИХ СЛУЧАЈЕВА КОРИШЋЕЊА АУКЦИЈСКЕ АПЛИКАЦИЈЕ

UC1: Приказ активних аукција (5.000)	
Auction:	opened, finished, item, last bid
Product:	name, category
Category:	name
Bid:	time, amount, bidder (last bid)
User:	name (bidder)
UC2: Приказ детаља аукције (2.000); 500/2.000 завршених аукција	
Auction:	opened, finished, all bids, seller, winner, item, manager
Product:	name, category
Category:	name, parent (full path)
Bid:	time, amount, bidder
Manager:	name
User:	name (for bidder and seller)
UC3: Давање понуде (4.000)	
Bid:	create (time, amount, auction)
UC4: Стварање нове аукције (100)	
Product:	create (id, name, category)
Auction:	create (id, started, product, seller)
UC5: Затварање аукције (100)	
Bid:	id
Auction:	winner

Слично томе, ОО изворни код приказан у листингу 2 приступа свим релевантним детаљима задате аукције према спецификацији случаја коришћења *UC2*.

Листинг 1:
Упит Q1: враћа списак активних аукција према спецификацији случаја коришћења UC1

```
select p.name, a.opened, c.name, b.time, b.amount, ub.name
from Auction a
join Product p  ON (p.id = a.item)
join Category c ON (p.category = c.id)
join User ub    ON (ub.id = b.bidder)
join Bid b      ON (b.auction = a.id) [last by b.time]
where  a.opened < now() and a.finished is null
```

Листинг 2:
Имплементација случаја коришћења UC2 у псеудојезику сличном програмском језику Јава

```
Auction a = findById(100);
Product p = a.item();
// Display general information
display (
       p.name(),    // product name
       a.opened(),
       a.finished(),
       a.seller().name(),  // seller's name
       a.manager().name(), // manager's name
);

// Display full product category path
Category c = p.category();
while (c != null) {
   display (c.name());
   c = c.parent();
}

// Display all bids
foreach (Bid b in a.bids()) {
   // Display bid information
   display (b.bidder().name(), b.amount(), b.time());
}

// Display winning bid if it exists
Bid w = a.winner();
if (w != null) {
   display(w.amount(), w.time(), w.bidder().name);
}
```

За разлику од случаја коришћења *UC1*, који је имплементиран као упит, случај коришћења *UC2* је имплементиран императивно, као апликативна процедура, због потребе за динамичким читањем свих надређених категорија

произвољне дубине, као и условног читања победничке понуде. Пошто дати програмски код учитава и ради са неколико објеката различитих (перзистентних) класа (*Auction*, *User*, *Category*, *Bid*), његовим извршавањем издаје се више (нпр. 10) узастопних упита бази података, узимајући у обзир усвојену стратегију објектно-релационог мапирања. Може се приметити да извршавање посматране процедуре почиње дохватањем изабране аукције, а затим се свим осталим објектима, повезаним са посматраном аукцијом, приступа искључиво навигацијом преко асоцијационих крајева. Каже се да овај програм врши навигацију кроз објектни простор почевши од објекта аукције да би прибавио све објекте који су неопходни за имплементацију понашања задатог у опису случаја коришћења *UC2*. На пример, позивом операције `a.item()` приступа се објекту производа који се продаје на аукцији на коју указује референца `a`. Сваки пут када се дохвати иста аукција резултат позива операције `a.item()` биће идентичан, односно, на основу задате аукције једнозначно се може одредити производ који је на њој продаје захваљујући постојању структурне везе између њих.

4.2. Метамодел ентитета и односа

У овом поглављу уводи се и усвојени метамодел података са концептима на којима почива наставак анализе проблема. Дати метамодел података је осмишљен тако да обухвати како објектне, тако и релационе концепте, у циљу добијања формалне и генеричке представе начина приступања подацима.

4.2.1. Ентитети

Од сада па надаље у овој дисертацији, класе и типови података у ОО моделу, као и релације у релационом моделу, моделују се концептом *типа ентитета* (енгл. *entity type*). Тип ентитета третира се као скуп ентитета. *Ентитет* представља примерак типа ентитета и карактерише се идентитетом којим се разликује од других ентитета. Идентитет ентитета одређен је вредношћу специјалног својства типа ентитета под називом *идентификатор*. Инстанце примитивних типова података представљају специјалне (предефинисане) ентитете чији је идентитет одређен самим њиховим вредностима (литералима). Вредност примитивног ентитета уједно је и вредност његовог идентификатора.

4.2.2. Својства типова ентитета

Атрибути класа и асоцијациони крајеви посматрају се као *односи* (енгл. *relationships*) између типова података у ОО моделима [4], [22]. За такве односе користи се заједнички термин *својства* типова података (енгл. *properties*) [22]. У ОО моделима, свако својство има дефинисан тип вредности. Поред типа вредности, својства имају и придружено ограничење мултипликативности којим се дефинише максималан број вредности које један објекат може да садржи у пољу које одговара датом својству (типично највише 1, ако се не прецизира другачије).

У датом метамоделу својство се третира као бинарна релација између два скупа (типа) ентитета, од којих један представља домен, и одговара типу ентитета који садржи дато својство у ОО моделу, док други представља кодомен релације и одговара типу вредности датог својства у ОО моделу. Сваком својству у овом метамоделу придружује се ограничење мултипликативности којим се дефинише максималан број ентитета у кодомену који могу бити у релацији са једним ентитетом у домену датог својства. Уколико је максимална вредност мултипликативности својства типа ентитета једнака 1, реч је о функционалном својству или функцији. Тип ентитета који представља кодомен функционалног својства означава се и као *функционално зависан тип ентитета*. На пример, асоцијациони крај *item* у класи *Auction*, чија је мултипликативност подразумевано једнака 1 (типично у UML-у), моделује се према предложеном метамоделу као функционално својство које пресликава тип ентитета *Auction* на тип ентитета *Product*. Једном успостављене, (структурне) везе између ентитета присутне су све док се експлицитно не измене операцијама уписа. Однос генерализације-специјализације такође може да се моделује као пресликавање специјализованог типа ентитета у основни тип ентитета из ког се директно специјализује.

4.2.3. Детаљи о перзистенцији својстава типова ентитета

Према усвојеној стратегији за ОРМ, функционална својства класа пресликавају се на атрибуте шема релација. Вредности примитивних типова својстава физички се смештају у пољима (колонама) редова. На пример, вредности текстуалног типа (стрингови) смештају се физички у поља редова у бази података, уместо индиректног указивања на њих коришћењем некаквих специјалних идентификатора (страних кључева). Када се један ред из табеле учита у оперативну

меморију из базе података, све „садржане“ вредности у његовим пољима постају доступне програму без потребе за додатном навигацијом и приступом бази података.

Са друге стране, уколико се у пољу реда табеле налази идентификатор ентитета неког непримитивног типа (као симболичка веза, енгл. *symbolic link* [56]) који се трајно чува у другој табели или у другом реду исте табеле, приступ вредностима својстава тог везаног ентитета захтева додатну операцију читања у бази података. Приступ ентитету на основу задатог идентификатора захтева прво његову локализацију у простору ентитета, а затим и пренос лоцираног ентитета на локацију за обраду. На пример, приступ реду у табели на основу идентификатора, захтева прво одређивање његове физичке адресе у фајлу, а затим и пренос физичке странице (која садржи дати ред) са диска у оперативну меморију.

Основна претпоставка на којој се заснива ово истраживање и даља анализа јесте та да је, у општем случају, приступ некој вредности из већ учитаног реда табеле у бази података значајно „јефтинији“ од приступа који би захтевао дохватање додатног реда из друге или исте табеле [57], [58].

4.3. Функционалне зависности типова ентитета

Сва својства у једном типу ентитета функционално су зависна од његовог идентификатора. Уколико је неки тип ентитета Y функционално зависан од неког типа ентитета X, онда су сва својства типа ентитета Y транзитивно функционално зависна од идентификатора типа ентитета X. Постојање такве функционалне зависности омогућава да се сва својства типа ентитета Y, или један њихов подскуп, могу додати (или, у екстремном случају, преместити) у тип ентитета X. На тај начин може да се елиминише потреба за посебним приступом неком (функционално зависном) ентитету типа Y сваки пут када је поред приступа вредностима својстава ентитета типа X потребно приступити и вредностима својстава ентитета типа Y (која се редундантно додају у тип ентитета X). Такође, на описани начин се елиминише потреба за додатном навигацијом и вишеструким и мање ефикасним навигационим приступима бази података, а уједно и повећава проценат релевантних података учитаних по једном приступу бази података у текућој трансакцији [59]. За описани поступак увођења редундантних транзитивно

функционално зависних својстава у неки тип ентитета користи се термин *кластеровање својстава.*

4.4. Појам обрасца приступа подацима

Постојање функционалних зависности између типова ентитета представља полазну претпоставку у предложеном моделу за потребе разматрања могућности кластеровања њихових својстава у циљу ефикаснијег приступа вредностима тих својстава у конкретним ентитетима. Међутим, трансакције типично приступају вредностима само релативно малог подскупа свих могућих (транзитивно) функционално зависних својстава типова ентитета. Да би се постигао максимално ефикасан приступ вредностима својстава типова ентитетима, у кластеровању је потребно разматрати само она (транзитивно) функционално зависна својства којима се често заједно приступа. Да би се поједноставио третман проблема и омогућило формално моделовање поступка кластеровања функционално зависних својстава типова ентитета, уводи се појам *обрасца приступа подацима* (енгл. *data retrieval pattern*), као модел дохватања података у смислу навигације преко функционалних својстава. Модел обрасца приступа подацима осмишљен је да буде довољно општи тако да буде примењив како на објектно оријентисане концептуалне моделе, тако и на релационе моделе података.

Табела 2:
Шема скраћеног именовања типова ентитета и својстава у концептуалном моделу аукцијске апликације

A	→	Auction	n	→	name
U	→	User	t	→	time
M	→	Manager	o	→	opened
P	→	Product	e	→	finished
C	→	Category	i	→	item
B	→	Bid	p	→	parent
T	→	Time	c	→	category
S	→	String	a	→	amount
N	→	Number	h	→	inherited
			m	→	manager
			s	→	seller
			b	→	bidder или bids
			ac	→	auction
			w	→	winner

Образац приступа подацима, по својој структури, представља повезано усмерено стабло (енгл. *directed connected tree*), као што је приказано на слици 2. Чворови стабла представљају типове ентитета, док усмерене гране одговарају својствима типова ентитета којима се приступало у току извршавања неког дела програма. Сваки образац има један корени чвор. Визуелно, корени чвор се од осталих чворова на приказаним сликама разликује бојењем (у сиво). За свако својство, представљено усмереном граном, каже се да пресликава тип ентитета на почетку гране на тип ентитета на другом крају гране (у смеру стрелице на приказу). Из особине транзитивности функционалне зависности проистиче и закључак да су својства свих типова ентитети у стаблу обрасца функционално зависна од идентификатора типа ентитета у кореном чвору. Тип ентитета се визуелно приказује унутар самог чвора. Примера ради, корени чвор у стаблу на слици 2 представља тип ентитета *A*. Слично, усмереном граном *s* моделује се читање својства *s* типа ентитета A и приступ типу ентитета *U* на другом крају гране.

Ради прегледнијег приказа, уведени су скраћени једнословни идентификатори за типове ентитета и својства у радном концептуалном моделу на слици 1. Шема скраћеног именовања елемената модела дата је у табели 2.

4.5. Откривање образаца приступа подацима

У решењу које је описано у овој тези, обрасци приступа подацима откривају се анализом извршавања операција приступа подацима у продукционом окружењу или окружењу за тестирање са реалним или бар приближно реалним радним оптерећењем (нпр. пилот имплементираног решења). Операције приступа подацима анализирају се на нивоу и у оквиру логичких јединица обраде под називом *конверзације*.

Конверзација представља секвенцу операција приступа подацима која се одвија у временском интервалу ограниченог трајања, извршава се независно од осталих конкурентних конверзација, има дефинисану семантику и сврху. На пример, то може бити или једна (сложена) трансакција или секвенца (малих) трансакција иницираних у кратком временском интервалу у интеракцији корисника са системом у склопу одређеног случаја коришћења.

Процес анализе образаца одвија се у два корака: а) снимањем извршених конверзација и б) анализом снимљених конверзација. Иако ова два корака теоретски могу да се одвијају у реалном времену, тако што се операције у конверзацијама анализирају одмах по пристизању (енгл. *online*), препорука је, а некада и једино решење у пракси, да се анализа конверзације ради независно од функционисања продукционог система (енгл. *offline*), при чему је минимално неопходно обезбедити снимање извршених конверзација у *дневник конверзација* (енгл. *conversation log*). Другим (*offline*) приступом се остварује минимална и најлабавија могућа спрега (енгл. *loose coupling*) продукционог система са софтверском компонентом за детекцију образаца приступа подацима. Алат који обезбеђује и снимање и анализу конверзација назива се *профајлер* (енгл. *profiler*).

Профилизација извршавања операција приступа подацима може да се врши како на објектном, тако и на нивоу релационог модела података, при чему анализа образаца на основу трага извршавања на објектном нивоу има смисла једино уколико између ОО и релационог модела податка важи усвојено објектно-релационо пресликавање из поглавља 1.4.1. Међутим, уколико се концептуална декомпозиција ОО модела значајно разликује од концептуалне декомпозиције релационог модела (који може бити значајно денормализован), анализа приступа подацима се једино може ваљано спровести на релационом нивоу. Подсећања ради, суштина методе оптимизације приступа подацима, која се предлаже у овој докторској дисертацији, јесте у прилагођавању и циљаној денормализацији релационог модела података у циљу поправљања перформанси приступа бази података како би се такве оптимизације учиниле транспарентним и спровеле без нарушавања „нормализованости“ и ваљане семантичке декомпозиције проблема на нивоу ОО концептуалног модела.

4.5.1. Профајлер операција приступа подацима

У склопу овог истраживања имплементиран је један наменски профајлер за радно окружење Солоист (енгл. *SOLoist* [60]) за развој објектно оријентисаних информационих система помоћу извршивих *UML* модела. Профајлер се састоји од две лабаво спрегнуте компоненте: а) снимача конверзација (енгл. *recorder*) и б) анализатора конверзација (енгл. *analyzer*). Лабава спрега између ове две компоненте остварује се осмишљеним наменским форматом (енгл. *custom format*)

за енкодовање конверзација (видети листинг 4). Формат података за енкодовање конверзација подсећа на асемблерске инструкције. Спецификација формата дата је виду бесконтекстне граматике у EBNF нотацији (листинг 3).

Свака конверзација има заглавље и зачеље. Заглавље и зачеље садрже јединствен идентификатор конверзације, као и временски тренутак кад је конверзација започета и завршена, респективно. Подржано је пет типова операција представљених следећим мнемоницима:

LDEN – операција за проналажење и учитавање ентитета на основу задатог идентификатора;

CREN – операција стварања новог ентитета са задатим идентификатором;

RMEN – операција брисања ентитета на основу задатог идентификатора;

PPRD – операција читања вредности својства ентитета на основу идентификатора ентитета и назива својства; уколико својство представља асоцијациони крај, наводи се листа учитаних везаних ентитета; уколико је својство функционално, наводи се специјална ознака мултипликативности “[1]“;

PPWR – операција ажурирања вредности својства ентитета на основу идентификатора својства и назива својства; уколико својство представља асоцијациони крај, наводи се листа везаних ентитета; уколико је својство функционално, наводи се специјална ознака мултипликативности “[1]“.

Дати формат се поред императивног кода може применити и за енкодовање декларативних OQL или SQL упита. Упити теоретски могу да се моделују навигацијом и преведу на дати формат за енкодовање конверзација. У наставку се даје нацрт једног могућег начина трансформације упита на дати формат дневника конверзација:

- имена табела трансформишу се у имена ентитета,
- појављивање колоне у пројекцији или провери услова трансформише се у операцију читања или уписа вредности својства у зависности од типа упита и места на којем се колона појављује,

  оператор спајања трансформише се у навигацију.

Листинг 3:
ФОРМАТ ЗА ЕНКОДОВАЊЕ КОНВЕРЗАЦИЈА ОПИСАН БЕСКОНТЕКСТНОМ ГРАМАТИКОМ У EBNF НОТАЦИЈИ

```
Conversation ::= Header { Operation } Trailer
Header ::= “CBEG” UniqueConversationId “,” Timestamp
Trailer ::= “CEND” UniqueConversationId “,” Timestamp
EntityId ::= TypeName “(” Identifier “)”
Operation ::= LoadEntity | RemoveEntity | ReadProperty | WriteProperty |
              CreateEntity
LoadEntity ::= “LDEN” EntityId
CreateEntity ::= “CREN” EntityId
RemoveEntity ::= “RMEN” EntityId
ReadProperty ::= “PPRD” EntityId “.” PropertyName [ “[1]” ]
                     [ TypeName “:{” { EntityId } “}” ]
WriteProperty ::= “PPWR” EntityId “.” PropertyName [ “[1]” ]
                     [ TypeName “:{” { EntityId } “}” ]
```

Проблем трансформације упита сам по себи захтева посебну студију. Наведени могући начин трансформације упита примерен је за једноставније и (пожељно) јако селективне упите. Важна претпоставка и ограничење примене предложене трансформације јесте та да у релационом моделу не постоје редундантне приступне структуре (осим кластерованих индекса над примарним кључевима) које би оптимизатор упита могао да укључи у план извршавања и тиме добије другачији план извршавања од оног који се моделује предложеним моделом навигације.

4.5.2. Анализатор конверзација

Без обзира на то да ли се радно оптерећење састоји од декларативних SQL упита или операција императивног кода, анализатор конверзација зависи искључиво од описаног формата конверзацијских дневника са типовима ентитета и њиховим својствима. Анализатор интерпретира операције у конверзацијама и поступно изграђује (стабла) образаца приступа подацима. У току анализе једне конверзације могуће је да настане више образаца приступа подацима. Сви обрасци који се откривају у току анализе текуће конверзације називају се *активни обрасци*. Обрасци приступа подацима формирају се интерпретацијом операција у конверзацији према следећем алгоритму.

Листинг 4:
Један траг извршавања програма за преглед детаља активне аукције (случај коришћења UC2).

```
CBEG   5CC731ACC94770E70884DAE2C4B29824, 1460372412239
LDEN   Auction(100)
PPRD   Auction(100).item[1] Product:{Product(1)}
// display auction details
PPRD   Product(1).name[1]
PPRD   Auction(100).opened[1]
PPRD   Auction(100).finished[1]
PPRD   Auction(100).seller[1] User:{User(1)}
PPRD   User(1).name[1]
PPRD   Auction(100).manager[1] User:{Manager(2)}
PPRD   Manager(2)._super[1] User{User(2)}
PPRD   User(2).name[1]
// display categories
PPRD   Product(1).category[1] Category:{Category(1)}
PPRD   Category(1).name[1]
PPRD   Category(1).parent[1] Category:{Category(2)}
PPRD   Category(2).name[1]
PPRD   Category(2).parent[1] Category:{Category(3)}
PPRD   Category(3).name[1]
// list all bids
PPRD   Auction(100).bids Bid:{Bid(1), Bid(2)}
PPRD   Bid(1).time[1]
PPRD   Bid(1).bidder[1] User:{User(3)}
PPRD   User(3).name[1]
PPRD   Bid(1).amount[1]
PPRD   Bid(1).time[1]
// --
PPRD   Bid(2).time[1]
PPRD   Bid(2).bidder[1] User:{User(4)}
PPRD   User(4).name[1]
PPRD   Bid(2).amount[1]
PPRD   Bid(2).time[1]
// display winner bid
PPRD   Auction(100).winner[1] {Bid(2)}
PPRD   Bid(2).bidder[1] User:{User(4)}
PPRD   User(4).name[1]
PPRD   Bid(2).amount[1]
PPRD   Bid(600).time[1]
CEND   5CC731ACC94770E70884DAE2C4B29824, 1460372412313
```

Алгоритам изградње обрасца приступа подацима интерпретацијом операција у дневнику конверзација:

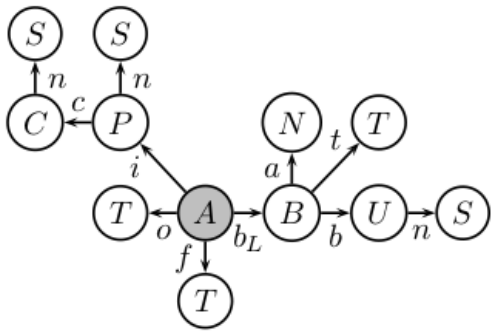
- Операција LDEN интерпретира се додавањем кореног чвора и стварањем новог активног стабла обрасца
- Операција PPRD резултује претрагом ентитета у свим активним стаблима.

- Ако је својство функционално, ствара се нови ентитет резултата читања својства, и везује се за пронађени ентитет у једном од активних стабала додавањем гране својства;
- Ако је својство мултифункционално, сваки од учитаних везаних ентитета третира се као да је учитан акцијом LDEN.

- Операција PPWR не утиче на активна стабла, већ увећава статистику броја ажурирања својства.

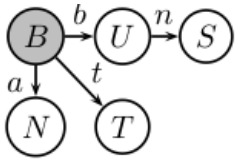
4.6. Примери образаца приступа подацима у радном примеру

На пример, образац приступа подацима *P1* на слици 2 добија се анализом упита *Q1* од стране профајлера на начин описан у поглављу 4.5.1. Интересантно је приметити грану b_L која представља функционално својство које не постоји изворно у концептуалном моделу. Међутим, под одређеним околностима могуће је извођење функционалних својстава из мултифункционалних, као што је, на пример, својство *bids* у типу ентитета *Auction*, будући да профајлер може да буде довољно софистициран да открије („научи“) непроменљиво правило по ком се увек дохвата последња понуда у аукцији. Таква изведена функција представља се такође граном у стаблу обрасца на слици 2.



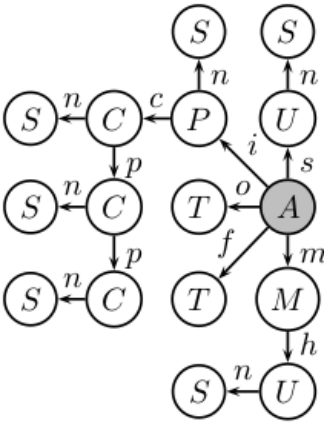
Слика 2: Образац приступа подацима P1 добијен профилизацијом извршавања случаја коришћења UC1

Анализом конверзације приказане у листингу 4, која је добијена снимањем извршавања процедуре UC2 (листинг 2), добијају се три обрасца приступа подацима приказана редом на сликама 3, 4 и 5.



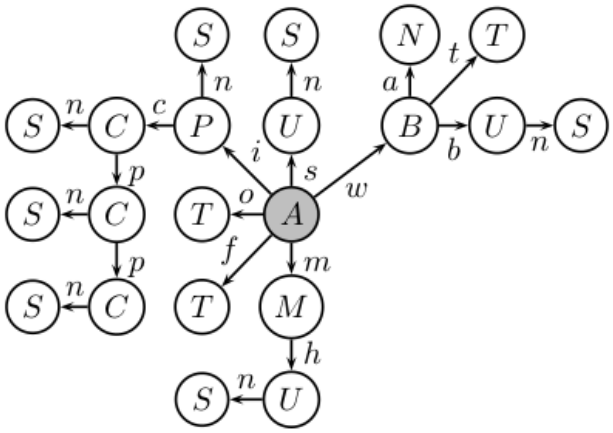
Слика 3: Образац приступа подацима P2 добијен профилизацијом извршавања случаја коришћења UC2

Образац приступа подацима *P2* приказан на слици 3 садржи сва својства о свакој појединачној понуди на задатој аукцији која су захтевана у току извршавања процедуре *UC2*. Сваки ентитет понуде, којем се приступало према описаном обрасцу, добијен је читањем својства *bids* задате аукције.



Слика 4: Образац приступа подацима активне аукције, P3, добијен профилизацијом извршавања случаја коришћења UC2

Образац *P3* приказан на слици 4 садржи сва својства и типове ентитета којима се приступало у току извршавања процедуре *UC2* за потребе приказа детаља било које активне аукције. Пошто победничка понуда не постоји у активним аукцијама, одговарајући условни блок за приказ детаља победничке аукције у процедури *UC2* није извршен.



Слика 5: Образац приступа подацима завршене аукције, P4, добијен профилизацијом извршавања случаја коришћења UC2

Образац *P4* приказан на слици 5 садржи сва својства и типове ентитета којима се приступало у току извршавања процедуре *UC2* за потребе приказа детаља било које завршене аукције. Пошто завршене аукције имају победничке понуде, образац *P4* садржи и подстабло са детаљима победничке понуде којима се приступа у условном блоку процедуре *UC2*.

4.7. Пример денормализације релационог модела

Добијени обрасци одсликавају навигациону природу приступа подацима у датим фрагментима имплементације посматране апликације. Нека се посматра следећи навигациони израз у објектном коду у листингу 2, `a.winner().bidder().name()`, којим се дохвата име корисника који је победио на аукцији `a`. Због концизности и прилагођавања већ усвојеној конвенцији скраћеног именовања (табела 2), дати израз се представља компактнијом нотацијом *a.w.b.n*.

Усвојеном стратегијом објектно-релационог мапирања, која мапира нередундантни концептуални објектни модел директно на (нормализовани) релациони модел података, добија се шема релационог модела података приказана у листингу 5. Извршавање посматраног објектног израза *a.w.b.n* захтева приступ табелама *A*, *B*, *U* у виду три независна појединачна захтева упућена серверу базе података, при чему у сва три случаја програм захтева само један ред из сваке од датих табела, а са диска се читају три блока. Сваки пут када програм врши навигацију по наведеном обрасцу да би приступио имену победника аукције,

мора да се приступи бази података на описани начин. Ако се оваква навигација ради довољно често, поставља се питање: да ли је могуће бази података приступити знатно ефикасније, а задржати идентичан објектни навигациони код? Одговор је да, уколико се подаци у релационом моделу организују на начин који је описан у наставку.

Листинг 5:
Нормализована шема релационог модела апликације за аукцијску продају

A (id, o, f, p, i:P.id, s:U.id, w:B.id, v:M.id)
P (id, n, c:C.id)
C (id, n, p:C.id, c:M.id)
B (id, t, a, b, a:A.id, b:U.id)
U (id, n)
M (id, s, l)

Нека наведене (нормализоване) табеле *A*, *B*, *U*, којима се приступа извршавањем посматраног навигационог израза *a.w.b.n*, имају шеме које су приказане у листингу 5. Пошто се колона *w:B.id* у шеми релације *A* и колона *b:U.id* у шеми релације *B* моделују као функционална својства типова ентитета *A* и *B*, респективно, може се формирати следећа денормализована шема релације груписањем свих транзитивно функционално зависних својстава релевантних за дату навигацију (из шеме денормализоване релације су искључени сви атрибути (својства) који нису релевантни за дати пример навигације):

$$A^d\,(id, w{:}B.id, w{:}B.id{:}B.b, w{:}B.id{:}B.b{:}U.id, w{:}B.id{:}B.b{:}U.n),$$

Уведени редундантни атрибути у шему релације A^d именовани су на начин који одсликава „размотану“ навигацију по којој се до њих долази у полазном нормализованом моделу. Симболи тачка (.) и двотачка (:) који се појављују у именима редундантних атрибута у шеми релације A^d немају специјалну семантику, осим што се користе за потребу концизнијег (непосредног) именовања датих атрибута из којих се оптимизована навигација уочава директно. Тачка се ставља између својства и типа ентитета који садржи то својство у концептуалном (и релационом) моделу, док се двотачка ставља између својства и типа тог својства.

Уколико се након описане денормализације поново изврши посматрани навигациони израз *a.w.b.n*, све захтеване вредности дуж датог ланца навигације могу да се дохвате из једног реда табеле са шемом A^d и тиме ова навигација учини знатно ефикаснијом без потребе за изменама у оригиналном (мање ефикасном) објектном коду. Овакво ефикасно мапирање треба да спроведе слој софтвера за објектно-релационо мапирање, што представља један од примера транспарентног објектно-релационог мапирања.

Након овог једноставног уводног примера денормализације релационе шеме посматрају се ефекти увођења редундансе у нормализовани релациони модел аукцијске апликације приказан у листингу 5 у циљу оптимизовања приступа подацима за случајеве коришћења у табели 4.

Листинг 6:
Мануелно изабрана денормализација релационе шеме аукцијске базе података

A^* *A.s, A.s:U.n, A.o, A.f, A.i, A.i:P.n, A.i:P.c,*
A.m, A.m:M.h, A.m:M.h:U.n,
$A.b_L$, $A.b_L$*:B.a,* $A.b_L$*:B.t, ,* $A.b_L$*:B.b, ,* $A.b_L$*:B.b:U.n,*
A.w, A.w:B.a, A.w:B.t, A.w:B.b, A.w:B.b:U.n

B^* *B.t, B.a, B.b, B.b:U.n*

C^* *C.n, C.p, C.p:C.n, C.p:C.p, C.p:C.p:C.n*

Ефекти примене денормализације посматрају се на примеру денормализоване релационе шеме $\{A^*, B^*, U^*\}$, приказане у листингу 6, предложене од стране неког аналитичара након увида у доступне обрасце приступа подацима *{P1, P2, P3, P4}* и њихове бројеве појављивања, као и прикупљене статистике ажурирања својстава типова ентитета у датим обрасцима. Све три денормализоване релације могу у потпуности да одговоре на захтеве операција приступа подацима у наведеним случајевима коришћења у табели 1. Постојање ових релација у релационом моделу, међутим, не искључује нужно постојање оригиналних релација (одлука о денормализацији оригиналне шеме релације или стварању нове, као материјализованог погледа, јесте ствар имплементације на нивоу релационог модела и она нема утицаја на даљу анализу нити се анализа везује за конкретну имплементацију на релационом нивоу – фокус се ставља искључиво на кластеровање функционално зависних својстава типова ентитета).

За имена колона у денормализованим шемама користи се описана конвенција именовања. На пример, у денормализованој шеми релације A^* постоји колона „*A.w:B.b:U.n*“ која редундантно чува име победника аукције. Када год се учита ред из табеле A^*, име победника на аукцији постаје доступно без потребе за додатним приступима бази података.

4.8. Процена добити и цене денормализације

За потребе квантитативне процене позитивних и негативних ефеката примењене денормализације, најпре се прикупљају статистике из модела података (просечне вредности) које су неопходне за даљу анализу, као што су: а) 2.000 понуда по кориснику, б) 10.000 аукција по менаџеру, в) 10 понуда по аукцији, г) 10 поткатегорија по категорији, д) 500 аукција по продавцу.

У табели 3 квантитативно је приказан утицај примењене денормализације на сваки случај коришћења посебно. Колоне r_N и w_N садрже бројеве редова које апликација учитава и ажурира, респективно, у нормализованом релационом моделу при једном појављивању сваког шаблона. Слично, колоне r_D и w_D садрже бројеве редова које апликација чита и пише, респективно, у датом денормализованом релационом моделу при једном појављивању сваког шаблона. Колоне R_N, W_N,R_D, W_D садрже укупне бројеве редова добијене множењем јединичних вредности бројева редова у колонама r_N, w_N, r_D, w_D и бројева појављивања случајева коришћења.

На основу резултата датих у табели 3 може се приметити да је за идентично радно оптерећење у случају нормализованог релационог модела података потребно прочитати укупно 82.100 редова и ажурирати укупно 4.300 редова, док је у случају предложеног денормализованог релационог модела података потребно прочитати укупно 38.400 редова и ажурирати укупно 8.300 редова у табелама. Концизније се дати резултати могу представити у нотацији са уређеним паровима вредности (rb, wp), где rb представља добит при читању (нпр. изражену бројем редова), а wp представља увећање цене при упису (нпр. изражено бројем редова). Коришћењем усвојене нотације, за нормализовани модел добија се процена приступа подацима $n = (82.100, 4.300)$, док за денормализовани релациони модел она износи $d = (38.400, 8.300)$. Као коначна процена учинка предложене

денормализације шеме релационог модела аукцијске базе података, добија се вредност $n - d = (43.700, -4.000)$ на основу које се закључује да се предложеном денормализацијом елиминише читање 43.700 редова на рачун повећања укупног броја редова за ажурирање од 4.000 (изражен као негативна добит при упису).

ТАБЕЛА 3:
ПРОЦЕНА ДОБИТИ И ЦЕНЕ ПРИ МАНУЕЛНОЈ ДЕНОРМАЛИЗАЦИЈИ ШЕМЕ РЕЛАЦИОНОГ МОДЕЛА АУКЦИЈСКЕ АПЛИКАЦИЈЕ

Случајеви коришћења	**Фрекв. случ. коришћења**	r_N	w_N	r_D	w_D	R_N	W_N	R_D	W_N
UC1	5.000	5	0	2	0	25.000	0	10.000	0
UC2	500	30	0	12	0	15.000	0	6.000	0
	1.500	28	0	12	0	42.000	0	18.000	0
UC3	4.000	0	1	1	2	0	4.000	4.000	8.000
UC4	100	0	2	3	2	0	200	300	200
UC5	100	1	1	1	1	100	100	100	100
Укупно						**82.100**	**4.300**	**38.400**	**8.300**

Међутим, у овом прорачуну није узета у обзир и цена потенцијалних ажурирања вредности својстава која постоје у моделу, али нису експлицитно ажурирана у доступним случајевима коришћења. Нека се посматра својство $U.n$ које представља име корисника портала. Корисници имају различите улоге у асоцијацијама у датом концептуалном моделу (слика 1), као што су продавац, понуђач или менаџер. Ако за својство $U.n$ постоје редундантне копије у посматраној релационој шеми, и ако се вредност тог својства промени само једном у свакој од поменутих улога (за три конкретна корисника), за одржавање свих копија имена корисника потребно је додатно ажурирати 12.700 редова у денормализованој бази података, што даје измењену процену добити предложене денормализације релационог модела од $(43.700, -16.700)$.

До процене додатних пенала при упису од 12.700 редова дошло се на следећи начин (имајући у виду да се имена корисника редундантно чувају у редовима аукција и понуда): приликом измене имена менаџера потребно је у просеку ажурирати 10.000 аукција (због задате просечне кардиналности од 10.000 аукција по менаџеру); приликом измене имена понуђача потребно је ажурирати 2.000 понуда (због задате просечне кардиналности од 2.000 понуда по понуђачу), али и

додатних 200 аукција, ако се у разматрање узме и могућност је 1/10 од 2.000 понуда по понуђачу последња на аукцијама (аукције у просеку имају 10 понуда, од којих је 1 последња); ако се измени име продавца, потребно је ажурирати копије његовог имена у 500 аукција у просеку (због задате просечне кардиналности од 500 аукција по продавцу).

За потребу тумачења датих резултата усваја се претпоставка да је цена (упита) читања реда из табеле у бази података упоредива са ценом његовог ажурирања. Иако процена учинка предложене денормализације у претходном примеру предвиђа побољшање ефикасности приступа бази података, тренутно није могуће одговорити на питање да ли се ради о најбољем могућем резултату који се може постићи денормализацијом.

Практични експерименти спроведени у оквиру овог истраживања показали су да процена добити денормализације, заснована искључиво на бројању редова, није увек тачна и поуздана метрика, јер тако упрошћен модел не узима у обзир скривену сложеност, софистицираност и оптимизације релационих база података. Због тога се јавља потреба за софистицираним моделом процене цене и добити ефеката увођења редундансе у релациони модел података.

Са друге стране, предложена денормализација релационе шеме у радном примеру, $\{A^*, B^*, U^*\}$, добијена је као резултат чисте професионалне интуиције аналитичара и искуства у моделовању и имплементацији таквих и сличних ОО софтверских система. Међутим, за примену посматраног типа денормализације на реалне и сложене индустријске релационе моделе података, неопходно је да постоји аутоматизован приступ одлучивања о увођењу редундансе у релациони модел података који би користио финије и прецизније хеуристике за квантитативну процену њеног учинка. У наредној глави биће показано да проблем избора денормализоване шеме са најбољим могућим односом цене и добити, узимањем у обзир динамичких аспеката приступа подацима (статистика радног оптерећења, кардиналности у моделу података), представља комбинаторни оптимизациони проблем. Глава 5 посвећена је формалном моделовању денормализације релационог модела података засноване на редунданси података.

5. Формални модел

У овој глави формално се дефинишу концепти обрасца приступа подацима и модел оптимизације приступа подацима заснован на денормализацији, који су кључни за формализацију предложене методе оптимизације приступа подацима у релационом моделу. Детаљно је описан модел за процену утицаја денормализације шеме релационог модела на поправљање перформанси приступа подацима, дата је дефиниција комбинаторног оптимизационог проблема за проналажење оптимално денормализоване шеме релационог модела на основу задатих карактеристика радног оптерећења и доказано да предложени оптимизациони проблем припада групи NP-комплетних проблема. У циљу његовог ефикасног решавања, дат је детаљан опис трансформације предложеног оптимизационог проблема у проблем бинарног линеарног програмирања за чије решавање постоје ефикасне хеуристике.

5.1. Дефиниције

Нека E представља скуп типова ентитета $\{E_1, E_2, \dots, E_k\}$ од којих сваки представља скуп ентитета проширен специјалним елементом $null$. Даље, нека F представља скуп функција $\{f_1, f_2, \dots, f_n\}$, где је свака функција дефинисана као пресликавање $f_k: E_i \to E_j,\ E_i, E_j \in E$. Да би се омогућило прављење композиција функција, уводи се правило $f(null) = null$ за свако $f \in F$.

Функцијама се моделују својства типова ентитета. Свакој функцији $f: X \to Y$ придружују се следећи атрибути:

- w – **број ажурирања / уписа** (енгл. *write frequency*) која се односи на укупан број операција измена функције према профилу извршавања програма (нпр. укупан број операција уписа својства f било ког ентитета x типа X),

- d – **реверзна кардиналност** (енгл. *reverse cardinality*), која представља просечан број ентитета из домена X посматране функције f који се пресликавају на идентичан ентитет y, $y \neq null$, из њеног кодомена Y,
- μ – **цена приступа податку** (енгл. *data retrieval cost*), представља цену дохватања ентитета y, $y \neq null$, из кодомена функције f за задати ентитет $x \in X$ дефинисану формулом $\mu = \lambda + \tau$, где λ представља цену лоцирања ентитета y у кодомену функције f, док τ представља цену преноса лоцираног ентитета на локацију за обраду (нпр. оперативну меморију).

И вишевредносна својства типова ентитета (енгл. *multivalued properties*) по потреби могу да се делимично укључе у овај модел помоћу функција на следећи начин. Анализом операција приступа подацима у конверзацијама профајлер може да закључи да се, са фреквенцијом изнад унапред дефинисаног прага, чита само одређени (пожељно мали) подскуп k вредности одређеног вишевредносног својства f (са кардиналношћу n) према неком непроменљивом критеријуму у програму (нпр. неколико првих/последњих вредности, или најскорије додата вредност). Ако се својство f посматра као мултифункција, из ње је могуће извести следећих k функција: $f[i_1], f[i_2], \dots, f[i_k]$, док се преосталих $n - k$ вредности не третирају на посебан начин и не укључују у овај модел. На пример, на описани начин изведена је функционална зависност b_L, приказана на слици 2, од мултифункционалне релације $bids$, која, подсећања ради, представља асоцијациони крај класе *Auction* преко ког се могу добити све понуде (типа *Bid*) на датој аукцији.

5.2. Композиција функција

За функције f_p и f_q каже се да су уланчане, записано као $f_p \curvearrowright f_q$, ако домен функције f_q представља кодомен функције f_p. Уланчаним функцијама моделује се навигација преко функционалних својстава типова ентитета. За ланац функција $f_{i1} \curvearrowright f_{i2} \curvearrowright \cdots \curvearrowright f_{ik}$ може се дефинисати композиција функција $f_x = f_{i1} \circ f_{i2} \circ \dots \circ f_{ik}$. Нека E_x представља домен функције f_{i1} и нека E_y представља кодомен функције f_{ik}. За посматрани ентитет $e_x \in E_x$ може се одредити његов функционално зависни ентитет $e_y \in E_y$ на крају ланца као вредност израза $e_y = f_{ik}\left(\dots f_{i2}\left(f_{i1}(e_x)\right)\right) = (f_{i1} \circ f_{i2} \circ \dots \circ f_{ik})(e_x)$. Редослед функција у композицији је одабран тако да буде

симетричан са редоследом њиховог појављивања у функцијском ланцу, при чему је усвојена семантика слагања функција приказана у претходном изразу. Сваки позив функције у посматраном изразу, ако се посматра као навигација у нормализованом релационом моделу, увећава цену рачунања вредности дате сложене функције, односно цену приступа ентитету e_y. Са друге стране, уколико би композиција посматраних функција, f_x, представљала додатно (редундантно) својство у типу ентитета E_x, помоћу ње би се посматрани ентитет e_x могао директно пресликати на ентитет e_y, без вишеструке навигације Истом ентитету би у том случају било могуће приступити алтернативном путањом помоћу само једног „позива“ функције, $e_y = f_x(e_x)$, чиме се уштеди $k-1$ позива функција, односно линеарна навигација преко $k-1$ функционалних веза у полазном нормализованом моделу. Другим речима, функција f_x представља „пречицу“ од ентитета e_x до његовог функционално зависног ентитета e_y, и назива се *оптимизацијом* приступа ентитету e_y (од ентитета e_x).

5.3. Обрасци приступа подацима

Образац приступа подацима дефинише се као усмерено стабло за кореним чвором, представљено уређеном торком $(V, U, M_V, M_U, r, \omega, \theta)$, где елементи торке редом представљају:

- V – скуп чворова,
- U – скуп усмерених грана,
- M_V – скуп маркирања чворова, односно функцију (не нужно инјекцију) $V \rightarrow E$, где је E скуп свих типова ентитета,
- M_U – скуп маркирања грана, односно функцију (не нужно инјекцију) $U \rightarrow F$, где F представља скуп свих функција,
- $r \in V -$ корени чвор,
- $\omega \in \mathbb{N}$ – број појављивања обрасца у профилу извршавања (односно његову фреквенцију),
- $\theta \in \mathbb{R}, \theta \in [0,1]$ – цену обрасца (нормализовану на 1) која одсликава релативну сложеност посматраног обрасца у односу на све откривене обрасце у профилу извршавања; добија се проценом сложености навигације у обрасцу анализом плана извршавања обрасца од стране оптимизатора

упита система за управљање базама података; користи се као показатељ утицаја (тежински фактор значаја) евентуалне оптимизације датог обрасца на поправљање перформанси радног оптерећења као целине.

Нека P представља скуп свих образаца приступа подацима. Свака путања суседних усмерених грана у стаблу обрасца одговара једном функционалном ланцу на основу ког се може добити једна оптимизација (композицијом функција). На пример, на основу секвенце уланчаних функција b_L, b, n, приказаних на слици 2, добија се оптимизација композицијом датих функција, $lbn = b_L \circ b \circ n$ која пресликава ентитет аукције на име последњег понуђача (ентитет типа текста). Унија свих композиција функција изведених из свих образаца у скупу P даје скуп свих могућих оптимизација, O. Подскуп скупа свих оптимизација, $C \subseteq O$, представља парцијалну селекцију оптимизација која се назива *конфигурација*. Број свих могућих конфигурација над скупом O износи 2^N, где $N = |O|$ представља кардиналност скупа O (укупан број свих подскупова скупа O).

Концепт конфигурације може се еквивалентно дефинисати увођењем концепта конфигурационе променљиве. Конфигурациона променљива се дефинише као N-димензиони вектор бинарних променљивих $c = [o_1, o_2, \dots, o_N]$, $N = |O|$, од којих свака одговара тачно једној оптимизацији из скупа O и дефинише да ли је посматрана оптимизација укључена у конфигурацију ($o_i = 1$) или не ($o_i = 0$). Пошто свака од N оптимизационих променљивих може да има једну од две вредности $\{0,1\}$, укупан број свих могућих вредности конфигурационе променљиве износи 2^N, што је еквивалентно броју могућих конфигурација дефинисаних над O.

Нека се посматра већ дефинисана оптимизација $f_x = f_{i1} \circ f_{i2} \circ \dots \circ f_{ik}$. Ако се претпостави да је она укључена у конфигурацију или, другим речима, „примењена“ (на пример, чувањем њених резултата у додатним редундантним пољима ентитета типа E_x), то значи да је њен резултат доступан сваки пут када се приступи неком од ентитета из E_x. Таквом оптимизацијом може да се елиминише $k - 1$ приступа табелама у бази података уколико је потребно пронаћи вредност функције на крају ланца за неки учитан ентитет $e_x \in E_x$, јер је вредност композиције функција већ израчуната и сачувана заједно са e_x.

5.4. Модел добити и цене

Ефекти увођења редундансе података у релационом моделу на описани начин квантификују се разликом између добити остварене при операцијама читања и додатне цене одржавања конзистентности редундантних података.

Нека параметар $\rho \in \mathbb{R}$, $\rho \in [0,1]$ представља удео броја својстава неког типа ентитета који су релевантни (користе се) у току извршавања неког обрасца приступа подацима. На пример, ако се у неком обрасцу читају 2 од 10 доступних колона у реду табеле у релационој бази података, уз претпоставку да све колоне имају исту ширину, добија се вредност $\rho = 0.2$. У релационим базама података (у којима се подаци записују по редовима, енгл. *row-oriented*) процена цене плана извршавања упита практично не зависи од наведеног фактора ρ. Међутим, основна дефиниција формуле за процену цене приступа ентитетима преко функционалне везе, $\mu = \lambda + \tau$, проширује се корективним фактором $1 + L(1 - \rho)$, $L \in \mathbb{R}, L \geq 0$ како би се у анализи цене и добити, која се предлаже у овој дисертацији, адаптивно повећала цена приступа ентитетима са мањим уделом броја релевантних својстава и на тај начин фаворизовало њихово редундантно смештање у моделу података. Коначна проширена формула за цену приступа ентитету преко функционалне везе постаје $\mu = (\lambda + \tau) \cdot \big(1 + L(1 - \rho)\big)$, где L представља емпиријски конфигурациони параметар за контролу утицаја удела релевантних података у процени цене функције и он се у општем случају дефинише на нивоу обрасца приступа подацима. Формално, L представља пресликавање $L: P \times F \to \mathbb{R}$, где P представља скуп свих образаца, а F скуп свих функција.

Добит при читању (енгл. *read benefit*), означена са $rb(p, c)$, даје умањење цене приступа подацима при извршавању обрасца p, захваљујући примењеним оптимизацијама у конфигурацији c. Укупна добит при читању (енгл. *total read benefit*), означена са $rb(c)$, дефинише кумулативну добит при читању у свим обрасцима приступа подацима на које су оптимизације у конфигурацији c имале (позитиван) утицај. Операциона дефиниција укупне добити при читању, формално дефинисане као пресликавање $rb: \{0,1\}^N \to \mathbb{R}$, дата је у једначини (1), где променљива ω_k представља фреквенцију обрасца p_k из скупа свих образаца P.

$$b(c) = \sum_{p_k \in P} rb(p_k, c)\, \omega_k \qquad (1)$$

Пенали при упису, означени са $wp(f_k, c)$, представљају додатну цену одржавања конзистентности редундантних копија вредности функције f_k, уведених оптимизацијама у конфигурацији c, сваки пут када се нека функција f_k промени. Укупни пенали при упису, означени са $wp(c)$, представљају кумулативне пенале при упису за све функције које учествују у оптимизацијама у конфигурацији c. Операциона дефиниција укупних пенала при упису, дефинисана формално као пресликавање $wp: \{0,1\}^N \to \mathbb{R}$, дата је у једначини (2), где променљива w_k означава фреквенцију уписа/измена функције f_k из скупа свих функција F.

$$wp(c) = \sum_{f_k \in F} wp(f_k, c)\, w_k \qquad (2)$$

5.5. Обрасци линеарне навигације

Нека запис $f_{i1} \curvearrowright f_{i2} \curvearrowright \cdots \curvearrowright f_{im}$ представља један функционални ланац изолован из једног једноставног линеарног обрасца навигације (нпр. попут обрасца приказаног на слици 6). Нека је скуп E_{i1} домен функције f_{i1} и нека је E_{im+1} кодомен последње функције f_{im}.

Скуп оптимизација $O_m = \{f_{12}, \dots, f_{1m}, f_{23}, \dots, f_{m-1m}\}$ садржи све могуће оптимизације изведене из задатог линеарног обрасца. Нека конфигурација $C_{1m} = \{f_{12}, f_{13}, \dots, f_{1m}\}$, садржи само оне оптимизације које имају функцију f_{i1} као почетну функцију у композицији функција. Оваквим избором конфигурације C_{1m} моделује се елементарни случај денормализације, у ком утицај примењених оптимизација може да се формално квантификује аналитичким формулама правилне форме (која не зависи од структуре и величине обрасца). Типови ентитета и функције у датом обрасцу одсликавају траг извршавања операција за приступ подацима које су статички дефинисане у имплементацији. Када год се извршава дати део програмског кода и приступа подацима према задатом обрасцу, на апликативном (објектном) нивоу се врши навигација са краја на крај датог функционалног ланца полазећи од неког ентитета $e_x \in E_x$ (којем се свакако прво приступа). Међутим,

уколико се примене посматране оптимизације у конфигурацији C_{1m}, таква навигација не мора директно да се преслика на навигацију на нивоу релационог модела података. Ако се вредности свих вредности функција у посматраном ланцу редундантно сачувају у реду табеле датог ентитета e_x, навигација на објектном нивоу може да се преусмери на читање свих потребних вредности из само једног реда (једне табеле) у релационом моделу и на тај начин се елиминише вишеструки и мање ефикасан случајан приступ блоковима на диску.

Даље, свака функција f_{1k} у конфигурацији C_{1m} дефинисана је као композиција $f_{1k} = f_{i1} \circ f_{i2} \circ \ldots \circ f_{ik}$ која пресликава ентитете из домена E_{i1} прве функције f_{i1} на ентитете у кодомену E_{ik+1} последње функције у композицији, f_{ik}. Уколико се дата функција f_{1k} редундантно чува као додатно својство типа ентитета E_{i1}, сваки пут када се приступи неком од његових ентитета, и вредност редундантне функције f_{1k} постаје одмах доступна програму, без потребе за додатном навигацијом дуж функционалног ланца до ентитета који садржи својство представљено функцијом f_{ik}. Ако се уведе конфигурациона променљива за скуп O_m, $c = [o_{12}, \ldots, o_{1m}, o_{23}, \ldots, o_{m-1m}]$, добит при читању за дати линеарни образац навигације дефинисана је једначином (3), где је $\mu_{ik} = \mu(f_{ik})$, μ_{ik} – цена „израчунавања“ функције f_{ik} (која се елиминише оптимизацијом), ω – број појављивања посматраног линеарног обрасца, а m – његова дужина.

$$rb(c_{1m}) = \omega \sum_{k=2}^{k=m} \mu_{ik} \qquad (3)$$

Вредност конфигурационе променљиве $c_{1m} = [1,1,\ldots,1,0,\ldots,0]$, са $m-1$ водећих јединица, представља разматрану конфигурацију C_{1m}. Свака примењена оптимизација f_{1k} из наведене конфигурације умањује укупну цену приступа ентитету (у нормализованом моделу, без оптимизација) за μ_{ik}.

Са друге стране, свака промена функције у линеарном обрасцу одражава се (негативно) на све оптимизације које садрже дату функцију у композицији, јер је потребно ажурирати сва пресликавања на која та промена утиче чиме она постају застарела. Промене функција у линеарном обрасцу у овој анализи разматрају се као независни догађаји, ради једноставности, чиме се истовремено моделује најгори случај за процену пенала при промени (ажурирању) функција.

$$wp(c_{1m}) = \sum_{k=1}^{k=m} \left(wb(f_{ik}) + ra(f_{ik})\right) w_{ik} \qquad (4)$$

Као што се може видети у једначини (4), процена пенала при ажурирању сваке од функција, f_{ik}, у задатом линеарном обрасцу представљена је збиром пенала два типа режијских операција за одржавање конзистентности редундантних копија података (помножених бројем промена дате функције, w_{ik}) и то:

а) пенала ажурирања уназад, $wb(f_{ik})$ (енгл. *write-backward penalty*), и

б) пенала читања унапред, $ra(f_{ik})$ (енгл. *read-ahead penalty*).

Метрика цене ажурирања уназад (енгл. *write-backward*), $wb(f_{ik})$, процењује цену ажурирања свих копија вредности функције f_{ik} које постоје у ентитетима у скупу E_{i1} (захваљујући примењеним оптимизацијама). Процењени број ентитета у скупу E_{i1} који садрже редундантне вредности функције f_{ik} (представљених оптимизацијама f_{1k}) једнак је производу реверзних кардиналности свих функција $f_{i1}, f_{i2} \dots f_{ik-1}$ које су у посматраним оптимизацијама из конфигурације C_{1m} наведене пре функције f_{ik}, односно $wb(f_{ik}) = \mu_{i1} \prod_{j=1}^{j=k-1} d_{ij}$, где је d_{ij} реверзна кардиналност функције f_{ij}, а μ_{i1} цена приступа једном ентитету у E_{i1}.

Уколико постоје оптимизације у којима посматрана промењена функција f_{ik} није последња у композицији функција, као што су $f_{1k+1}, \dots, f_{1m}$, није довољно ажурирати искључиво редундантне вредности функције f_{ik}, већ и редундантне вредности функција $f_{1k+1}, \dots, f_{1m}$. Будући да се промена функције f_{ik} своди на „превезивање“ ентитета из њеног домена на други ентитет у њеном кодомену, нове вредности функција $f_{1k+1}, \dots, f_{1m}$ добијају се навигацијом преко функционалних веза од новог ентитета унапред. Због тога се те додатне режије називају пенали при читању унапред (енгл. *read-ahead penalty*) и рачунају према формули $ra(f_{pk}) = \sum_{j=k+1}^{j=m} \mu'_{ij}$, што представља просту суму цена приступа појединачним ентитетима на новој стази (или, у општем случају, новом стаблу ентитета). Треба поменути и то да се и читање унапред може оптимизовати редундантним функционалним везама, што је детаљније објашњено у следећем поглављу.

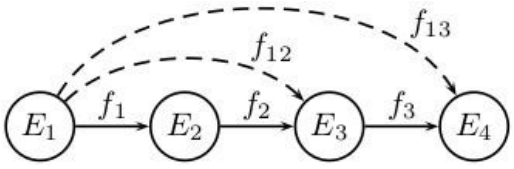
Две поменуте врсте пенала при упису не морају увек заједно да фигуришу у формули за процену укупних пенала при упису. На пример, рачунање пенала при

промени функције f_{i1}, која је прва у композицији, своди се искључиво на пенале због читања унапред, док се рачунање пенала при промени функције f_{im}, која је последња у композицији, своди искључиво на пенале због ажурирања уназад. Након детаљног описа рачунања пенала при променама функција, коначна формула за рачунање укупних пенала при упису, за задату конфигурацију c_{1m}, дефинисана је једначином (5), где m представља дужину посматраног линеарног обрасца.

$$wp(c_{1m}) = \sum_{k=1}^{k=m} \left(\mu_{i1} \prod_{j=1}^{j=k-1} d_{ij} + \sum_{x=k+1}^{x=m} \mu_{ix} \right) w_{ik} \tag{5}$$

5.6. Пример рачунања цене и добити за линеарни образац навигације

Описане једначине модела цене и добити илустроване су на примеру линеарног обрасца $f_1 \rightsquigarrow f_2 \rightsquigarrow f_3$ приказаног на слици 6.



Слика 6: Пример линеарног обрасца приступа подацима

На основу функционалног ланца $f_1 \rightsquigarrow f_2 \rightsquigarrow f_3$ изводи се скуп свих могућих оптимизација $O_x = \{f_{12}, f_{13}, f_{23}\}$, као и њему одговарајућа конфигурациона променљива $c_x = [o_{12}, o_{13}, o_{23}]$. Пошто је образац приступа подацима линеаран, уведене једначине за процену цене и добити илуструју се на примеру конкретне конфигурације $c_{110} = [1,1,0]$. Дата конфигурација визуелизована је испрекиданим гранама на слици 6. Функција f_{23} није укључена у дату конфигурацију ($o_{23} = 0$). Ради добијања једноставнијег облика једначина цене и добити, физичким параметрима модела додељују се следеће вредности: $\tau = 0$, $\lambda = 1$, $\rho = 0.1$ и $L = 0{,}1$.

Конкретне једначине модела цене и добити приказане су у наставку.

$$rb(c_{110}) = 2 \cdot (1 + 0{,}09)\omega = 2{,}18\omega$$

$$wp(c_{110}) = wp(c_{110}, f_1) + wp(c_{110}, f_2) + wp(c_{110}, f_3)$$

$$= 2{,}18w_1 + 1{,}09(1 + d_1)w_2 + 1{,}09 \cdot d_1 d_2 w_3$$

где су парцијални изрази за процену пенала при упису редом дефинисани следећим једначинама:

$$wp(c_{110}, f_1) = w_1(\mu_2 + \mu_3) = 2w_1\lambda(1 + 0{,}09) = 2{,}18w_1$$

$$wp(c_{110}, f_2) = w_2(\mu_2 + \mu_1 d_1) = 1{,}09(1 + d_1)w_2$$

$$wp(c_{110}, f_3) = \mu_1 d_1 d_2 w_3 = 1{,}09 \cdot d_1 d_2 w_3$$

5.6.1. Дискусија добијених резултата

У овој секцији дискутују се добијене конкретне једначине за процену цене и добити приступа ентитетима уколико би се предложене оптимизације примениле на посматрани образац линеарне навигације. Дискусија се базира на конкретном примеру три ентитета, $e_1 \in E_1$, $e_2 \in E_2$, $e_3 \in E_3$, који формирају функционални ланац као на слици 6.

Ажурирање вредности функције $e_1.f_1$ (преусмеравање на нови ентитет $e_2' \in E_2$) захтева читање унапред почевши од ентитета e_2' да би се приступило новим вредностима редундантних функција $e_1.f_{12}$ и $e_1.f_{13}$.

Ажурирање вредности функције $e_2.f_2$ (преусмеравање на нови ентитет $e_3' \in E_3$) утиче на оптимизације f_{12} и f_{13}. Уколико би f_{12} била једина примењена оптимизација, читање унапред не би било потребно у овом случају. Међутим, пошто је и оптимизација f_{13} примењена, неопходно је прочитати унапред нову вредност функције $e_3'.f_3$ како би се ажурирале старе вредности функције $e_1.f_{13}$ у ентитетима скупа E_1 који функционално одређују ентитет e_2. Процењени број ентитета у скупу E_1 који садрже старе вредности редундантних функција $e_1.f_{12}$ и $e_1.f_{13}$ одређен је реверзном кардиналношћу функције f_1.

Ажурирањем вредности функције $e_3.f_3$ (преусмеравање на нови ентитет $e_4' \in E_4$), једначина за процену пенала при упису има само компоненту ажурирања уназад. Производ $d_1 d_2$ одређује процењени број ентитета $e_1^* \in E_1$ (где важи: $f_2(f_1(e_1^*)) = e_3$), који садрже вредности функције $e_1^*.f_{13}$ које треба ажурирати.

5.7. Модел цене и добити базиран на бројању ентитета (редова)

Утицај наведених физичких параметара у једначинама може бити потпуно елиминисан тако што им се доделе следеће вредности: $\tau = 0$, $\lambda = 1$, $\rho = 1$ и $L = 0$. У том случају, добијене једначине у претходном примеру трансформишу се у следеће једноставније форме засноване на бројању ентитета (редова).

$$rb(c_{110}) = 2\omega$$

$$wp(c_{110}) = wp(c, f_1) + wp(c, f_2) + wp(c, f_3)$$

$$wp(c_{110}) = 2w_1 + (1 + d_1)w_2 + d_1 d_2 w_3$$

У складу са тим, опште једначине (3) и (5) трансформишу се на једначине засноване на бројању ентитета. Добит при читању у овом случају изражена је бројем ентитета (у нормализованом моделу) који не морају да се прочитају када год програм приступа подацима према посматраном обрасцу и примењене су оптимизације у конфигурацији $c_{110} = [1,1,0]$.

$$rb(c_{1m}) = (m - 1)\omega \tag{6}$$

Слично томе, једначина за процену пенала при упису процењује број ентитета (у денормализованом моделу) које треба ажурирати када год се нека од функција, која учествује у примењеним оптимизацијама, промени.

$$wp(c_{1\mathrm{m}}) = \sum_{k=1}^{k=m} \left(\prod_{j=1}^{j=k-1} d_{ij} + (m - k) \right) w_{ik} \tag{7}$$

Да би се коначно проценила добит предложених оптимизација у конфигурацији $c_{110} = [1,1,0]$, неопходно је познавати и вредности својстава w и d сваке функције која се појављује у посматраним оптимизацијама, као и број појављивања обрасца ω који се разматра.

На пример, уколико се задају вредности $w = 5$ и $d = 2$ за сваку функцију, као и број појављивања датог обрасца, $\omega = 50$, добија се следећа рачуница

коришћењем једначина (6) и (7) заснованих на бројању ентитета: $rb = 100$ и $wp = 45$. Добијене вредности се интерпретирају на следећи начин.

Уколико се посматрани образац линеарне навигације $f_1 \rightsquigarrow f_2 \rightsquigarrow f_3$ изврши $\omega = 50$ пута, и ако су примењене посматране оптимизације у конфигурацији c_{110}, може се уштедети $rb = 100$ приступа ентитетима (редовима), који су неопходни у нормализованом моделу податка без редундансе. Практично, по свакој појави обрасца уштеди се читање два реда (два обраћања бази података).

Уколико се свака од функција у обрасцу промени $w = 5$ пута (за било које ентитете), потребно је ажурирати (поред основних) и $wp = 45$ додатних ентитета (редова).

5.8. Математички оптимизациони модел

Помоћу изведених једначина модела цене и добити могуће је квантитативно проценити позитивне и негативне ефекте оптимизација базираних на увођењу редундантних података у релациони модел. Међутим, још један од циљева овог истраживања био је развој методе за проналажење конфигурација денормализације релационог модела са најбољим односом цене и добити узимајући у обзир прикупљене статистике и обрасце приступа подацима добијене анализом радног оптерећења.

Проблем проналажења оптималне конфигурације формално се дефинише на следећи начин: пронаћи конфигурацију оптимизација у релационом моделу која даје максималну добит при читању података, при чему пенали при ажурирању не смеју бити већи од добити при читању нити од дефинисане граничне вредности T која представља максималну прихватљиву вредност пенала при упису. Формално, дати оптимизациони проблем се може записати математички на следећи начин:

$$MAX\ rb(c), \qquad c = [o_1, o_2, \dots, o_N]$$

$$s.t.\ \ rb(c) > wp(c),\ wp(c) \leq T,\ T \geq 0.$$

Овај оптимизациони проблем назван је *проблемом равнотеже редундансе* у релационом моделу података (енгл. *Data Redundancy Equilibrium*, *DRE*). За његово решавање неопходно је прво трансформисати постојеће једначине модела цене и добити параметризовањем помоћу оптимизационих променљивих. Таква

трансформација омогућава аналитичко моделовање утицаја оптимизација на добит при читању и додатну цену при упису која се може применити на обрасце приступа подацима произвољне структуре стабла. Оптимизационе променљиве представљају независне променљиве у дефиницији оптимизационог проблема. Најпре ће бити илустроване трансформације на постојеће једначине (3) и (5) изведене за линеарне навигационе обрасце, а затим се даје опис трансформација посматраних једначина цене и добити за општа стабла образаца приступа подацима.

Једначина (3) за процену добити при читању примењена на линеарни образац приказан на слици 6 трансформише се додавањем оптимизационих променљивих на следећу форму:

$$rb(c) = \omega \cdot (\mu_1 E_1^{opt} + \mu_2 E_2^{opt} + \mu_3 E_3^{opt}) \qquad (8)$$

У једначини (8) фигуришу бинарне променљиве E_k^{opt} од којих свака контролише да ли је приступ одговарајућем типу ентитета E_k у посматраном обрасцу приступа подацима оптимизован (елиминисан) или не. Ове променљиве називају се и *ентитетске оптимизационе променљиве* (приметити ознаку *opt* у експоненту чија је улога да се уведе нотациона разлика између ентитетских оптимизационих променљивих и типова ентитета на које се односе). Уколико променљива E_k^{opt} има вредност 1, приступ ентитету типа E_k је оптимизован, што значи да се цена приступа том ентитету у нормализованом моделу елиминише и додаје на страну добити.

Вредности ентитетских бинарних променљивих формирају се (сложеним) логичким изразима у којима учествују основне оптимизационе бинарне променљиве, као и друге ентитетске оптимизационе променљиве. За посматрани образац на слици 6 почетно ограничење јесте $E_1^{opt} \stackrel{\text{def}}{=} 0$ због тога што у предложеном моделу програм мора да учита бар први ентитет у обрасцу навигације (према раније датом образложењу). Приступ ентитету $e_2 \in E_2$ може бити елиминисан уколико се сва његова својства, којима се приступа у посматраном обрасцу, редундантно чувају у ентитету $e_1 \in E_1$. Другим речима, у случају приступа подацима према посматраном обрасцу, приступ реду ентитета e_2 се елиминише ако је примењена оптимизација f_{12}, што се моделује активном

вредношћу њене придружене оптимизационе променљиве, $o_{12} = 1$. На основу ове рационализације формира се логички израз $E_2^{opt} = o_{12}$ за рачунање вредности ентитетске оптимизационе променљиве E_2^{opt}. Надаље, приступ ентитету $e_3 \in E_3$ може се елиминисати уколико је: а) примењена оптимизација f_{13} и биће расположива након приступа ентитету e_1, или је б) примењена оптимизација f_{23} и биће расположива програму након приступа ентитету e_2. Описани услов за оптимизацију приступа ентитету e_2 у датом линеарном обрасцу представља се сложенијим логичким изразом: $E_3^{opt} = o_{13} \vee (o_{23} \wedge \neg E_2^{opt})$. Другим речима, оптимизација f_{23} има смисла само ако ће програм сигурно приступити оригиналном ентитету e_2 (гледано из перспективе полазног нормализованог модела). То је могуће остварити једино уколико се оптимизација f_{12} не примени, тако да програм мора сигурно да приступи ентитету e_2, што се формулише изразом $\neg E_2^{opt}$. Ако се ентитетска оптимизациона променљива E_2^{opt} замени својим логичким изразом, добија се логички израз који зависи директно од основних оптимизационих променљивих, $E_3^{opt} = o_{13} \vee (o_{23} \wedge \neg o_{12})$.

Уколико приступ ентитету e_2 није оптимизован, оптимизација приступа вредности његовог својства $e_2.f_2$ нема ефекта јер се свакако не елиминише приступ ентитету e_2. У општем случају, приступ ентитету, у оквиру неког обрасца, може се елиминисати само ако су вредности свих својстава тог ентитета, којима се приступа у посматраном обрасцу, редундантно смештене у ентитете којима се у обрасцу приступа пре њега (у смислу поретка у функционалном ланцу). У супротном, ако програм свакако приступа неком ентитету, нема смисла оптимизовати приступ до вредности његових својстава јер се свакако не елиминише приступ посматраном ентитету (и случајан приступ блоку на диску).

С тим у вези, у формални модел се уводи појам *функционалних оптимизационих променљивих*, $f_{i1}^{opt}, f_{i2}^{opt}, \dots, f_{ik}^{opt}$ које се придружују својствима $f_{i1}, f_{i2}, \dots, f_{ik}$ ентитета неког типа E_x. Свака функционална оптимизациона променљива f_{ij}^{opt} дефинише да ли је приступ својству (функцији) f_{ij} посматраног ентитета оптимизован или не. Описани контекстни услов за оптимизацију приступа ентитету и његовим својствима у оквиру задатог обрасца може се

формално представити следећим логичким ограничењима (енгл. *constraints*) у предложеном оптимизационом моделу (по принципу „све или ништа“):

$$E_x^{opt} = f_{x1}^{opt} \wedge f_{x2}^{opt} \wedge \ldots \wedge f_{xk}^{opt}$$

$$f_{x1}^{opt} \oplus f_{x2}^{opt} \oplus \ldots \oplus f_{xk}^{opt} = 0$$

Коначни облик трансформисаних једначина модела цене и добити за анализирани образац приступа подацима на слици 6 дате су у наставку (усвојене вредности физичких параметара су редом $\tau = 0, \lambda = 1, \rho = 1, L = 0$).

$$rb(c) = (o_{12} + (o_{13} \vee (\, o_{23} \wedge \neg o_{12}\,)))\omega$$

$$wp(c) = (o_{12} + o_{13}) * w_1 + (d_1(o_{12} \vee o_{13}) + o_{13}) * w_2 + (d_1 d_2 o_{13} + d_2 o_{23}) * w_3$$

Вредности свих логичких израза могу бити или 1 или 0, па се третирају као мале целобројне вредности и користе у аритметичким изразима једначина цене и добити. Множење реалног израза са бинарном променљивом или бинарним (логичким) изразом даје или 0 или саму вредност тог израза. На тај начин се може у потпуности контролисати рачунање цене и добити редундансе у зависности од вредности основних оптимизационих променљивих.

Будући да су све независне променљиве у добијеним линеарним једначинама бинарне, као и да се комплексност образаца приступа одражава искључиво на логичке изразе (при чему једначине и даље остају линеарне), дефинисани оптимизациони проблем *DRE* може да се редукује на познати NP-комплетан проблем бинарног линеарног програмирања (енгл. *binary linear programming – BLP*). У наредном поглављу описан је формално / алгоритамски детаљан поступак редукције проблема постизања равнотеже редундансе у релационом моделу (*DRE*) на проблем бинарног линеарног програмирања (*BLP*).

5.9. Редукција проблема *DRE* на *BLP*

Оптимизације разматране у претходном поглављу биле су изведене из обрасца који је имао специфичну линеарну структуру. У пракси су обрасци ретко линеарни, већ су заправо разграната стабла. Форма једначина модела цене и добити није статичка нити предефинисана, већ зависи од структуре стабла обрасца (осим у специјалном случају за линеарне обрасце). Међутим, изведене једначине за линеарне обрасце могу се искористити и применити на појединачне линеарне

путање у нелинеарним стаблима образаца јер се од линеарних путања у стаблима могу извести описане линеарне оптимизације. Реални обрасци приступа подацима имају сложену структуру и могу бити бројни у општем случају. Са друге стране, поред структурне сложености самих стабала образаца, суштинска комплексност овог проблема крије се и у потреби за адекватним аналитичким моделовањем интерференције између оптимизација, што према доступној литератури, до сада није урађено на генерички и свеобухватан начин какав је предложен и описан у овој дисертацији. Редукција проблема DRE на проблем BLP спроводи се помоћу алгоритма чији су најважнији делови описани у наставку овог поглавља.

Алгоритам на улазу добија скуп образаца приступа подацима P и скуп свих функција које се појављују у обрасцима, F. Ради једноставности и фокусирања на суштинске елементе алгоритма, физички параметри у формулама цене и добити биће намерно изостављени у изразима алгоритма.

Излаз алгоритма јесте модел бинарног линеарног програмирања (BLP), у виду једначина цене уписа и добити при читању и придружених ограничења. Овакав програмски модел представља улаз за стандардне решаваче оптимизационих проблема (енгл. *solver*). У овом научном истраживању коришћен је решавач Gurobi [61].

5.9.1. Полазне дефиниције и конвенције именовања

Пре описа алгоритма, који је подељен на неколико семантичких делова, прво се уводе полазне претпоставке, дефиниције и конвенције именовања у овој секцији.

Претпоставља се да сваки ентитет има атрибут $type$ који пружа информацију о типу ентитета (елемент из скупа E), као и да свака функција има додатно својство ent преко ког се може добити тип аргумента функције. Свака оптимизација посматра се као уређена торка функција у композицији, $o = (f_{i1}, f_{i2}, \dots, f_{ik})$. Свакој функцији у оптимизацији o може се приступити индексирањем у нотацији $o[n]$. Приступ првој функцији записује се изразом $o[1]$, док се за динамички приступ последњој функцији у композицији уводи специјална нотација $o[last]$. Израз $o[i..j]$ даје оптимизацију добијену композицијом уланчаних функција из полазне

оптимизације почевши од функције $o[i]$ и закључно са функцијом $o[j]$, при чему мора да важи услов $i < j$.

Као што је већ дефинисано, један образац приступа подацима p представља се као торка $(V, U, M_V, M_U, r, \omega, \theta)$. Било којој компоненти торке обрасца може се приступити коришћењем нотације са оператором тачка. На пример, резултат израза $p.M_V$ представља скуп маркирања чворова стабла обрасца. Скуп свих грана стабла обрасца U дефинише се као скуп уређених парова $\{(src, dst) \mid src, dst \in V\}$, где свака усмерена грана има почетни чвор src и одредишни чвор dst. У складу са тим, дефинише се рестрикција $U[x]$ релације U као подскуп уређених парова који задовољавају услов x. На пример, скуп грана које полазе од кореног чвора у шаблону p представља се изразом рестрикције $p.U[src = p.r]$, где израз $p.r$ означава корени чвор.

Претпоставља се да су следеће услужне функције дефинисане над скупом свих оптимизација O:

- $opt(f_x, f_y)$: враћа скуп свих оптимизација у којима је f_x почетна функција, а f_y завршна функција у ланцу;
- $via(f_x)$: враћа скуп свих оптимизација које садрже функцију f_x у композицији на било којој позицији.

Комплетан алгоритам редукције проблема равнотеже редундансе у релационом моделу на бинарно линеарно програмирање подељен је на 4 мања семантичка фрагмента. У наставку текста дати фрагменти су нумерисани редом од 1 до 4 (нпр. Алгоритам 1).

5.9.2. Алгоритам за извођење оптимизација

Алгоритам 1 (приказан у листингу 7) користи се за извођење оптимизација у релационом моделу из стабла улазног обрасца приступа подацима p. Извођење оптимизација своди се на проблем налажења свих могућих путања у смеру грана у усмереном стаблу обрасца приступа подацима полазећи од корена. Алгоритам 1 почиње извођење оптимизација (композиција функција) од скупа грана суседних са кореним чвором стабла, као што је дефинисано изразом $p.U[src = p.r]$ на линији 1 у листингу 7 и делегира извођење оптимизација Алгоритму 2. Временска

сложеност Алгоритма 1 износи $O(D \cdot |V|)$, где D представља дужину најдужег ланца у задатом обрасцу приступа подацима, док $|V|$ представља број чворова у његовом стаблу.

Листинг 7:
Алгоритам за извођење оптимизација из стабла обрасца

Algoritam A.1: $opts\_from\_pattern$

Ulaz: p, obrazac pristupa podacima
Rezultat: O_p, skup optimizacija izvedenih iz obrasca p

```
for edge in p.U[src = p.r] do
    O_p ← O_p ∪ opts_from_subtree(p, edge)
end for
return O_p
```

Алгоритам 2 ($opts\_from\_pattern$), приказан у листингу 8, налази све могуће функционалне стазе у подстаблу полазног обрасца одређеног одредишним чвором улазне гране *edge*. Од свих пронађених функционалних путања у задатом подстаблу формирају се све могуће композиције функција (оптимизације) и враћају као резултат.

Листинг 8:
Алгоритам за извођење оптимизација из подстабла обрасца приступа подацима

Algoritam A.2: $opts\_from\_subtree$

Ulaz: p, obrazac pristupa podacima
Ulaz: $edge$, grana iz stabla obrasca p
Rezultat: O_p, skup optimizacija

```
for x in p.U[src = edge.dst] do
    o_x ← p.M_U(edge) ∘ p.M_U(x)
    O_p ← O_p ∪ {o_x}
    for o_s in opts_from_subtree(p, x) do
        O_p ← O_p ∪ {o_s, o_x ∘ o_s}
    end for
end for
return O_p
```

Алгоритам 2 обилази задато подстабло полазног обрасца рекурзивно. Да би се од грана подстабла обрасца формирале композиције функција, неопходно је

користити скуп маркирања грана. На линији 2 Алгоритма 2 гране стабла $edge$ и x пресликавају се на функције $p.M_U(edge)$ и $p.M_U(x)$ од којих се прави елементарна композиција. Добијена композиција c чува се у резултујућем скупу O_P, који је актуелан и локалан за тренутну активацију Алгоритма 2 (линија 3). На линији 4 функција врши рекурзиван позив тако што се као улазни аргумент, поред самог шаблона, прослеђује и тренутна грана x која је суседна са кореним чвором тренутног подстабла које је улазни аргумент позиваоца. Све оптимизације, које су добијене као резултат рекурзивног позива алгоритма, стављају се у резултујући скуп O_P позиваоца и додатно се од сваке оптимизације у резултату рекурзивног позива прави нова композиција за композицијом c формираном у току извршавања позиваоца на линији 5.

5.9.3. Извођење једначина за процену добити при читању

Једначине за процену добити при читању формирају се посебно за сваки образац из скупа P. Алгоритам 3 (листинг 9) генерише једначину за процену добити при читању за задати образац p. Као што је објашњено, у једначини за процену добити при читању на највишем нивоу фигуришу ентитетске оптимизационе променљиве. Утицај оптимизација на добит при читању у задатом обрасцу p пропорционалан је броју ентитета у обрасцу којима не мора да се приступи. У изразима ентитетских оптимизационих променљивих фигуришу функционалне оптимизационе променљиве, а оне напослетку зависе од основних оптимизационих променљивих. Због тога се за потребе постепеног формирања једначина овог типа користе две помоћне структуре података (мапе):

- E_o – мапа која пресликава ентитетску оптимизациону променљиву на скуп функционалних оптимизационих променљивих које одговарају функцијама које припадају типу ентитета представљеног датом променљивом и којима се приступа у посматраном обрасцу, и
- F_o – мапа која пресликава функционалну оптимизациону променљиву на скуп парцијалних логичких израза који дефинишу све могуће услове по којима се може оптимизовати приступ функције представљене датом функционалном оптимизационом променљивом.

Парцијални логички изрази, који се додају у мапу функција, наводе се у алгоритму у апстрактној нотацији помоћу угластих заграда ⟨...⟩. Оптимизације, ентитети и функције трансформишу се у одговарајуће оптимизационе променљиве (основне, ентитетске, функционалне, респективно) у време извршавања. На пример, дати апстрактни израз $\langle o \wedge o[1].ent \rangle$ може се трансформисати у конкретан логички израз $o_1 \wedge E_1^{opt}$. Приметити следеће: пошто се једначине дефинишу на нивоу појединачних образаца, алгоритам мора да обезбеди да свака променљива која се генерише и фигурише у моделу за BLP мора имати јединствено име.

Листинг 9:
Алгоритам за извођење једначине за процену добити при читању

Algoritam A.3: $rb\_equation$

Ulaz: p, obrazac pristupa podacima

1: $F_o : \{(fid, fcond : \{\})\}$
2: $E_o : \{(eid, fcond : \{\})\}$
3: $O_p \leftarrow opts\_from\_pattern(p)$
4: **for** o in O_p **do**
5: $\quad F_o[o[last]] \leftarrow \langle o \wedge \neg o[1].ent \rangle$
6: $\quad E_o[o[last].ent] \leftarrow \langle o[last] \rangle$
7: **end for**
8: **for** kv in F_o **do**
9: $\quad out\ \langle kv.fid = \bigvee_{c \in kv.fcond} c \rangle$
10: **end for**
11: **for** kv in E_o **do**
12: $\quad out\ \langle kv.eid = \bigwedge_{c \in kv.fcond} c \rangle$
13: $\quad out\ \langle \bigoplus_{c \in kv.fcond} c = 0 \rangle$
14: **end for**
15: $out\ \langle rb(p) = p.\omega \cdot \sum_{kv \in E_o} kv.eid \rangle$

Алгоритам прво изводи све оптимизације $opts$ из обрасца p коришћењем описаног алгоритма 1 на линији 3. У циклусу на линијама 4-7 за сваку добијену оптимизацију формирају се парцијални логички изрази који се додају у наведене мапе пресликавања E_o и F_o. На линији 5 алгоритам формира парцијалан услов оптимизације последње функције у тренутној оптимизацији. Приступ до вредности последње функције тренутне оптимизације је оптимизована ако је тренутна оптимизација примењена (нпр. $o_1 = 1$), и ако приступ до типа ентитета прве функције у оптимизацији није елиминисан у датом обрасцу (нпр. $E_1^{opt} = 0$).

Парцијални услов оптимизације за ентитетску оптимизациону променљиву која одговара домену последње функције у тренутној оптимизацији формира се на линији 6. Након тога, у циклусу на линијама 8-10 комплетирају се оптимизациони услови за функционалне оптимизационе променљиве и емитују се у BLP модел операцијама *out*. Приступ функцији је оптимизован (линија 9), ако је примењена бар једна од њених оптимизација (OR). У циклусу на линијама 11-14 комплетирају се контекстни услови за ентитетске оптимизационе променљиве. Приступ ентитету је оптимизован (линија 12) ако су све функције које му оригинално припадају и читају се у задатом обрасцу, оптимизоване (AND). У супротном (линија 13), ниједна функција не би требало да буде оптимизована, посматрајући изоловано само тренутни образац, будући да парцијалне оптимизације функција не елиминишу цену учитавања ентитета којем оригинално припадају (XOR). Коначно, једначина за процену добити при читању комплетира се на линији 15 формирањем суме ентитетских оптимизационих променљивих помножене фреквенцијом задатог обрасца. Временска сложеност алгоритма 3 идентична је као сложеност алгоритма 1, јер оба зависе од формирања свих могућих оптимизација из задатог стабла обрасца.

5.9.4. Извођење једначина за процену пенала при упису

Алгоритам 4 приказан у листингу 10 генерише једначине за процену пенала при упису за задату функцију f. Једначина се изводи поступно коришћењем помоћних структура података RA и WB за привремено смештање парцијалних израза за процену цене читања унапред (енгл. *read-ahead*), као и ажурирања уназад (енгл. *write-backward*), респективно. Структура података RA пресликава ентитетске оптимизационе променљиве ентитета (у стаблу за читање унапред) на пар скупова оптимизационих променљивих, *load* и *skip*. Скуп *load* представља скуп оних оптимизација које при променама својих функција захтевају приступ ентитету чија се ентитетска променљива пресликава на дати скуп *load*. Међутим, приступ до ентитета у стаблу читања унапред може да се оптимизује уколико примењене оптимизације у стаблу читања унапред то омогућавају – такве оптимизације се чувају у скупу *skip*. Слично томе, структура података WB пресликава сваки функционални ланац на његову одговарајућу реверзну кардиналност и скуп свих оптимизација у којима се тај функционални ланац појављује као префикс.

Листинг 10:
Алгоритам за извођење једначине за процену пенала при упису

Algoritam A.4: $wp\_equation$

Ulaz: f, funkcija
Ulaz: O, skup svih mogućih optimizacija

1: $RA : \{(eid, (load : \{\}, skip : \{\}))\}$
2: $WB : \{(fchain, (rc, opts : \{\}))\}$
3: **for** o in $O.via(f)$ **do**
4: **for** p in $o.positions(f)$ **do**
5: **if** $WB[o[1..p-1]] = null$ **then**
6: $WB[o[1..p-1]].rc \leftarrow \prod_{i=p-1}^{i=1} o[i].d$
7: **end if**
8: $WB[o[1..p-1]].opts \leftarrow o$
9: **for** $f_s = o[p+1]$ to $o[last]$ **do**
10: $RA[f_s.ent].load \leftarrow o$
11: **for** $f'_s = o[p+1]$ to f_s **do**
12: $RA[f_s.ent].skip \leftarrow O.opt(f'_s, f_s)$
13: **end for**
14: **end for**
15: **end for**
16: **end for**
17: $wp\_eq \leftarrow [\,]$
18: **for** wb in WB **do**
19: $wp\_eq \leftarrow \langle wb.rc \cdot \bigvee_{o \in wb.opts} o \rangle$
20: **end for**
21: **for** ra in RA **do**
22: $wp\_eq \leftarrow \langle (\bigvee_{o \in ra.load} o) \wedge \neg(\bigvee_{o \in ra.skip} o)) \rangle$
23: **end for**
24: $out\ \langle wp(f) = f.w \cdot \sum_{i=1}^{i=wp\_eq.len} wp\_eq[i] \rangle$

Алгоритам 4 прво одређује оптимизације на које утиче промена улазне функције *f* (линија 3). Добијене оптимизације се обрађују итеративно у петљи на линијама 3-16. На линији 4 одређују се позиције свих појављивања дате улазне функције у оптимизацији која се тренутно обрађује. Разматрањем више позиција исте функције у функционалном ланцу тренутне оптимизације решава се проблем рекурзивних функционалних ланаца. За сваку добијену позицију улазне функције проверава се да ли префикс функционалног ланца (лево од дате позиције у оптимизацији) постоји у мапи *WB* (линија 5). Уколико не, израчунава се реверзна кардиналност за посматрани функционални префикс и додаје се ново

пресликавање у мапу (линија 6). Додатно се у дати улаз у мапи додаје и оптимизациона променљива тренутне оптимизације на линији 9.

У петљи на линијама 9-14 алгоритам пролази кроз суфикс функционалног ланца тренутне оптимизације десно од тренутне позиције и додаје оптимизациону променљиву тренутне оптимизације у скуп *load* за сваки тип ентитета у ланцу читања унапред (линија 10). С тим у вези, у петљи на линијама 11-13 проналазе се све оптимизације, у ланцу читања унапред, које могу да елиминишу приступ типу ентитета који представља домен тренутне функције у суфиксу функционалног ланца и додају се у скуп *skip*.

Коначно, алгоритам саставља парцијалне изразе у комплетан израз за процену пенала при упису. Изрази за ажурирање уназад спајају се у петљи на линијама 18-20. На линији 19 генерише се израз за пенале ажурирања уназад као производ реверзне кардиналности и логичког услова под којим се врши ажурирање редундантних података уназад. Пенали се урачунавају у коначну цену ако је бар једна од оптимизација из скупа $WB.opts$ примењена. Изрази за пенале читања унапред изводе се у петљи на линијама 21-23. Ентитет у стаблу за читање унапред треба да буде учитан ако је најмање једна оптимизација у скупу $RA.load$ примењена и ниједна од оптимизација у скупу $RA.skip$ није примењена (линија 22). Комплетна једначина за процену пенала при упису формира се на линији 24 као сума описаних парцијалних израза.

Временска сложеност алгоритма 4 износи $O(|P| \cdot D^4 \cdot |V|)$, где $|P|$ представља број свих образаца, D представља дужину најдужег ланца на нивоу свих образаца и $|V|$ представља максималан број ентитета у неком од доступних образаца приступа подацима.

5.10. Сложеност проблема постизања равнотеже редундансе података (DRE)

У претходном поглављу показано је да постоји полиномијална редукција проблема постизања равнотеже редундансе података (DRE) на проблем бинарног линеарног програмирања (BLP), $DRE \leq_{poly} BLP$ са максималном временском сложеношћу $O(|P| \cdot D^4 \cdot |V|)$. Другим речима, показано је само да је проблем BLP најмање исте тежине као и проблем DRE (ако не и тежи).

Међутим, полазна хипотеза у овом поглављу јесте да оптимизациони проблем *DRE* припада класи NP-комплетних проблема. Доказ ове хипотезе базиран је на редукцији познатог NP-комплетног проблема *Binary Knapsack* на *DRE* у полиномијалном времену. Потребно је доказати да важи следећа неједнакост, $Binary\ Knapsack \leq_{poly} DRE$, или да је проблем *DRE* најмање исте сложености као и било који други NP-комплетан проблем. [62]

Нека се посматра инстанца X оптимизационог проблема *Binary Knapsack* коју сачињава скуп од n предмета, од којих сваки има вредност v_i и масу w_i и „ранац“ који може да понесе максималну масу W. Циљ оптимизације јесте да се одреди (под)скуп предмета $S \subseteq X$ (решење) који имају максималну укупну вредност уз услов да њихова укупна маса не пређе максималну носивост „ранца“ W. Дати оптимизациони проблем формулише се на следећи начин:

$$MAX \quad V = \sum_{i=1}^{i=n} v_i \cdot x_i$$

$$s.t. \quad \sum_{i=1}^{i=n} w_i \cdot x_i \leq W$$

$$s.t. \quad x_i \in \{0,1\}, v_i, w_i \in \mathbb{R}$$

Са друге стране, нека се посматра инстанца Y оптимизационог проблема DRE као скуп n линеарних навигационих образаца облика $A_i \xrightarrow{f_{bi}} B_i \xrightarrow{f_{ci}} C_i$, $1 \leq i \leq n$, где је број појављивања сваког од образаца означен са ω_i. У датој анализи не разматрају се физички параметри модела због поједностављења једначина, будући да физички параметри суштински не доприносе или не мењају ток доказа. Из сваког обрасца може се извести оптимизација $f_{bc} = f_{bi} \circ f_{ci}$, $1 \leq i \leq n$. Свака таква оптимизација, уколико је имплементирана, даје допринос при читању $r_{bi} = \omega_i \cdot o_i$, где o_i представља бинарну оптимизациону променљиву која одговара функцији f_{bc}. Без губитка општости, може се усвојити претпоставка да се функције f_{ci} не мењају након иницијализације, док се свака функција f_{bi} ажурира $w(f_{bi}) \equiv w_{bi}$ пута. На овај начин процена цене уписа за сваки шаблон има компоненту пенала читања унапред.

На основу наведених претпоставки, инстанца проблема DRE може се формулисати на следећи начин, где T представља максималну дозвољену вредност додатних режија при упису:

$$MAX \quad RB = \sum_{i=1}^{i=n} \omega_i \cdot o_i$$

$$s.t. \quad \sum_{i=1}^{i=n} w_{bi} \cdot o_i \leq T$$

$$s.t. \quad o_i \in \{0,1\}, \qquad \omega_i, w_{bi} \in \mathbb{R}$$

На основу дефиниција двеју инстанци оптимизационих проблема, може се дефинисати редукција $Q: X \to Y$ којом се инстанца X оптимизационог проблема *Binary Knapsack* може трансформисати на инстанцу Y оптимизационог проблема *DRE*.

$$Q = \{(x_i, o_i), (v_i, \omega_i), (w_i, w_{bi}), (W, T)\}, 1 \leq i \leq n$$

Из формалне дефиниције редукције уочава се да она представља једноставну трансформацију преименовања у линеарном времену $O(n)$.

$$\begin{cases} \sum_{i=1}^{i=n} v_i \cdot x_i \overset{Q}{\Leftrightarrow} \sum_{i=1}^{i=n} \omega_i \cdot o_i \\ \sum_{i=1}^{i=n} w_i \cdot x_i \leq W \overset{Q}{\Leftrightarrow} \sum_{i=1}^{i=n} w_{bi} \cdot o_i \leq T \end{cases}$$

Када год постоји решење за инстанцу проблема DRE, постоји и решење за инстанцу проблема *Binary Knapsack*. Постојање алгоритма за *Binary Knapsack*, који позива алгоритам оптимизационог проблема *DRE* као потпрограм, доказује да оптимизациони проблем *DRE* припада класи NP-комплетних проблема.

Листинг 11:
РЕДУКЦИЈА NP-КОМПЛЕТНОГ ПРОБЛЕМА *BINARY KNAPSACK* НА ПРОБЛЕМ *DRE*

Algoritam R.1 Redukcija opt. problema Binary Knapsack na problem DRE

Parametri: $X_K = \{(v_1, w_1), (v_2, w_2), ..., (v_n, w_n)\}$
Parametri: $W_K -$maksimalna nosivost "ranca"
Rezultat: $S_K \subseteq X_K, max \sum v_i$ and $\sum w_i \leq W_K$
Linearna redukcija:
1: $X_K = \{(v_i, w_i)\} \xrightarrow[w_i \to w_{bi}]{v_i \to \omega_i} Y_E = \{(\omega_i, w_{bi})\}$
2: $W_K \to T_E$
Poziv potprograma DRE:
3: $S_E = DRE(Y_E, T_E)$, where $S_E \subseteq Y_E$
Linearno reverzno preslikavanje rezultata problema DRE na Binary Knapsack:
4: $S_E = \{(\omega_i, w_{bi})\} \xrightarrow[w_{bi} \to w_i]{\omega_i \to v_i} S_K = \{(v_i, w_i)\}$
5: **return** S_K

5.10.1. ПРИМЕНА АУТОМАТСКЕ ДЕНОРМАЛИЗАЦИЈЕ НА АУКЦИЈСКУ АПЛИКАЦИЈУ

Након детаљног описа алгоритма за динамичко извођење једначина за процену добити при читању и додатних режија при упису, уз придружена ограничења модела, у овој секцији се формално представља метода оптимизације приступа подацима на примеру који је описан у глави 4. Опис методе је конципиран као секвенца корака до коначног решења у виду аутоматски изабраног скупа оптимизација у релационом моделу на основу карактеристика радног оптерећења.

Корак 1: Процес оптимизације релационог модела почиње прибављањем скупа образаца приступа подацима анализом задатог радног оптерећења. На основу датих (измерених) фреквенција извршавања случајева коришћења, добијају се фреквенције образаца приказане у табели 4.

Табела 4:
БРОЈЕВИ ПОЈАВЉИВАЊА ОБРАЗАЦА ПРИСТУПА ПОДАЦИМА У АУКЦИЈСКОЈ АПЛИКАЦИЈИ

образац	P1	P2	P3	P4
број појављивања	5.000	20.000	1.500	500

Корак 2: Обрадом образаца алгоритмом 1 (листинг 7) добија се скуп свих могућих оптимизација O, које су приказане у наставку.

$o_{17} = b_L \circ a$ $o_{20} = i \circ c \circ p$ $o_7 = b \circ n$
$o_{14} = b_L \circ b$ $o_{23} = i \circ c \circ p \circ n$ $o_{21} = p \circ n$
$o_{16} = b_L \circ b \circ n$ $o_{26} = i \circ c \circ p \circ p$ $o_{24} = p \circ p$
$o_{13} = b_L \circ t$ $o_{30} = i \circ c \circ p \circ p \circ n$ $o_{28} = p \circ p \circ n$
$o_{10} = m \circ h$ $o_{29} = c \circ p \circ p \circ n$ $o_{11} = h \circ n$
$o_{12} = m \circ h \circ n$ $o_{33} = w \circ b \circ n$ $o_3 = c \circ n$
$o_1 = i \circ n$ $o_{35} = w \circ t$ $o_{19} = c \circ p$
$o_2 = i \circ c$ $o_{34} = w \circ a$ $o_{22} = c \circ p \circ n$
$o_4 = i \circ c \circ n$ $o_{31} = w \circ b$ $o_{25} = c \circ p \circ p$
$o_{18} = s \circ n$

Корак 3: На основу добијених оптимизација изводе се једначине за процену добити при читању појединачних образаца коришћењем алгоритма 3 (листинг 9). За сваки образац алгоритам 3 је генерисао по једну једначину.

$$rb(p_1, c_x) = 5000 \cdot \left(A_{p1}^{opt} + B_{p1}^{opt} + P_{p1}^{opt} + C_{p1}^{opt} + U_{p1}^{opt}\right)$$

$$rb(p_2, c_x) = 20000 \cdot \left(B_{p2}^{opt} + U_{p2}^{opt}\right)$$

$$rb(p_3, c_x) = 1500 \cdot \left(A_{p3}^{opt} + M_{p3}^{opt} + P_{p3}^{opt} + C_{p31}^{opt} + C_{p32}^{opt} + C_{p33}^{opt} + U_{p3s}^{opt} + U_{p3m}^{opt}\right)$$

$$rb(p_4, c_x) = 500 \cdot \left(A_{p4}^{opt} + M_{p4}^{opt} + M_{p4}^{opt} + C_{p31}^{opt} + C_{p32}^{opt} + C_{p33}^{opt} + U_{p1s}^{opt} + U_{p4m}^{opt} + U_{p4b}^{opt} + P_{p3}^{opt}\right)$$

Вредност оптимизационе променљиве B_{p1}^{opt} (за тип ентитета B у обрасцу P1) формира се помоћу следеће једначине.

$$B_{p1}^{opt} = \left(o_{13} \wedge \neg A_{p1}^{opt}\right) \wedge \left(o_{14} \wedge \neg A_{p1}^{opt}\right) \wedge \left(o_{17} \wedge \neg A_{p1}^{opt}\right)$$

Приступ типу ентитета B у обрасцу $P1$ је оптимизован акко је оптимизован приступ до свих његових својстава (функција) које се појављују у обрасцу $P1$.

$$\left(o_{13} \wedge \neg A_{p1}^{opt}\right) \oplus \left(o_{14} \wedge \neg A_{p1}^{opt}\right) \oplus \left(o_{17} \wedge \neg A_{p1}^{opt}\right) = 0$$

Корак 4: Једначине за процену пенала при упису, добијене су коришћењем алгоритма 4 (листинг 10). Алгоритам 4 је генерисао 24 једначине за процену режија при упису. Претрага најбољег решења контролисана је помоћу 206 ограничења. Због величине генерисаног бинарног линеарног програма у тексту ће бити илустровано свега неколико најрепрезентативнијих једначина.

У једначини за процену режија при ажурирању функције b_L (у неком ентитету аукције) постоји само компонента читања унапред јер је функција b_L водећа у изведеним функционалним ланцима.

$$wp(b_L, c_x) = 4000 \cdot [(o_{16} \wedge \neg o_7) + (o_{13} \vee o_{14} \vee o_{16} \vee o_{17})]$$

Свака промена пресликавања b_L захтева читање нових вредности редундантних података из стабла ентитета нове понуде. Да би приступ типу ентитета B био неопходан, услов $o_{13} \vee o_{14} \vee o_{16} \vee o_{17}$ мора да буде задовољен/истинит. Исто тако, приступ типу ентитету U је неопходан ако је оптимизација $o_{16} = b_L \circ b \circ n$ примењена и функција n није редундантно копирана у тип ентитета B, што се контролише логичким условом $o_{16} \wedge \neg o_7$.

Једначина за процену режија при ажурирању функције c садржи и компоненту за процену пенала ажурирања уназад, као и компоненту за процену пенала читања унапред. Пошто операције за промену категорија производа нису биле присутне у задатом радном оптерећењу, приликом поставке оптимизационог проблема подразумевано је разматран случај једне промене функције c.

$wp(c, c_x) = (o_2 \vee o_4 \vee o_{20} \vee o_{23} \vee o_{26} \vee o_{30}) \cdot d_i + (o_3 \vee o_4 \vee o_{19} \vee o_{20} \vee o_{22} \vee o_{23} \vee o_{25} \vee o_{26} \vee o_{29} \vee o_{30}) + \big((o_{22} \vee o_{23} \vee o_{25} \vee o_{26} \vee o_{29} \vee o_{30}) \wedge (\neg o_{21} \wedge \neg o_{24})\big) + \big((o_{29} \vee o_{30}) \wedge (\neg o_{21} \wedge \neg o_{28})\big)$.

Израз $(o_2 \vee o_4 \vee o_{20} \vee o_{23} \vee o_{26} \vee o_{30}) \cdot d_i$ процењује пенале ажурирања уназад коришћењем реверзне кардиналности функције i. Сви остали чланови израза дефинишу услове приступања (унапред) типовима ентитета C. Пошто у обрасцима *P3* и *P4* постоје функционални ланци категорија дужине 3, у изразу се појављују три посебна израза за сваки тип ентитета у поменутом рекурзивном ланцу.

Корак 5: За решавање описаног бинарног линеарног програма коришћен је стандардни решавач Gurobi [61]. Овај решавач је после 5 итерација и за око 0.05s пронашао конфигурацију са најбољим односом добити читања и пенала при упису заснованим на бројању редова. Оптимизације у коначном решењу дате су у листингу 12 и приказане на слици 7. Испрекидане линије одсликавају примењене оптимизације. Сиви чворови представљају типове ентитета којима се обавезно приступа у трансакцијама радног оптерећења.

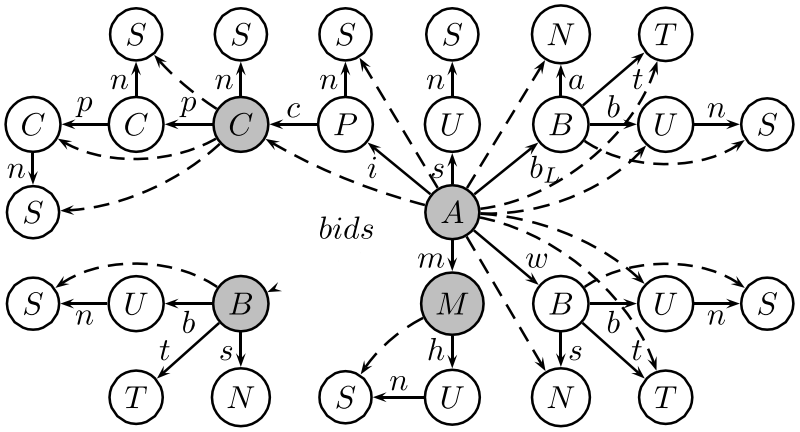
Листинг 12:
Аутоматска денормализација релационог модела података аукцијске апликације

$o_1 = i \circ n$
$o_2 = i \circ c$
$o_{21} = p \circ n$
$o_{24} = p \circ p$
$o_{28} = p \circ p \circ n$
$o_7 = b \circ n$
$o_{35} = w \circ t$
$o_{34} = w \circ a$
$o_{31} = w \circ b$
$o_{11} = h \circ n$
$o_{17} = b_L \circ a$
$o_{14} = b_L \circ b$
$o_{13} = b_L \circ t$

Укупна процењена вредност пенала при упису за све функције, које учествују у аутоматски изабраним оптимизацијама износи 7.260 редова, док укупна добит при читању аутоматски денормализованог релационог модела за аукције износи 38.600 редова, или концизније *(38.600, 7.260)*. У поређењу са решењем из главе 4, које је предложено мануелним поступком, $m = (43.700, 16.700)$, аутоматски пронађено решење $a = (38.600, 7.260)$ даје бољи однос цене и добити са разликом $a - m = (-5.100, -9.440)$, где је добит при читању умањена одустајањем од неких оптимизација зарад постизања мање цене пенала при упису, узимајући у обзир карактеристике радног оптерећења. Добити при читању појединачних образаца приступа подацима остварене мануелним и аутоматским поступком упоређене су у табели 5. Пажљивим прегледом резултата у табели 5 наизглед се може учинити да је добит појединачних образаца при ручно одабраној денормализацији већа од укупне процењене. Међутим, дати резултати су разматрали само добит при читању, док је занемарена и цена пенала читања унапред код мануелне денормализације. Резултати показују како се аутоматском денормализација циљано одустаје од оптимизација читања зарад одржавања њихове равнотеже за пеналима при упису.

Табела 5:
Поређење добити при читању образаца приступа подацима у аукцијској апликацији остварених мануелном и аутоматском денормализацијом

	P1	**P2**	**P3**	**P4**
број појављивања обрасца	5.000	20.000	1.500	500
добит при читању (аутоматски)	10.000	20.000	6.000	2.500
добит при читању (ручно)	15.000	20.000	9.000	4.000



Слика 7: Приказ аутоматске денормализације релационог модела података аукцијске апликације

На обједињеном дијаграму свих образаца за приступ подацима у посматраној аукцијској апликацији приказаног на слици 7, који визуализује аутоматску денормализацију нормализованог релационог модела података аукцијске апликације, датог у листингу 5, на основу унапред задатих параметара радног оптерећења, може се уочити да су оптимизације попут $o_{33} = w \circ b \circ n$ или $o_{30} = i \circ c \circ p \circ p \circ n$ одбачене због велике цене пенала при упису у случају потенцијалних измена имена корисника и назива категорија, респективно. На превелику цена пенала при ажурирању поменутих података доминантно утичу реверзне кардиналности функција $b: B \to U$ и $c: P \to C$ које су сувише велике узимајући у обзир карактеристике задатог радног оптерећења.

Предложена метода оптимизације приступа подацима у објектно-релационом мапирању експериментално је испитана помоћу једног реалног теста перформанси. Најважнији резултати, као и детаљан опис експерименталних услова, дати су у глави 6.

6. Експериментална анализа

Предложена метода оптимизације приступа подацима у релационој бази података експериментално је анализирана коришћењем теста перформанси (енгл. *benchmark*) за трансакционе системе доступног на веб порталу непрофитне организације која се бави дефинисањем тестова перформанси за базе података и индустријске трансакционе системе – *The Transaction Processing Performance Council* (*TPC*). Организација *TPC* дефинише тестове перформанси како за трансакционе (OLTP), тако и за аналитичке (OLAP) системе.

Спецификацијама тестова перформанси за базе података, које дефинише ова организација, прописују се: а) релационе шеме података, б) карактеристике и статистике самих података у бази података, в) дозвољени оквири финог подешавања (енгл. *fine tuning*) система пре извршавања теста перформанси и г) карактеристике радног оптерећења (енгл. *workload*). Сви од наведених критеријума валидног теста перформанси дефинисани су на основу анализе реалних система са циљем да што верније симулирају њихово стање и радно оптерећење.

За потребе оцене квалитета (енгл. *evaluation*) предложене методе коришћен је тест перформанси из категорије „E“, под називом TPC-E, за трансакционе (OLTP) системе. Према спецификацији теста TPC-E [63], дати тест симулира активност брокерске куће и састоји се од 12 типова трансакција различите сложености и фреквенције која прати емпиријску расподелу. Разматрају се следећи типови трансакција: *Broker Volume (BV)*, *Customer Position (CP)*, *Market Watch (MW)*, *Security Detail (SD)*, *Trade Lookup (TL)*, *Trade Order (TO)*, *Trade Result (TR)*, *Trade Status (TS)*, *Trade Update (TU)*, *Data Maintenance (DM)*. Све наведене трансакције извршавају се над базом података са нормализованом шемом која је дефинисана у спецификацији [63].

TPC-E тест перформанси долази и са два генератора података: а) генератор података за тестирање и б) генератор трансакција (радног оптерећења). Радно оптерећење представљено је скупом трансакција генерисаних у складу са задатом емпиријском расподелом њихове заступљености [63]. Поред тога, генератор трансакција је пројектован тако да омогући и прописану фреквенцију извршавања појединих условних делова трансакција, који уносе варијације у радно оптерећење.

6.1. Поставка окружења за тестирање

За тестирање перформанси коришћен је систем за управљање базама података Microsoft SQL Server верзија 2016 на стандардној корисничкој машини (енгл. *commodity hardware*) са оперативним системом Windows 10, процесором Intel i5 са 4 језгра и учестаношћу такта од 3.2GHz.

Инсталиране су иницијално две базе података:

а) референтна TPC-E база података са нормализованом шемом и

б) копија референтне TPC-E базе података, али са денормализованом релационом шемом која је добијена применом предложене оптимизационе методе на основу анализе трансакција у прописаном радном оптерећењу.

Обе базе података садржале су идентичне податке, с тим што је база са денормализованом релационом шемом додатно садржала и редундантне податке добијене применом предложених оптимизација.

Наведене базе података биле су размештене (енгл. *deployed*) на два диска:

а) SSD диск за смештање фајлова са подацима и

б) HDD за смештање трансакционих дневника (енгл. *transaction log files*).

Основни показатељи перформанси коришћених дискова приказани су у табели 6.

Табела 6:
Перформансе дискова коришћених у експерименталној анализи

	Секвенцијално читање	**Секвенцијално писање**	**Случајно читање**	**Случајно писање**
SSD	200MB/s	126MB/s	133MB/s	90MB/s
HDD	180MB/s	180MB/s	0.7MB/s	1.7MB/s

6.2. Карактеристике улазних података

Референтна нормализована база података добијена је учитавањем 75GB сирових података (енгл. *raw data*) генерисаних помоћу поменутог генератора података за TPC-E. Величина референтне базе података након учитавања података износила је 120GB (са свим прописаним приступним структурама). Дати генератор података долази са унапред уграђеном емпиријском расподелом величина појединих референтних табела, тако да се њихове конкретне величине могу добити задавањем одређених мултипликативних (енгл. *sizing*) фактора на улазу. Додатно се може подесити да генератор података произведе податке симулацијом функционисања брокерске куће у задатом временском периоду, на пример.

За потребе тестирања генератор података је подешен тако да се постигну бројеви редова у референтним табелама приказани у табели 7.

Табела 7:
Бројеви редова у референтним табелама TPC-E базе података

CUSTOMER	20.000
CUSTOMER_ACCOUNT	100.000
TRADE	172.800.000
SETTLEMENT	172.800.000
COMPANY	10.000
SECURITY	13.700

Према спецификацији [63], трансакције су подељене у логичке целине које се називају оквири (енгл. *frames*). Оквири представљају јединичне (атомске) функционалне делове трансакција, и могу се извршавати опционо. За потребе тестирања перформанси предложене методе, оквири су имплементирани као SQL процедуре (енгл. *stored procedures*), а оптерећење је генерисано као скуп позива датих SQL процедура.

6.3. Обрасци приступа подацима у TPC-E радном оптерећењу

Обрасци приступа подацима добијени су анализом извршавања (профилисањем) контролне смесе од 10.000 трансакција коришћењем алата SQL Server Profiler. Број трансакција у контролној смеси изабран је тако да се осигура

да се сваки упит у трансакционим оквирима (енгл. *frames*) појави бар једном у профилу чиме се добија репрезентативан узорак, а у исто време и довољно ефикасна припремна анализа образаца приступа подацима. У табели 8 приказано је 13 образаца приступа подацима добијених анализом профила извршавања контролне смесе трансакција.

За сваки образац у табели 8 наводи се:

- кратко кодно име (ради концизнијег референцирања у даљем тексту),
- индикатор сложености стабла обрасца представљен као комбинација броја чворова и максималне дубине стабла,
- број појављивања обрасца у контролној смеси,
- густина ρ релевантних података који се дохватају из нормализованих табела у обрасцу,
- физичка цена (*cost*) упита који чине посматрани шаблон добијена од оптимизатора плана упита,
- нормализована цена обрасца θ у односу на све остале обрасце у смеси.

Кодно име обрасца састоји се од три сегмента: а) двословног акронима пуног имена у спецификацији, б) идентификатора трансакционог оквира у ком се појављује, као и в) јединственог идентификатора обрасца. На пример, кодно име *BVF1P1* односи се на образац *P1* добијен из оквира *Frame 1* трансакције под називом *Broker Volume* (*BV*).

Наведени обрасци добијени су анализом имплементације трансакција и релевантни делови те имплементације могу се пронаћи на следећим страницама спецификације теста перформанси [63]: *BVF1P1* (p.89), *CPF1P2* (p.87), *CPF2P3* (p.89), *MWF1P4* (p.99), *MWF1P5* (p.99), *MWF1P6* (p.99). *SDF1P7* (p.105), *SDF1P8* (p.106), *SDF1P9* (p.108), *TLF1P10* (p.112), *TRF3P11* (p.154), *TSF1P12* (p.162), *TSF1P13* (p.163).

6.4. Одређивање вредности физичких параметара оптимизационог модела

Физички параметри су одређени на следећи начин:

- $\tau = 0$: пошто је експериментална анализа спроведена над базом података која није дистрибуирана, занемарује се цена преноса података до чвора на ком се врши обрада података (усвојено је да је вредност овог параметра идентична за сваки приступ подацима, па је занемарен у даљој анализи, али свакако је један од планова наставка истраживања анализа оптимизационог модела и његова примена у домену дистрибуираних база података),
- $L = 0.1$: емпиријски тежински фактор за увећање цене приступа подацима који враћају мањи проценат релевантних података у извршавању,
- $\lambda = 0.01$: просечна цена лоцирања реда у било којој табели у бази података добијена као средња вредност цене приступа реду у свим табелама које су учествовале у обрасцима (добијена од оптимизатора упита),
- w: број промена функција добијене су претрагом профила извршавања контролне смесе добијеног помоћу алата SQL Server Profiler,
- d: реверзне кардиналности функција добијене су упитима над нормализованом базом података након учитавања генерисаних података.

6.5. Аутоматска денормализација TPC-E релационе шеме

На основу задатих образаца Алгоритам 1 је генерисао 198 могућих оптимизација.

Применом алгоритама 3 и 4 на наведени скуп образаца и карактеристика контролног оптерећења (учестаности појављивања образаца, фреквенција уписа појединачних функција, физичких параметара) добијен је бинарни линеарни програм са 598 једначина (ограничења).

Решавач Gurobi [61] пронашао је најбоље решење, односно конфигурацију, за 0,07s (на стандардном корисничком рачунару) која је садржала 63 од 198 могућих оптимизација. Изабране оптимизације у најбољој пронађеној конфигурацији утицале су на следећих 10 од 13 образаца приступа подацима: *B1F1P1*, *CPF1P2*, *MWF1P4*, MWF1P5, *MWF1P6*, *SDF1P7*, *SDF1P9*, *SDF1P9*, *TRF3P11* и *TSF1P13*.

Као резултат извршавања добијеног бинарног линеарног програма добијене су две процене добити: а) прва заснована на бројању ентитета и б) друга

софистицираније која укључује и физичке параметре, као што је приказано у последње две колоне у табели 8.

ТАБЕЛА 8:
ПРЕДИКЦИЈА ДОБИТИ ОБРАЗАЦА ПРИСТУПА ПОДАЦИМА ИЗ КОНТРОЛНИХ TPC-E ТРАНСАКЦИЈА ЗА НАЈБОЉУ ПРОНАЂЕНУ ДЕНОРМАЛИЗАЦИЈУ

			Параметри упита			**Предикција**	
Образац	**Број чворова / макс. дубина**	**Фреквенција појављивања**	ρ	$cost$	θ	**Бројање редова**	**Цена**
BVF1P1	9 / 5	548	0,34	1,600	0,551	1.096	252,85
CPF1P2	7 / 3	1.462	0,73	0,170	0,059	1.462	63,20
CPF2P3	9 / 2	744	0,45	0,088	0,030	0	0,00
MWF1P4	3 / 2	1.215	1,00	0,007	0,002	1.215	1,58
MWF1P5	5 / 3	117	0,32	0,030	0,010	234	0,85
MWF1P6	3 / 2	118.274	0,19	0,006	0,002	118.274	154,07
SDF1P7	39 / 4	1.567	0,73	0,023	0,008	4.701	8,16
SDF1P8	4 / 2	1.567	0,47	0,014	0,005	3.134	5,74
SDF1P9	3 / 2	1.567	0,86	0,085	0,003	1.567	3,06
TLF1P10	10 / 2	1.024	0,47	0,700	0,241	0	0,00
TRF3P11	3 / 2	428	1,00	0,200	0,002	428	0,55
TSF1P12	13 / 3	2.130	0,43	0,240	0,083	0	0,00
TSF1P13	6 / 2	2.130	0,23	0,009	0,003	4.260	4,68

На основу добијене оптималне конфигурације, оптимизатор је произвео SQL скрипту за денормализацију референтне TPC-E шеме базе података, и у складу са тим ажурирана је имплементација трансакција тако да користе имплементиране оптимизације. За потребе ове експерименталне анализе и доказивања концепта, преусмеравање приступа подацима у трансакцијама рађено је мануелно. Међутим, шира идеја и филозофија приступа овом проблему, из које је проистекла и ова дисертација као део тог истраживања, јесте да је за ефектну употребу оваквих оптимизација неопходан софистицирани слој софтвера који је у стању да динамички, у време извршавања прилагоди приступ подацима који је најефикаснији за тренутни образац приступа подацима.

6.6. ТЕСТ ПЕРФОРМАНСИ АУТОМАТСКИ ДЕНОРМАЛИЗОВАНЕ TPC-E РЕЛАЦИОНЕ ШЕМЕ

За потребе тестирања перформанси генерисана је смеса од 100.000 трансакција (енгл. *transaction mix*). Да би се јасно пратили ефекти оптимизација на

сваки од наведених типова трансакција, од полазне интегралне смесе генерисане су додатно и хомогене смесе трансакција, тако да је свака хомогена парцијална смеса садржала искључиво један тип трансакција (видети табелу 9).

6.6.1. Опис коришћене методе мерења перформанси

За сваку парцијалну смесу трансакција резултати приказани у табели добијени су понављањем мерења неколико пута (нпр. 5-10) појединачно над нормализованом и денормализованом базом података. Пре промене базе података, сервер базе података је рестартован да би се елиминисао међусобни утицај мерења. Након поновног покретања (енгл. *restart*) сервера базе података, прво би се пустило контролно извршавање сваке парцијалне смесе, које није мерено, са циљем да се прође фаза „загревања“ (енгл. *warm-up*) и систем уведе у стационарно стање, а затим би се извршавање сваке смесе поновило још неколико пута и као коначан резултат израчунала аритметичка средина добијених мерења (као статистика централне тенденције мерених показатеља перформанси).

6.6.2. Физички параметри базе података разматрани при мерењу

Коришћењем алата *SQL Server Profiler* мерени си следећи параметри перформанси извршавања упита (на физичком нивоу):

- просечан број логичких читања по трансакцији (*LRd*),
- просечан број логичких уписа по трансакцији (*LWr*),
- време одзива (парцијалне) смесе трансакција (*T*).

Логичка читања и уписи [64] односе се на број (логичких) приступа страницама базе података (било да су у питању индекси или табеле са подацима) од стране система за управљање базом података. Добијени број логичких приступа (читања или писања) не мора нужно да одсликава реалан улазно-излазни пренос података због тога што се странице са подацима некада налазе у оперативној меморији или у процесорском кешу и на тај начин (логички) приступ таквим страницама не захтева приступ секундарној меморији. Може се закључити да број логичких читања засигурно одражава имплицитну сложеност упита, али да се не може довести у линеарну корелацију са временом одзива, па су у том смислу и анализирани добијени експериментални резултати.

ТАБЕЛА 9:
УПОРЕДНИ ПРИКАЗ РЕЗУЛТАТА ТЕСТА ПЕРФОРМАНСИ TPC-E ЗА НОРМАЛИЗОВАНУ, ОПТИМАЛНО ДЕНОРМАЛИЗОВАНУ И ПОТПУНО ДЕНОРМАЛИЗОВАНУ БАЗУ ПОДАТАКА

		Нормализована			**Оптимално денормализована**			**Потпуно денормализована**		
Транс.	**Фрекв.**	**LRd**	**LWr**	**T_N[s]**	**LRd**	**LWr**	**T_O[s]**	**LRd**	**LWr**	**T_F[s]**
BV	4.946	47.718	0	1.762	25.061	0	193	62.439	0	228
CP	13.116	226	0	27	212	0	25	551	0	54
MW	18.165	1.422	0	64	649	0	48	648	0	48
SD	14.128	2.214	0	244	2.137	0	226	2.126	0	230
TL	8.069	405	0	159	422	0	153	628	0	210
TO	10.191	100	5	18	104	6	19	112	6	20
TR	10.192	155	5	53	156	5	56	127	6	99
TS	19.175	533	0	79	431	0	73	166	0	137
TU	2.018	1.089	18	61	1.087	18	60	1.111	31	106
DM	1.000	14.757	4	65	14.609	4,4	69	14.307	4,6	71

6.6.3. ДИСКУСИЈА РЕЗУЛТАТА ПАРЦИЈАЛНИХ СМЕСА ТРАНСАКЦИЈА

У табели 9 може се запазити, упоредном анализом резултата добијених над нормализованом и оптимално денормализованом TPC-E базом података, да је највеће побољшање времена одзива остварено за парцијалне смесе у којима су постојали упити који су приступали потпуно денормализованим пакетима свих захтеваних података у једном реду табеле ($BV - 89\%$, $MW - 25\%$).

У парцијалним смесама *SD* (7%) и *TS* (8%) упити у трансакцијама нису приступали потпуно денормализованим табелама, тако да су још увек била потребна спајања табела (JOIN). Време одзива парцијалних смеса *TO*, *TR* и *DM*, које искључиво имају упите за ажурирање података, било је очекивано деградирано због повећаних режија одржавања конзистентности редундантних података, али за само 6%, јер ажурирања нису била честа.

6.6.4. ЗНАЧАЈ ОПТИМАЛНЕ ДЕНОРМАЛИЗАЦИЈЕ

Да би се указало на значај и потребу за постизањем оптималне денормализације, као резултата софистицираног оптимизационог проблема, извршена је експериментална провера хипотезе да је оптимална денормализација супериорна у односу на методу потпуне и неселективне денормализације у оквиру

које се занемарују негативни ефекти примењених оптимизација на операције ажурирања података, као и заузеће додатног простора.

За потребе овог експеримента припремљена је и трећа база података добијена потпуном денормализацијом оптимално денормализоване базе података тако што су примењене и оптимизације (преосталих 135 од укупно 198) које је оптимизатор иницијално одбацио као неефикасне у складу са описаним моделом цене и добити. Резултат потпуне денормализације релационог модела довео је до тога да је сваки од образаца приступа подацима могао да се изврши упитом над само једном одговарајућом денормализованом табелом јер су сви релевантни подаци „агресивно“ сложени у појединачним редовима датих табела.

Интересантно је приметити да се предложеним моделом оптимизације одлука о одбијању или прихватању оптимизација није доносила само на основу учестаности операција читања и ажурирања подацима који су учествовали у примењеним оптимизацијама. Поред учестаности, разматран је још један важан параметар, а то је степен гранања уназад (реверзна кардиналност). Првобитно одбачене оптимизације у већини случајева односиле су се на оптимизацију навигације која би полазила од редова у великим табелама, као што су TRADE и TRADE_HISTORY и настављала се ка (знатно) мањим табелама. Након додавања редундантних података из знатно мањих табела у такве велике табеле, због велике реверзне кардиналности, величина базе података значајно је порасла. Примера ради, мали подаци као што су разне врсте статуса, чије редундантно копирање у велике табле је одбачено предложеним оптимизационим моделом, мултиплицирани су потпуном денормализацијом у милионима редова великих табела! Величине све три базе података коришћених у овом експерименту приказане су у табели 10.

Потпуно денормализована велика табела TRADE заузимала је простор од 162GB, док је потпуно денормализована табела TRADE_HISTORY заузимала простор од 64GB у потпуно денормализованој бази података. Извршавање упита над овако великим табелама било је практично немогуће без адекватних покривајућих индекса (енгл. *covering indexes*). Највећи покривајући индекс над табелом TRADE заузимао је око 50GB, док је за највећи покривајући индекс над табелом TRADE_HISTORY било потребно одвојити 26GB простора на диску. Овако

велики индекси су и сами допринели повећању улазно-излазног протока података током извршавања теста перформанси. Добијени резултати тестирања парцијалних смеса трансакција над потпуно денормализованом TPC-E шемом приказани су у оквиру табеле 9.

Посебна пажња на овом месту посвећује се измереним показатељима перформанси за трансакцију типа *BV* који су приказани у првом реду табеле 9 за све три наведене базе података. Може се приметити да је потпуном денормализацијом релационе шеме којој се приступа у датој трансакцији, и потпуним задовољењем захтева трансакције приступом само једној табели и једном реду у њој, број логичких приступа страницама значајно смањен (са 47.718 над нормализованом на 25.061 логичких приступа над денормализованом базом података), што се према очекивањима и постављеним хипотезама одразило и на смањење времена одзива (са 1.762 секунди над нормализованом на 193 секунде над денормализованом базом података). Међутим, у случају треће, потпуно денормализоване базе података, постоји наизглед неочекивана негативна корелација између броја логичких приступа и времена одзива поређењем добијених резултата са резултатима извршавања трансакција типа BV над нормализованом базом података. У случају потпуне денормализације, примећује се повећање броја логичких приступа (са 47.718 над нормализованом на 62.439 над потпуно денормализованом шемом), али истовремено и драстично смањење времена одзива трансакције BV при приступу потпуно денормализованој TPC-E бази података (228 секунди) у односу на приступ подацима у референтној нормализованој бази података (1.762 секунди).

Иако није потврђено експерименталним мерењем, које би захтевало софистицираниј инструментацију на нивоу оперативног система и додатно време, аутор износи могуће објашњење посматраног ефекта искључиво на теоретском нивоу, али засновано на искуству и резултатима експерименталних мерења која јесу била извршена.

Потпуном денормализацијом, која је имала утицај посебно на велике табеле у TPC-E бази података, драстично је порасла величина датих табела као и броја блокова за њихово смештање на диску. Са повећањем броја блокова за смештање

датих табела, расте и број улаза, као и величина, њихових приступних структура (потребан је већи број блокова на диску и за перзистенцију индекса).

Повећање броја логичких приступа страницама у односу на нормализовану шему, може да потиче од драстичног повећања броја блокова којима је било потребно приступити услед смањења броја редова по блоку (због денормализације) и случајног приступа реду наметнутог природом теста.

Са друге стране, драстично смањење времена одзива услед денормализације, упркос повећању броја логичких приступа, може да буде последица чињенице да је за свако извршавање навигације у трансакцији типа *BV* било довољно приступити само једном блоку денормализоване табеле. У сваком случају, у будућности, вреди обратити пажњу и на овај ефекат.

6.6.5. Мерење перформанси комплетне смесе трансакција

Коначно, комплетна смеса од 100.000 случајно уређених трансакција извршена је над све три базе података и резултати теста су приказани у табели 10.

Табела 10:
Утицај стратегије денормализације на време одзива смесе трансакција и величину TPC-E базе података

TPCE шема	**T [s]**	**ΔT [%]**	**Вел. базе [GB]**	**Промена вел. базе [%]**
Нормализована	2.910	--	120	--
Оптимално денормализована	1.005	-65	150	+25
Потпуно денормализована	1.240	-57	380	+316

У односу на време одзива добијено извршавањем комплетне смесе трансакција над референтном нормализованом TPC-E базом података, време одзива исте смесе побољшано је за 65% захваљујући оптималној денормализацији и за 57% применом потпуне денормализације. Заузеће простора на диску код оптимално денормализоване базе података увећано је за 25%, док је заузеће простора код потпуно денормализоване увећано за 316% у односу на нормализовану базу података. Иако је оптимална денормализација дала боље време одзива за само 19% у односу на потпуну денормализацију (због тога што редундантни подаци у великим табелама нису уопште ажурирани), оптимална

денормализација је била далеко ефикаснија по питању контроле експанзије простора у секундарној меморији.

6.7. Анализа квалитета предложеног модела цене и добити

У склопу експерименталне анализе испитиван је и квалитет предикција добијених предложеним оптимизационим моделом на контролној смеси од 10.000 трансакција. Предикције модела поређене су са резултатима хомогених парцијалних смеса контролне смесе. Резултати анализе приказани су у табели 11. Бројеви логичких читања по сваком типу трансакције приказани су у колони *N*, за нормализовану базу података и у колони *D* за денормализовану базу података.

Табела 11:
Поређење предикција предложеног оптимизационог модела са измереним логичким приступима страницама у контролној смеси трансакција

		Измерени логички приступи			**Предикција**	
Трансакција	**Фрекв.**	**N [pages]**	**D [pages]**	**△P=N-D**	**Цена**	**Редови**
BV	495	23.620.749	12.405.379	11.215.370	252,85	1.096
CP	1.312	297.332	278.887	18.445	63,20	1.462
MW	1.817	1.754.453	349.778	1.404.675	156,50	120.500
SD	1.413	161.897	49.870	112.027	16,96	9.300
TR	1.019	42.058	30.645	11.413	0,55	428
TS	1.918	1.023.405	828.559	194.846	4,68	4.260
Укупно		**26.899.894**	**13.943.118**	**12.956.776**		**137.046**

Укупно умањење броја логичких читања услед оптималне денормализације, према резултатима у табели 11, износи 12.956.776 (страница). Модел се свакако може додатно побољшати чвршћом спрегом са оптимизатором упита базе података [53], када се примењује на оптимизовање претрага. Међутим, за навигациони приступ релационом моделу, који типично подразумева приступ подацима преко кластерованих индекса над идентификаторима ентитета, софистицирана анализа планова извршавања упита на тако ниском нивоу детаља није неопходна.

Испитана је и корелација између процењених и измерених вредности добити коришћењем Пирсонове функције корелације (видети табелу 12). Мала вредност коефицијента корелације између предикције на бази бројања редова и измереног

броја редова јасно показује да бројање редова није довољно поуздана метрика. Међутим, не мора нужно да значи да ову метрику треба безусловно одбацити, као и то да не постоји контекст у којем процена заснована на бројању редова има добру корелацију са измереним показатељима перформанси. Са друге стране, бројање редова појачано физичким параметрима допринело је јакој линеарној корелацији између процењених добити (табела 8) и измерених промена времена одзива трансакција (табела 9), као и измерених промена логичких приступа страницама (табела 11).

ТАБЕЛА 12:
ПРИКАЗ КОРЕЛАЦИЈЕ ПРОЦЕЊЕНИХ И ИЗМЕРЕНИХ ПОКАЗАТЕЉА ПОБОЉШАЊА ПЕРФОРМАНСИ TPC-E РАДНОГ ОПТЕРЕЋЕЊА

	Мерене вредности	
	$\Delta T/T_N$	**$\Delta P/N$**
Софистицирани модел предикције	0.90	0.87
Предикција заснована на бројању редова	0.02	-0.10

Коначно, дата анализа квалитета предложеног оптимизационог модела показала је да предложени модел цене и добити, унапређен физичким коефицијентима модела података, даје предикцију која је практично употребљива.

7. Закључак

У оквиру овог истраживања проучаване су могућности примене техника денормализације релационог модела великих база података за повећање ефикасности приступа подацима коришћењем објектно оријентисаног апликативног софтвера (апликација).

Перформансе објектно оријентисаних апликација за приступ подацима у релационим базама података, развијеним коришћењем традиционалних метода објектно-релационог мапирања, драстично се погоршавају са порастом количине података у бази података и бројем трансакција које треба обрадити.

Проучавањем доступне релевантне научно-стручне литературе дат је преглед могућих узрока овог проблема, као и преглед постојећих приступа и принципа његовог решавања. Може се закључити да се наведени проблем јавља како у трансакционим (OLTP), тако и у аналитичким (OLAP) системима. Колико год да је принцип нормализације један од најважнијих принципа пројектовања релационог модела података, који је данас широко распрострањен у индустрији софтвера, јер обезбеђује семантички стабилан модел података који је отпоран на аномалије при упису, таква организација шеме у великим релационим базама података базираним на диску идентификује се као главни извор проблема лоших перформанси приступа подацима у њима. Примењена искључиво статички, без разматрања начина приступа подацима, нормална форма доводи то тога да се сложени и софистицирани захтеви за приступ подацима, издати на апликативном нивоу, трансформишу на обрасце приступа подацима на релационом нивоу који нису погодни за технологију блоковских секундарних меморија (дискова) на којима традиционалне релационе базе података данас почивају. Такви обрасци приступа подацима захтевају случајан приступ блоковима (енгл. *random block access*) на диску, као и приступ већем броју блокова због мање „концентрације“ релевантних података по блоку у односу на захтеве трансакција.

Постојећа решења приступају овом проблему на неколико начина: а) редукцијом броја захтева бази података и повећању процента релевантних података у резултатима извршавања упита добијених по једном обраћању, б)

додавањем секундарних приступних структура (индекса, материјализованих погледа) у релациони модел у складу са обрасцима приступа подацима добијених из радног оптерећења, и в) денормализацијом релационог модела података.

Денормализација је техника оптимизације приступа подацима у релационим базама података која је посебно експлоатисана у домену аналитичких (OLAP) система и своди се на (парцијално и циљано) одступање од нормалних форми у релационом моделу за потребе оптимизација сложених (и честих) аналитичких упита. Њена суштина је у повећању „концентрације“ релевантних података за потребе трансакција по јединици меморије и њиховој просторној кластеризацији (густом паковању) са циљем да се њихов пренос из секундарне меморије учини максимално ефикасним (са минималним потребним бројем улазно-излазних захтева). Позитивни ефекти денормализације на операције читања података (па чак и супериорност у односу на индексирање [51], [52]), потврђени су у доступној научној литератури. Међутим, процесу денормализације још увек недостаје формална систематична метода, због чега он углавном још увек зависи од људског фактора [15], и због тога има ограничене домете и примену.

Стога је ово истраживање било фокусирано на изналажење формалних основа за систематичну и ефикасну примену денормализације релационог модела података за оптимизацију приступа подацима у трансакционим (OLTP) системима на основу прикупљеног „знања“ о приступу подацима добијеног профилизацијом радног оптерећења.

У овој докторској дисертацији проучавана је искључиво техника денормализације базирана на увођењу редундансе у модел података у циљу оптимизовања операција читања, узимајући у обзир негативне ефекте по операције уписа, при чему се не нарушавају и не мењају постојећи односи између релација у нормализованом моделу (технике партиционисања нису ушле у оквире овог истраживања). Тако уведени редундантни подаци названи су оптимизације. Дефинисан је формалан модел цене и добити (енгл. *cost-benefit*) за процену позитивних ефеката оптимизација на операције читања и њихових негативних ефеката на операције уписа. На тој основи формулисан је и оптимизациони проблем за проналажење скупа оптимизација које дају максималну добит при читању уз минималну цену операција уписа за потребе одржавања

конзистентности редундантних података. Доказано је у овој тези да предложени оптимизациони проблем припада класи тешких NP-комплетних проблема и детаљно је описана његова редукција на проблем бинарног линеарног програмирања, такође из класе NP-комплетних оптимизационих проблема за чије решавање постоје квалитетне и ефикасне хеуристике.

Предложена метода оптимизације приступа подацима експериментално је анализирана коришћењем *de facto* стандардног теста перформанси за трансакционе системе и базе података, TPC-E. Тестови перформанси извршени су коришћењем система за управљање базама података SQL Server 2016. Аутоматским одређивањем и имплементацијом оптималне редундансе у релационом моделу TPC-E базе података, на основу унапред задатог радног оптерећења, прописаног спецификацијом теста TPC-E, време одзива трансакција у том радном оптерећењу поправљено је за 65% у односу на његово извршавање над референтном нормализованом TPC-E базом података. Исто тако, предикција односа цене и добити оптимизација била је у јакој позитивној корелацији са измереним вредностима времена одзива трансакција. Експериментално је још показано да је оптимална денормализација супериорна и у односу на потпуну денормализацију, која не разматра негативне ефекте примењених оптимизација на бази редундансе података.

Добијеним експерименталним резултатима потврђено је то да је предложена метода оптимизације приступа подацима у великим релационим базама података постављена на реалним основама, ефикасна и скалабилна, што отвара могућности и за њену озбиљнију примену у индустријској пракси.

У оквиру научно-истраживачког рада на овој докторској дисертацији остварени су следећи научни доприноси:

- Детаљан преглед и класификација постојећих проблема и решења у доступној литератури из домена оптимизација приступа подацима у релационим базама података;
- Систематизација и интеграција знања из домена оптимизација приступа подацима из доступне литературе;

- Формални теоретски модел проблема оптимизације приступа подацима заснован на денормализацији релационог модела увођењем редундантних података који елиминишу или ублажавају неефикасне обрасце приступа подацима у секундарној меморији;
- Формална дефиниција оптимизационог проблема за проналажење оптималне редундансе у релационом моделу података чиме се максимизирају њени позитивни ефекти на операције читања, уз плаћање минималне могуће додатне цене за одржавање конзистентности редундантних података;
- Формални доказ сложености предложеног оптимизационог проблема који припада класи NP-комплетних проблема;
- Детаљан опис редукције предложеног NP-комплетног оптимизационог проблема на бинарно линеарно програмирање.

Поред наведених научних доприноса, у склопу рада на овом истраживању додатно је још урађено:

- Имплементација профајлера за детекцију навигационих образаца приступа подацима из трагова извршавања објектно оријентисаних апликација;
- Имплементација редукције предложеног оптимизационог проблема на бинарно линеарно програмирање;
- Имплементација релевантних артефаката за TPC-E тест перформанси према важећој спецификацији (имплементација трансакција, проширење генератора трансакција);
- Експериментална анализа предложене методе оптимизације приступа у великим релационим базама података и њене потенцијалне примене у индустријској пракси.

7.1. Идеје и правци даљег развоја предложене методе

У току истраживања уочене су и неке додатне могућности за унапређење предложене методе, које би проширили оквире њене применљивости, како на

традиционалне проблеме у инжењерству софтвера тако и на актуелне проблеме у области великих података.

1) Побољшање и надоградња предложеног модела цене и добити (енгл. *cost-benefit*) за потребе прецизније оптимизације операција претраживања (енгл. *queries*) на основу анализе планова извршавања упита и чвршћој спрези са оптимизатором упита (енгл. *query plan optimizer*);

2) Предложена метода не укључује формално моделовање утицаја оптимизација на ограничења меморијског простора. Због своје инхерентне сложености, проблем је декомпонован на оптимизацију временске и оптимизацију просторне компоненте. Фокус истраживања био је на оптимизацији времена одзива приступа подацима, будући да је време данас критичнији ресурс од простора. Упркос томе, у постојећи формални модел уграђене су одговарајуће хеуристике за спречавање „експлозије“ простора, док је разматрање предикције потребног простора остало ван оквира ове тезе.

3) Разрада и експериментална анализа предложене методе за примену у домену система дистрибуираних података. Проблем дистрибуиране обраде података данас је актуелан више него икад до сада у историји рачунарства. Основна идеја у домену обраде дистрибуираних великих података данас почива на чињеници да је јефтиније пренети „израчунавање“ него (велике) податке. Уколико потреба за преносом израчунавања временом драстично порасте, због нарастајућих потреба софистицираности обраде података, комуникација и мрежна инфраструктура може да постане уско грло. Оптимално распоређивање података (енгл. *data placement*) и размештање израчунавања јесте домен у којем предложена метода, са додатним проширењима и разрадом, може наћи своју примену и то је један од следећих корака у наставку овог истраживања.

Литература

[1] P. Kettunen и M. Laanti, „Future software organizations – agile goals and roles,“ *European Journal of Futures Research,* т. 5, бр. 1, pp. 2195-2248, 2017.

[2] K. Schwab, „The fourth industrial revolution. Currency,“ 2017.

[3] D. L. Parnas, „On the criteria to be used in decomposing systems into modules,“ *Communications of the ACM,* т. 15, бр. 12, pp. 1053-1058, 1972.

[4] A. Olivé, Conceptual Modeling of Information Systems, New York, Inc., Secaucus, NJ, USA: Springer-Verlag, 2007.

[5] D. Milicev, Model-driven development with executable UML, John Wiley & Sons, 2009.

[6] P. A. Bernstein, S. Pal и D. Shutt, „Context-based prefetch–an optimization for implementing objects on relations,“ *The VLDB Journal—The International Journal on Very Large Data Bases,* т. 9, бр. 3, pp. 177-189, 2000.

[7] J. Gray и A. Reuter, Transaction processing: concepts and techniques, Elsevier, 1992.

[8] N. Leavitt, „Whatever happened to object-oriented databases?,“ *IEEE Computer,* бр. 8, pp. 16-19, 2000.

[9] B. Baesens, A. Backiel и S. v. Broucke, „The state of database access in Java: Passchendaele revisited,“ Cutter Consortium, Research Center for Management Informatics (LIRIS), Leuven, 2015.

[10] E. F. Codd, „A relational model of data for large shared data banks,“ *Communications of the ACM,* т. 13, бр. 6, pp. 377-387, 1970.

[11] R. Elmasri и S. Navathe, Fundamentals of database systems, Pearson, 2017.

[12] S. Navathe, S. Ceri, G. Wiederhold и J. Dou, „Vertical partitioning algorithms for database design,“ *ACM Transactions on Database Systems (TODS),* т. 4, pp. 680-710, 1984.

[13] E. F. Codd, The relational model for database management: version 2, Addison-Wesley Longman Publishing Co. Inc., 1990.

[14] E. F. Codd, „Extending the database relational model to capture more meaning,“ *ACM Transactions on Database Systems (TODS),* т. 4, бр. 4, pp. 397-434, 1979.

[15] S. K. Shin и G. L. Sanders, „Denormalization strategies for data retrieval from data warehouses.,“ *Decision Support Systems,* т. 42, бр. 1, pp. 267-282, 2006.

[16] H. Lee, „Justifying database normalization: a cost/benefit model,“ *Information processing & management,* т. 31, бр. 1, pp. 59-67, 1995.

[17] V. V. Khodorovskii, „On normalization of relations in relational databases,“ *Programming and Computer Software,* т. 28, бр. 1, pp. 41-52, 2002.

[18] J. W. Rahayu, E. Chang, T. S. Dillon и D. Taniar, „Performance evaluation of the object-relational transformation methodology,“ *Data & Knowledge Engineering,* т. 38, бр. 3, pp. 265-300, 2001.

[19] S. Agarwal, C. Keene и K. M. Arthur, „Architecting object applications for high performance with relational databases,“ у *OOPSLA Workshop on Object Database Behaviour, Benchmarks, and Performance*, Austin, 1995.

[20] N. Kojić и D. Milićev, „A survey of object-relational transformation patterns for high-performance UML-based applications,“ у *International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, Anger, 2015.

[21] S. Mostarda, M. D. Sanctis и D. Bochicchio, Entity Framework 4 in Action, Manning Publications Co., 2011.

[22] G. Booch, J. Rumbaugh и a. I. Jacobson, Unified Modeling Language User Guide (The 2nd Edition), Addison-Wesley Object Technology Series, 2005.

[23] „Entity Framework,“ Microsoft Corporation, [На мрежи]. Available: https://docs.microsoft.com/en-us/ef. [Последњи приступ 1 10 2019].

[24] „Enterprise Java Beans,“ Oracle Corporation, [На мрежи]. Available: https://www.oracle.com/technetwork/java/javaee/ejb/index.html. [Последњи приступ 1 10 2019].

[25] „Hibernate ORM,“ Red Hat, [На мрежи]. Available: https://hibernate.org. [Последњи приступ 1 10 2019].

[26] R. Buyya, R. N. Calheiros и a. A. V. Dastjerdi, Big data: principles and paradigms, Morgan Kaufmann, 2016.

[27] M. Chen, S. Mao и Y. Liu, „Big data: A survey,“ *Mobile networks and applications,* т. 19, бр. 2, pp. 171-209, 2014.

[28] „Java Language Specification,“ [На мрежи]. Available: https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf.

[29] D. Tsirogiannis, S. Harizopoulos, M. A. Shah, J. L. Wiener и G. Graefe, „Query processing techniques for solid state drives,“ у *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, 2009.

[30] J. Do, Y.-S. Kee, J. M. Patel, C. Park, K. Park и D. J. DeWitt, „Query processing on smart ssds: Opportunities and challenges,“ у *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013.

[31] T. Zäschke, S. Leone, T. Gmünder и M. C. Norrie, „Optimising Conceptual Data Models through Profiling in Object Databases,“ у *International Conference on Conceptual Modeling*, Berlin, 2013.

[32] A. Ibrahim и W. R. Cook, „Automatic prefetching by traversal profiling in object persistence architectures,“ у *ECOOP*, 2006.

[33] T.-H. Chen, „Improving the quality of large-scale database-centric software systems by analyzing database access code,“ у *Data Engineering Workshops (ICDEW), 2015 31st IEEE International Conference*, 2015.

[34] T.-H. Chen, W. Shang, Z. M. Jiang, A. E. Hassan, M. Nasser и P. Flora, „Detecting performance anti-patterns for applications developed using object-relational mapping,“ у *Proceedings of the 36th International Conference on Software Engineering*, Hyderabad, 2014.

[35] T.-H. Chen, W. Shang, Z. M. Jiang, A. E. Hassan, M. Nasser и P. Flora, „Finding and evaluating the performance impact of redundant data access for applications that are developed using object-relational mapping frameworks,“ *IEEE Transactions on Software Engineering*, т. 42, бр. 12, pp. 1148-1161, 2016.

[36] S. Guéhis, V. Goasdoué-Thion и a. P. Rigaux, „Speeding-up data-driven applications with program summaries,“ у *Proceedings of the 2009 International Database Engineering & Applications Symposium ACM*, Calabria, 2009.

[37] S. Agrawal, N. Vivek и Y. Beverly, „Integrating vertical and horizontal partitioning into automated physical database design,“ у *Proceedings of the 2004 ACM SIGMOD international conference on Management of Data*, 2004.

[38] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman и S. G. e. al., „Spanner: Google’s globally distributed database.,“ *ACM Transactions on Computer Systems (TOCS)*, т. 31, p. 8, 2013.

[39] J. Shute, R. Vingralek, B. Samwel, B. Handy, C. Whipkey, E. Rollins и M. O. e. al., „F1: A distributed SQL database that scales,“ у *Proceedings of the VLDB Endowment*, 2013.

[40] J. E. H. Han, G. Le и J. Du, „Survey on NoSQL database,“ у *Pervasive computing and applications (ICPCA)*, 2011.

[41] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes и R. E. Gruber, „Bigtable: a distributed storage system for structured data,“ *ACM Transactions on Computer Systems (TOCS) 26*, т. 26, бр. 2, pp. 4-18, 2008.

[42] https://cloud.google.com/spanner/docs/schema-and-data-model, „Schema and data model,“ Google, [На мрежи]. Available: https://cloud.google.com/spanner/docs/schema-and-data-model. [Последњи приступ 25 9 2019].

[43] M. Schkolnick и P. Sorenson, „Denormalization: a performance-oriented database design technique,“ у *Proceedings of the AICA*, Bologna, Italy, 1980.

[44] U. Rodgers, „Denormalization: why, what, and how?,“ *Database Programming & Design,* т. 12, p. 46–53, 1989.

[45] M. Hanus, „To normalize or denormalize, that is the question,“ у *Proceedings of 19th International Conference for the Management and Performance Evaluation of Enterprise Computing Systems*, San Diego, CA, 1994.

[46] P. Wegrzynowicz, „Performance antipatterns of one to many association in hibernate,“ у *Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2013.

[47] M. Berler, J. Eastman, D. Jordan, C. Russell, O. Schadow, T. Stanienda и F. Velez, The object data standard: ODMG 3.0, San Francisco: Morgan Kaufmann, 2000.

[48] S. Chaudhuri и V. R. Narasayya, „An efficient, cost-driven index selection tool for Microsoft SQL server,“ у *Proceedings of the 23rd VLDB Conference*, Athens, 1997.

[49] S. Chaudhuri и V. Narasayya, „AutoAdmin “what-if” index analysis utility,“ у *ACM SIGMOD Record*, Seattle, 1998.

[50] S. Chaudhuri и V. Narasayya, „Self-tuning database systems: a decade of progress,“ у *Proceedings of the 33rd international conference on Very large data bases*, Vienna, 2007.

[51] S. Agarawal, S. Chaudhuri и V. Narasayya, „Automated selection of materialized views and indexes for SQL databses,“ у *Proceedings of 26th International Conference on Very Large Databases*, Cairo, Egypt, 2000.

[52] R. Chirkova, A. Y. Halevy и S. Dan, „A formal perspective on the view selection problem,“ у *VLDB*, 2001.

[53] D. Dash, N. Polyzotis и A. Ailamaki, „CoPhy: a scalable, portable, and interactive index advisor for large workloads,“ *Proceedings of the VLDB Endowment 4,* т. 4, бр. 6, pp. 362-372, 2011.

[54] P. A. Bernstein, M. Jacob, J. Pérez, G. Rull и J. F. Terwilliger, „Incremental mapping compilation in an object-to-relational mapping system,“ у *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013.

[55] A. Boniewicz, P. Wisniewski и K. Stencel, „On redundant data for faster recursive querying via ORM systems,“ у *Computer Science and Information Systems (FedCSIS)*, Kraków, 2013.

[56] C. E. Dabrowski, D. K. Jefferson, J. V. Carlis и S. T. March, „Integrating a knowledge-based component into a physical database design system,“ *Information & Management,* т. 17, бр. 2, pp. 71-86, 1989.

[57] D. J. Abadi, P. A. Boncz и S. Harizopoulos, „Column-oriented database systems,“ у *Proceedings of the VLDB Endowment 2*, 2009.

[58] S. Harizopoulos, V. Liang, D. J. Abadi и S. Madden, „Performance tradeoffs in read-optimized databases,“ у *Proceedings of the 32nd international conference on Very large data bases*, 2006.

[59] D. J. Abadi, D. S. Myers, D. J. DeWitt и S. R. Madden, „Materialization strategies in a column-oriented DBMS,“ у *ICDE 2007. IEEE 23rd International Conference on Data Engineering*, 2007.

[60] „SOLoist,“ [На мрежи]. Available: https://www.soloist4uml.com/. [Последњи приступ 24 08 2019].

[61] „Gurobi Solver,“ Guroby Optimization, [На мрежи]. Available: www.gurobi.com.

[62] R. M. Karp, „Reducibility among combinatorial problems,“ *50 Years of Integer Programming 1958-2008,* pp. 219-241, 2010.

[63] TPC, „TPCE,“ 2010. [На мрежи]. Available: http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-e_v1.14.0.pdf.

[64] „SQL 2000,“ Microsoft Corporation, [На мрежи]. Available: https://download.microsoft.com/download/5/4/A/54AFD350-6477-4910-9DF2-4C472C906684/SQL2000_release.pdf. [Последњи приступ 29 09 2019].

Списак слика

Списак табела

Списак листинга

Биографија аутора

Немања Којић, мастер инжењер електротехнике и рачунарства, рођен је 22. септембра 1984. у Лозници, Република Србија. Гимназију „Вук Караџић“ у Малом Зворнику, општи смер, завршио је као носилац Вукове дипломе.

Основне академске студије на Одсеку за рачунарску технику и информатику Електротехничког факултета Универзитета у Београду уписао је 2003. године. Дипломирао је 29. фебруара 2008. године са просечном оценом 9,60 и оценом 10 на дипломском раду на тему „Интелигентни систем за праћење присуства на послу“. У току основних студија обављао је летњу праксу у предузећу "СОЛ софтвер" 2005. године, био је полазник стручних курсева компаније Мајкрософт под окриљем Рачунског центра у Београду, а током 2007. године боравио је на стручној пракси на Институту за технологије у Тантојуки, Веракруз, Мексико.

Након дипломирања, Немања Којић био је запослен на позицији програмера у предузећу "СОЛ софтвер" где се бавио иновативним методама развоја информационих система, заснованим на извршивим моделима, у оквиру неколико интернационалних пројеката током 2008. и 2009. године.

Дипломске академске студије – мастер уписао је 2008. године на Електротехничком факултету Универзитета у Београду, модул Рачунарска техника и информатика. Мастер студије је завршио 18. фебруара 2010. са просечном оценом 9,60 и оценом 10 на завршном раду на тему „Перзистенција UML објектног простора у релационим базама података за вишеслојне и скалабилне веб архитектуре". Докторске студије уписао је 2010. године на Електротехничком факултету Универзитета у Београду на модулу Софтверско инжењерство и бавио се темом оптимизације приступа подацима у великим базама података.

Од 2010. до 2018. године био је запослен као асистент у настави при Катедри за рачунарску технику и информатику на Електротехничком факултету Универзитета у Београду. У звање сарадника изабран је 22.12.2009. године, затим и поново изабран у исто звање 11.01.2011. У звање асистента у настави за ужу научну област Рачунарска техника и информатика изабран је 6. септембра 2011. године, а затим и поново изабран у исто звање 1. јануара 2015. године. Сарађивао

је у настави на укупно 8 различитих предмета на основним академским и мастер студијама.

Током докторских студија био је ангажован на више пројеката у државним и међународним институцијама, као што су Министарство просвете, науке и технолошког развоја, Регистар националног интернет домена Србије, *United Nations Office for Project Services (UNOPS), World Health Organization (WHO).* Реализовао је и неколико комерцијалних решења у домену дигиталне форензике, складиштења великих података, а бавио се и анализом, пројектовањем и интеграцијом финансијских софтверских система.

У свом истраживачком раду показао је посебно интересовање за симулационе методе и оптимизације приступа подацима у великим базама података, а један краћи период бавио се и истраживањем из области аутоматског генерисања софтверских тестова, као и алгоритмима за анализу података у бежичним сензорским мрежама.

Током докторских студија Немања Којић објавио је као коаутор два рада у међународним часописима са SCI листе, од тога један рад у врхунском међународном часопису, три рада на међународним конференцијама, три рада на домаћим конференцијама, као и један рад у домаћем часопису.

СПИСАК РАДОВА АУТОРА

Научни радову у међународним часописима међународног значаја – М20:

1) **Kojic, N.**, Milicev, D. (2019). **"Equilibrium of Redundancy in Relational Model for Optimized Data Retrieval"**, *IEEE Transactions on Knowledge and Data Engineering.* [**M21, IF=3,857**], DOI: 10.1109/TKDE.2019.2911580, Print ISSN: 1041-4347, Electronic ISSN: 1558-2191

2) Vujičić-Stanković, S., **Kojić, N.**, Rakočević, G., Vitas, D., & Milutinović, V. (2013). **"A Classification of Data Mining Algorithms for Wireless Sensor Networks, and Classification Extension to Concept Modeling in System of Wireless Sensor Networks Based on Natural Language Processing"**, *Advances in Computers*, 90, 223-283, [M23, IF=0,489], DOI: 10.1016/B978-0-12-408091-1.00004-X, ISSN 0065-2458.

Научни радови у зборницима међународних научних скупова – М30:

3) Stankovic V.S., Rakocevic G., **Kojic N.**, Milicev D., "**A Classification and Comparison of Data Mining Algorithms for Wireless Sensor Networks**", *Proceedings of IEEE ICIT*, March 2012, Athens, Greece, [**M33**]

4) S. Žitnik, L. Šubelj, M. Janković, B. Furlan, D. Drašković, **N. Kojić**, M. Mišić, M. Bajec, **"Iterative End-to-end Information Extraction based on Linear Models"**, *Proceedings of the 22nd International Electrotechnical and Computer Science Conference ERK 2013*, pp. B47-B50, IEEE Slovenian Section, Portorož, Slovenia, Sep, 2013, [**M33**]

5) **N. Kojić**, D. Milićev, **"A Survey of Object-Relational Transformation Patterns for High-performance UML-based Applications"**, *International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2015)*, Angers, France, Feb, 2015, [**M33**]

Научни радови у часописима националног значаја – М50:

6) Milićev D i **Kojić N.**, **"Modelovanje sistema velikih razmera primenjeno u praksi: razvoj složenih poslovnih aplikacija pomoću izvršivog UML-a"**, *InfoM*, vol. 43, pp. 4-12, 2012, [M53]

Научни радови у зборницима скупова националног значаја – М60:

7) **Kojić N.**, Milićev D., **“Perzistencija UML objektnog prostora u relacionim bazama podataka za višeslojne veb arhitekture”**, *Zbornik radova konferencije YU Info*, pp. 265-270, Kopaonik, Srbija, 2012, [M63]

8) **Kojić N.**, Oklapi E., Drašković D., Protić J., **“Softverska realizacija jezgra za analitičku i simulacionu evaluaciju performansi računarskih sistema”**, *Zbornik radova konferencije YU Info*, pp. 548-553, Kopaonik, Srbija 2013, [M63]

9) Protić J., Marjanović A., Drašković D., **Kojić N.**, Romić U., Marković D., **“Sofverski aspekti procedure samovrednovanja Elektrotehničkog fakulteta u Beogradu”**, *Zbornik radova konferencije YU Info*, pp. 595-600, Kopaonik, Srbija, 2013, [**M63**]

A. Изјава о ауторству

Име и презиме аутора: **Немања Којић**

Број индекса: **2010/5043**

Изјављујем

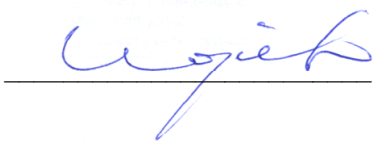
да је докторска дисертација под насловом

Оптимизација приступа подацима у објектно-релационом мапирању заснована на аутоматској денормализацији

- резултат сопственог истраживачког рада;
- да дисертација у целини ни у деловима није била предложена за стицање друге дипломе према студијским програмима других високошколских установа;
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио/ла интелектуалну својину других лица.

Потпис аутора

У Београду, 3. 10. 2019. год.



B. Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора: **Немања Којић**

Број индекса: **2010/5043**

Студијски програм: **Електротехника и рачунарство**

Наслов рада: **Оптимизација приступа подацима у објектно-релационом мапирању заснована на аутоматској денормализацији**

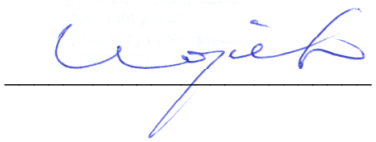
Ментори: **др Драган Милићев, редовни професор**

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла ради похрањена у **Дигиталном репозиторијуму Универзитета у Београду.** Дозвољавам да се објаве моји лични подаци везани за добијање академског назива доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис аутора

У Београду, 3. 10. 2019. год.



C. Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Оптимизација приступа подацима у објектно-релационом мапирању заснована на аутоматској денормализацији

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигиталном репозиторијуму Универзитета у Београду и доступну у отвореном приступу могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство (CC BY)

2. Ауторство – некомерцијално (CC BY-NC)

3. Ауторство – некомерцијално – без прерада (CC BY-NC-ND)

(4.) **Ауторство – некомерцијално – делити под истим условима (CC BY-NC-SA)**

5. Ауторство – без прерада (CC BY-ND)

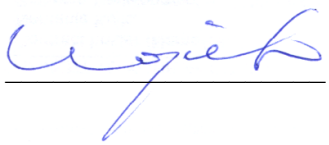
6. Ауторство – делити под истим условима (CC BY-SA)

(Молимо да заокружите само једну од шест понуђених лиценци.

Кратак опис лиценци је саставни део ове изјаве).

Потпис аутора

У Београду, 3. 10. 2019. год.



1. **Ауторство**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце, чак и у комерцијалне сврхе. Ово је најслободнија од свих лиценци.

2. **Ауторство – некомерцијално**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела.

3. **Ауторство – некомерцијално – без прерада**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца не дозвољава комерцијалну употребу дела. У односу на све остале лиценце, овом лиценцом се ограничава највећи обим права коришћења дела.

4. **Ауторство – некомерцијално – делити под истим условима**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца не дозвољава комерцијалну употребу дела и прерада.

5. **Ауторство – без прерада**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, без промена, преобликовања или употребе дела у свом делу, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце. Ова лиценца дозвољава комерцијалну употребу дела.

6. **Ауторство – делити под истим условима**. Дозвољавате умножавање, дистрибуцију и јавно саопштавање дела, и прераде, ако се наведе име аутора на начин одређен од стране аутора или даваоца лиценце и ако се прерада дистрибуира под истом или сличном лиценцом. Ова лиценца дозвољава комерцијалну употребу дела и прерада. Слична је софтверским лиценцама, односно лиценцама отвореног кода.