УНИВЕРЗИТЕТ У БЕОГРАДУ
ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Милош М. Цветановић

СИСТЕМ ЗА ИНТЕРАКТИВНУ ПРОВЕРУ СЛИЧНОСТИ КОНЦЕПТУАЛНИХ И ЛОГИЧКИХ МОДЕЛА РЕЛАЦИОНИХ БАЗА ПОДАТАКА

докторска дисертација

Београд, 2012

UNIVERSITY OF BELGRADE
SCHOOL OF ELECTRICAL ENGINEERING

Miloš M. Cvetanović

A SYSTEM FOR INTERACTIVE SIMILARITY COMPARISON OF CONCEPTUAL AND LOGICAL RELATIONAL DATABASE MODELS

DOCTORAL DISSERTATION

Belgrade, 2012

Ментор

др Мирослав Бојовић, ванредни професор

Електротехнички факултет Универзитета у Београду

Чланови комисије

др Мирослав Бојовић, ванредни професор

Електротехнички факултет Универзитета у Београду

др Владимир Благојевић, ванредни професор

Електротехнички факултет Универзитета у Београду

др Јован Ђорђевић, редовни професор

Електротехнички факултет Универзитета у Београду

др Душан Старчевић, редовни професор

Факултет организационих наука Универзитета у Београду

Датум одбране: 04.06.2012.

Изјава захвалности

Желео бих да изразим велику захвалност свима који су ми помогли да истражим и реализујем идеје описане у овој дисертацији, било да су ми давали савете или ме охрабривали и подржавали мој рад на многе друге начине.

Професору Мирославу Бојовићу, ментору, који је ме је увео у свет база података и пружио прилику да спроведем у дело моје истраживање у току рада на дисертацији. Професору Владимиру Благојевићу од кога сам научио како да дизајн база података посматрам увек из неког новог угла и како да сваком изазову приступам стрпљиво и аналитично.

Професору Јовану Ђорђевићу чија су ми помоћ и корисни савети како да изразим своје мисли на јасан и концизан начин много значили у припреми мог првог рада, за часопис од међународног значаја, у коме су објављени резултати најбитнијих делова ове дисертације.

Желео бих да се захвалим свим члановима катедре за рачунарску технику и информатику, на Електротехничком факултету у Београду, на разумевању уколико неку од својих обавеза нисам обавио у планираном року због своје ангажованости на дисертацији. Посебно, професору Вељку Милутиновићу са којим сам написао своје прве истраживачке радове и блиском пријатељу професору Бошку Николићу који је увек имао користан савет за мене и

разумевање када нисам стизао да дам свој максимални допринос у Рачунском центру. Јако сам захвалан и блиском пријатељу и колеги Саши Стојановићу који је увек имао разумевање за мене и увек довољно снаге да преузме огроман део посла у нашим заједничким пословним изазовима.

Такође, захваљујем се и студентима Николи Арсићу, Вељку Зрнићу, Дејану Стојнићу и Милошу Поповићу на помоћи при имплементацији прототипа система.

Међу особама које имају посебно место у мом животу, неизмерну захвалност дугујем великом пријатељу и колеги Захарију Радивојевићу, који је био присутан у сваком аспекту мог истраживања. Заиста је право задовољство и привилегија блиско сарађивати са њим. У раду са њим успели смо да чак и тешке истраживачке изазове претворимо у забаву и ужитак.

Такође, захваљујем се и Владимиру Блажићу који је много допринео да сваки тренутак који сам провео са породицом у току рада на дисертацији буде много лепши и забавнији.

Специјално сам захвалан и мојој девојци Бојани. Њена љубав ме константно инспирише, даје снагу и буди моју креативност.

Ову дисертацију посвећујем родитељима, Душанки и Миодрагу, и брату Андреји, који су прави извор свега лепог што ми се икада догодило у животу.

Доприноси свих споменутих особа учинили су да мој рад на дисертацији буде права награда и незаборавно искуство. Захваљујем се свима.

Наслов

Систем за интерактивну проверу сличности концептуалних и логичких модела релационих база података

Резиме

Развој концептуалног и логичког модела представља основне кораке у дизајну релационе базе података. Поступак дизајна, иако методолошки добро дефинисан, веома често у пракси полази од двосмислене и непотпуне спецификације захтева. Резултат тога је да не постоји једно најбоље решење, већ често неколико различитих модела задовољава исте захтеве. Из тог разлога, образовање студената, будућих инжењера, у области база података у великој мери зависи од практичног рада на коме би стекли потребно знање и вештине. Постојање система који би био подршка практичном раду, у виду лабораторијских вежби, било би од велике користи студентима и предавачима. Основни проблем у развоју једног таквог система јесте у веома захтевним поступцима провере исправности концептуалних и логичких модела.

Ова дисертација представља резултате истраживања и развоја новог система, названог ADVICE, који служи као подршка на курсевима из база података и помаже студентима да савладају разлике између теорије и праксе. У раду су такође описани нови поступци провере сличности концептуалних и логичких модела, развијени за потребе ADVICE система. Употреба система је представљена кроз скуп лабораторијских вежби развијених тако да помогну студентима да савладају концептуално и логичко моделовање, SQL, формалне упитне језике и нормализацију. Користећи ADVICE, студенти могу да приступају базама података,

решавају конкретне проблеме, а систем им пружа подршку у виду порука о њиховим решењима. Из перспективе предавача, систем омогућава лаку организацију лабораторијских вежби и континуирано праћење напретка студената. У раду су такође описана практична искуства приликом употреба система ADVICE на курсу из база података у току трогодишњег периода.

Кључне речи

Дизајн база података, модели података, упитни језици база података, формална анализа концепата, образовање у рачунарским наукама, образована технологија.

Научна област

Електротехника и рачунарство

Ужа научна област

Рачунарска техника и информатика

УДК

621.3:004.65

Title

A System for Interactive Similarity Comparison of Conceptual and Logical Relational Database Models

Abstract

Development of conceptual and logical models represents basic design activities for a relational database. Design process in practice, although methodologically thoroughly defined, very often starts from an ambiguous and incomplete specification of requirements. For such specification, there is no one best solution, but often several different models meet the same requirements. That is the reason why, the education of students, future engineers in the field of databases, depends largely on a practical work on which they acquire the necessary knowledge and skills. The existence of a system that would be a support for practical work, in the form of laboratory exercises, would be of great benefit to both students and lecturers. The core problem in developing the system is that it depends on very demanding procedures needed for similarity comparison of both conceptual and logical models.

The dissertation presents the results of a research that led to the development of a new system, named ADVICE, which can be used as a support on database courses to help students to bridge the gap between database management theory and practice. The dissertation also defines the new procedures, specifically developed for the ADVICE system, that are used for similarity comparison of both conceptual and logical database models. The usage of the system is presented through a set of laboratory exercises developed to teach students conceptual and logical modeling, SQL, formal query languages, and normalization. While working on the exercises, students use the system

to access real databases, and the system provides them with feedback about their solutions. From the perspective of an instructor, the system allows easy exercise management and continual progress monitoring. The dissertation also describes a practical experience with the use of ADVICE on a database course over a three-year period.

Keywords

Database Design, Data Models, Database Query Languages, Formal Conceptual Analysis, Computer Science Education, Educational Technology.

Scientific field

Electrical and Computer Engineering

Scientific subfield

Computer Engineering and Information Theory

UDC

621.3:004.65

Садржај

Листа слика

Листа табела

1. Увод

Базе података представљају колекцију сачуваних, међусобно повезаних података који служе потребама једног или већег броја корисника. Системи заслужни за функционисање база података су комплексни софтверски системи који се називају системи за управљање базама података (DBMS). Ови системи могу на различите начине организовати чување података, односно, могу бити засновани на различитим моделима података. Један од најзаступљенијих начина јесте релациони модел података. У случају релационих база података, доминантан начин приступа и коришћења релационих база података заснива се на употреби стандардног структурног упитног језика (SQL). Користећи SQL корисник може манипулисати подацима у бази, односно дохватати их и мењати, али такође може и дефинисати структуре у којима се подаци чувају као и правила конзистенције која над њима важе.

Корисници база података се могу поделити у три веће групе корисника. Прву групу представљају инжењери који раде на развоју система за управљање базама података. Другу групу чине инжењери који развијају апликације које користе релационе базе података и администратори база података који се брину о свакодневном раду релационих база података. Трећа група су крајњи корисници

релационих база података, који базама података приступају користећи апликације, а веома ретко и директно подацима у бази података. Образовање овако разноврсних корисника доноси велики број изазова, али су два најизраженија. Први изазов јесте питање тема које ће бити обухваћене образовањем у области база података, а други је питање практичног рада током трајања образовања. Из тога проистиче проблем, како, у току што краћег времена, што успешније обрадити што већи број тема и све њих покрити одговарајућим практичним радом.

Међу курсевима посвећеним базама података, највећи проблеми се јављају управо на првим, уводним, курсевима, од којих се очекује адекватно покрију све оне теме које су заједничке за све врсте корисника. Најпре је потребно упознати студенте са релационим моделом, а потом објаснити поступак дизајна релационих база података. Дизајн релационе базе почиње прављењем концептуалног модела, као основног вида евидентирања захтева за чувањем података, односно захтева по питању који се подаци чувају и који су односи који међу њима треба да важе. Добијени концептуални модел се потом преводи у одговарајући логички модел, чиме се добијају релационе шеме, тј. основни елементи релационог модела. Квалитет добијених релационих шема се потврђује правилима нормализационе теорије. Потом се употребом SQL релационе шеме преводе у одговарајуће табеле, и при томе се дефинишу сва правила интегритета података која укључују и пословна правила која није било могуће представити у концептуалном моделу. Исправност добијених табела се коначно проверава дефинисањем упита којима ће се приступати подацима, односно дефинисањем исказа којим ће се обављати манипулација подацима.

Иако класичне методе учења у виду предавања и вежбања на табли, могу да буду довољне за упознавање са основним концептима дизајна, студенти ипак не могу стећи потребне вештине искључиво похађањем наставе. Када се у настави укључе и практичне лабораторијске вежбе, предавач своју пажњу расипа на читаву групу студената, а неминовност тога је да свако од њих добија само ограничену личну помоћ. Управо због, постојање система за подршку на лабораторијским вежбама, који би помогао студентима у стицању вештина потребних за дизајн база података, би био од велике користи. Основни проблем у развоју једног таквог система био би у томе што би требао да буде подршка процесу дизајна који, по својој природи, није прецизно дефинисан. Односно, веома често поставка проблема који се јављају код дизајна базе може да буде двосмислена и непотпуна, што представља реалну слику ситуација у пракси. Другим речима, не само да не постоји једно најбоље решење за задати проблем, већ и често неколико различитих модела задовољава исте захтеве.

Циљеви овог рада су дефинисање поступака провере сличности концептуалних и логичких модела релационих база података, као и употреба тих поступака у сврху имплементације система који би био подршка спровођењу практичних лабораторијских вежби на курсевима из база података. Такав систем би користио поступке провере сличности модела како би упоредио одговор студента са решењем које је професор дао. Провера сличности би одредила разлике међу моделима, а на основу тога би систем могао да студенту да укаже на грешке. Студент не би морао одмах да види тачно решење, већ би га систем, кроз низ итерација, наводио ка тачном решењу. Резултат употребе оваквог система, могао

би да буде бољи успех студената, с обзиром да би сваки студент употребом система самостално долази до решења.

У наставку рада биће презентовани резултати у домену провере сличности концептуалних и логичких модела релационих база података, као и конкретна примена тих резултата у виду система за интерактивну проверу сличности који се користи као подршка практичном раду на курсевима из база података. Најпре је у другој глави кроз детаље о релационим базама података, концептуалном и логичком моделовању, детаљније објашњен проблем и предлог решења. Након тога је у првом делу главе три, дат преглед постојећих решења у области провере сличности концептуалних модела, уз класификовање према аспекту на који се приступа проблему при решавању. У другом делу главе три, дат је предлог поступка провере сличности концептуалних модела који је употребљен у систему за интерактивну проверу сличности. Слично глави три, и глава четири је организована у два дела. У прво делу је дат преглед постојећих решења у области провере сличности логичких модела, уз класификовање према аспекту на који се приступа проблему при решавању, док је у другом делу те главе дат предлог поступка провере сличности логичких модела који је употребљен у системи за интерактивну проверу сличности. Глава пет је посвећена систему за интерактивну проверу сличности концептуалних и логичких модела. Глава пет објашњава мотивацију за развој система, даје преглед постојећих система и њихову компаративну анализа, даје детаље најбитнијих делова имплементације система односно као и детаље употребе система на примеру конкретних лабораторијских вежби како би се демонстрирале функционалности система и његова употребљивост. Глава пет такође садржи и квалитативну и квантитативну

евалуацију резултата употребе система. Закључак рада је у глави шест, којом се овај ради завршава.

2. Дефиниција проблема и предлог решења

У овој глави ће најпре бити представљене релационе базе података и њихов настанак. Потом ће бити објашњени поступци концептуалног и логичких моделовања релационих база података. Након тога ће у делу посвећеном провери сличности модела релационих база података бити дефинисан проблем и предлог решења који представљају основу овог рада. На крају главе су дати критеријуми који се могу користити при компаративној анализи различитих приступа провери сличности модела релационих база података.

2.1. Релационе базе података

Прва појава термина база података подудара се са развојем медија за чување података са директним приступом, средином шездесетих година двадесетог века. Термин је представљао алтернативу тада употребљаваним системима заснованим на тракама који су се углавном користили за пакетску обраду, док су базе података требале да омогуће интерактивнији рад корисника. Системи заслужни за функционисање база података су комплексни софтверски системи који се називају системи за управљање базама података (DBMS). Код најранијих система, перформансе су вероватно била примарна брига, међутим већ тада је препознато да постоје и други битне карактеристике о којима треба бринути.

Прве генерације система за управљање базама података биле су навигационог типа код којих се подацима приступало пратећи показиваче са једног записа на други. Два најзаступљенија модела података, на којима су базе података тада биле засноване, су били хијерархијски модел и мрежни модел. Седамдесетих година двадесетог века, појавио се релациони модел, који се удаљио од дотадашње праксе, инсистирајући на томе да се претрага података у базама податак треба обављати на основу садржаја, а не пратећи навигационе везе. Због своје захтевности по питању рачунарске снаге, релационо модел је преузео водећу позицију средином деведесетих година и од тада остао на том месту изузев у неким специјализованом применама. Доминантни начин приступа и коришћења релационих база података заснива се на употреби стандардног структурног упитног језика (SQL).

С обзиром да релациони модел потенцира претрагу, а не навигацију, он не прави односе међу подацима у виду показивача, већ их репрезентује у виду вредности које по једнакости реферишу једне друге. Овај приступ је добра основа за упитне језике, али је дизајн базе података отежан. Управо је из тог разлога, дизајн релационих база података, представља сложен поступак који се састоји из већег броја корака. Дизајн релационе базе почиње прављењем концептуалног модела, као основног вида евидентирања захтева за чувањем података, односно захтева по питању који се подаци чувају и који су односи који међу њима треба да важе. Добијени концептуални модел се потом преводи у одговарајући логички модел, чиме се добијају релационе шеме, тј. основни елементи релационог модела. Елементи релационог модела ће бити детаљније објашњени у глави четири, док ће

детаљи концептуалног и логичког моделовања релационих база података бити дати у наставку који следи.

2.2. Концептуално и логичко моделовање релационих база података

Концептуални модел података је најуспешније средство за дизајн база података и комуникацију између дизајнера и крајњег корисника у току анализе захтева. Успех је последица чињенице да је концептуални модел лако разумети [1]. Други разлог за ефикасност овог модела је тај што се ради о приступу одозго-наниже, захваљујући концепту апстракције.

Дакле, користећи ентитете као апстракцију за елемента података и фокусирајући се на односе између ентитета у великој мери се смањује број објеката у разматрању чиме се поједностављује анализа. Број ентитета у бази података је обично далеко мањи од броја појединачних елемената, јер елементи обично представљају атрибуте.

Иако је, и код концептуалног модела, још увек потребно представити све елементе, далеко их је лакше анализирати, с обзиром да се они групишу у виду атрибута неког од ентитета на концептуалном нивоу. Везе између појединих атрибута се обично завршавају унутар самог ентитета, тј. оба атрибута припадају истом ентитету. У неким случајевима, могу постојати везе између атрибута различитих ентитета, али, чак и тада, најчешће постоји директна веза између ентитета којима ти атрибути припадају.

Концептуално моделовање почиње класификацијом ентитета и атрибута, иза чега следи идентификација хијерархија, а завршава се дефинисањем односа. Ентитета

треба да садрже описне информације, и атрибути треба увек да буду придружени ентитетима којима су најблискији. Вишевредносни атрибути би требали да буду класификовани као ентитети. Идентификација хијерахије представља напредну апстракцију, и у неким нотацијама чак није ни присутна у виду засебног графичког елемента.

Хијерархија омогућава нови степен апстракције, у смислу генерализације, чиме модел добија на дуговечности. Модел са хијерархијама ентитета, је флексибилнији и лакше одговара касним промена корисничких захтева. Међутим, флексибилност има своју цену, и то се најчешће огледа у повећању потребног времена за развој апликација које користе базу података са превише хијерархијских односа.

Односи између ентитета могу бити такви да представљају однос између два или више ентитета. У случају односа са више од два ентитета, однос не утиче на структуру учесника. Међутим, у случају односа са тачно два учесника, тзв. бинарних односа, може, а не мора, доћи до промене структуре једног од ентитета учесника. Та ситуација настаје када однос између два ентитета представља однос између надређеног и подређеног. Због честе потребе за бинарним односима, нотације које су употреби за концептуалне моделе, обично разликују ниво условљавања код бинарних односа, тако да се разликује егзистенцијална и идентификациона зависност.

Након што је концептуални модел направљен, а потом и ефективно прегледан и потврђен, следи корак логичког моделовања. Логичко моделовање треба да преведе концептуални модел, у одговарајућу форму нижег нивоа апстракције. У

случају релационих база података, спроводи се трансформација у одговарајуће релационе шеме, и потом и у адекватну имплементацију тих релационих шема у виду SQL табела.

Формирање логичког модела доноси две битне погодности. Прва погодност проистиче из чињенице да он настаје на основу концептуалног модела, користећи прецизно дефинисана правила превођења. Тиме се постиже да логички модел осликава све информације о пословним захтевима, без опасности да се нешто изостави. Такође, логички модел поред односа међу атрибутима који су дефинисани на концептуалном моделу, може да садржи и додатне односе, попут информација о функцијским зависностима, али и другим ограничењима, што значи да је квалитет податак у бази података на вишем нивоу.

Друга погодност, која проистиче из употребе логичког модела, јесте да он омогућава ниво апстракције који гарантује да у случају преношења базе података са једног система за управљање базама података на други, неће бити потребе за поновним дизајном од самог почетка, већ ће бити поновљен део који се односи на физичко моделовање.

Процес превођења концептуалног у логички модел, се може представити као низ корака који почиње спецификацијом табела њихових атрибута за оне ентитета који нису у подређеном односу према другим ентитетима, односно представљају јаке ентитета. У њима се идентификују примарни кључеви. Након тога се обрађују хијерахије ентитета. Хијерархије могу бити трансформисане на више начина, а конкретан начин зависи од одлука о жељеном нивоу апстракције. Тако се на пример, над-ентитет и сви под-ентитети могу представити једном табелом, а

такође као друга крајност јесте да сваки од ентитета постане засебна табела. Потом се прелази на спецификацију табела слабих ентитета, тј. ентитета који јесу у подређеном односу са бар једним од осталих ентитета. Овом приликом се прати ток зависности, тако да се најпре обрађују ентитети који зависе искључиво од јаких ентитета. Код слабих ентитета се прво обави спецификација основних атрибута, а потом спецификација примарног кључа. Приликом спецификације примарног кључа разрешавају се односи идентификационих зависности. Потом се поступак наставља спецификацијом страних кључева и разрешавањем свих егзистенцијалних зависности. На крају се разрешавају односи веза. Везе резултују у настанку нових табела, код којих спецификација примарних кључева зависи од спецификације кардиналности дефинисане на концептуалном нивоу. Крајњи резултат трансформације концептуалног у логички модел јесте шема базе података са дефинисаним табелама, атрибутима и ограничењима међу њима.

2.3. Провера сличности модела релационих база података

Проблем упаривања различитих модела података је пре свега интересантан у области интеграције шема и интеграцији рачунарских система, где се за независно развијене шеме конструише глобални поглед ради формирања јединствене онтологије [2]. Са развојем концепта складишта података, појавила се потреба за аутоматском трансформацијом података екстрахованих из оперативне базе података ради њиховог смештања у базу података намењену аналитичкој обради [3]. Слично, са развојем електронског пословања, појавила се потреба за упаривањем порука које системи размењују, при чему су поруке засноване на различитим XML шемама [4]. Упаривање XML шемама има велики значај и за

генералније случајеве, на пример, за потребе интеграције рачунарских система [5, 6].

У случају релационих модела података, поступак одређивања сличности треба да одреди мапирање појединих елемената једног модела на одговарајуће елементе другог модела података. Одређивање сличности и мапирање елемената једног релационог модела података на одговарајуће елементе другог релационог модела података у општем случају представља НП-потпун проблем. Из тог разлога, најбољи комерцијално доступни алати за упоређивање релационих модела података су пре свега графички програмски алати. Прецизније, ти алати омогућавају специфицирање мапирања модела података који се разматрају са циљем дефинисања процедура попуњавања релационе базе података или извршавања одређеног упита чиме се избегава ручно писање програмског кода, међутим ти алати не нуде никакав ниво аутоматизма приликом самог поступка мапирања. Овакво, ручно мапирање, захтева доста времена и подложно је грешкама.

Претходна истраживања су показала да је приликом упоређивања релационих модела података могуће посматрати само релационе шеме тих модела чиме се упоређују њихови концептуални модели. У случајевима када су доступне, поред релационих шема користе се и саме релације над тим шемама, односно подаци у одговарајућим базама података. Од информација које могу бити доступне, а представљају битан аспект релационих база података, јесте скуп поступака употребе података. Уколико се приликом упоређивања неких релационих база података жели покривање и тог манипулативног аспекта онда је поред

концептуалних модела потребно упоређивати и логичке моделе тих релационих база података.

2.3.1. Дефиниција проблема

Досадашња истраживања нису посветила пажњу посебној групи проблема мапирања модела података који су настали на основу идентичних корисничких захтева. У тим ситуацијама простор проблема мапирања модела је донекле редукован јер је могуће искористити претпоставке по питању именовања и употребљаване терминологије. Ситуације у којима постоје различити модели података настали на основу истих захтева су пре свега присутни у системима који треба да задовоље одређене унапред дефинисане стандарде. Интеграција софтверских система, нарочито у смислу вертикалне интеграције, је управо пример када се на основу исте спецификације може захтевати размена података који се иначе чувају у релационим моделима података [7, 8]. С обзиром да се интеграција може заснивати на принципима реверзног инжењерства, то би могло да значи и то да се резултати могу у некој каснијој фази користити и за потребе екстракције података и откривање информација [9].

Проблем провере сличности модела релационих база података у случају када су модели креирани на основу истих захтева ни на који начин није ограничен на домен интеграције софтверских система, већ супротно томе, присутан је у већем броју домена, на пример у домену образовања из области база података. Наиме, студенти на основу задатог проблема, предлажу решење, које је касније потребно упоредити са решењем које је наставник задао као референтно. Уобичајена пракса је да проверу исправности обавља предавач лично. Развој система који аутоматизује проверу сличности између одговора студента и решења које је дао

предавач захтева претходни развој поступака којима би се проверавала сличност. Постојање поступка за упоређивање студентских решења са референтним решењем омогућило би развој едукативног система који би омогућио већу самосталност у раду студентима, а на тај начин и већу ефикасност наставницима у свакодневном раду.

2.3.2. Предлог решења

Развој система који би се користио у образовању на курсевима из база података требао би да покрије теме које су од највећег значаја. Наиме, основни циљ на почетним курсевима из база података јесте савладавање концепата релационих база података и њиховог дизајна. То значи да би систем требао да омогући интерактиван рад студента и проверу исправности сваке од активности током дизајна.

Систем би најпре требао да подржи прављење концептуалних модела као прве активности у дизајну релационих база. Концептуални модел евидентира захтеве за чувањем података, односно захтеве по питању који се подаци чувају и који су односи који међу њима треба да важе. Систем би поред могућности прављења концептуалних модела, омогућио и проверу исправности која би се састојала у провери сличности између концептуалних модела, и то одговора студента, са једне стране, и решења предавача, са друге. С обзиром да између ових модела може постојати више разлика, систем би могао да студенту укаже на једну од грешака, уколико их је направио више. Тиме би се захтевало од студента да у складу са грешком на коју му је указано, покушава да модификује свој одговор. Ово би управо симулирало ситуацији када предавач лично помаже студенту да размишљањем и анализом своје грешке долази до новог предлога решења.

Поступак провере исправности би требао да посвети посебну пажњу честим проблемима које студенти праве у концептуалним моделима.

Након што је креиран концептуални модел, дизајн релационе базе се наставља, превођењем тог модела у одговарајући логички модел, чиме се добијају релационе шеме. Квалитет добијених релационих шема се потврђује правилима нормализационе теорије. Управо због тога, би систем требао да подржи проверу исправности спровођења нормализације. Алгоритми нормализације су прецизно дефинисани, тако да би се подршка система по питању провере сличности одговора и решења огледала у имплементацији одговарајућих алгоритама за декомпозицију до одговарајуће нормалне форме, као и осталих помоћних алгоритама. У случају да погреши студент би добио поруку од система, у која од његових релационих шема не задовољава жењени ниво квалитета, односно нормалну форму.

Логички модел, обухвата и имплементацију базе података употребом SQL. чиме се релационе шеме преводе у одговарајуће табеле, и при томе се дефинишу сва правила интегритета података која укључују и пословна правила која није било могуће представити у концептуалном моделу. Систем би требао да подржи проверу сличности имплементираних логичких модела. Имајући у виду да је структура модела већ проверена кроз проверу сличности концептуалног модела, провера сличности логичког модела би била фокусирана на проверу правила интегритета и пословних правила које није било могуће проверити провером сличности концептуалних модела. Студент би у случају да погреши требао да добије информацију о томе, који подаци не би смели да буду прихваћени његовим моделом.

У поступку дизајна, исправност добијених табела на логичком моделу се такође проверава дефинисањем упита којима ће се приступати подацима, односно дефинисањем исказа којим ће се обављати манипулација подацима. Самим тим би систем требао да омогући проверу сличности и овог аспекта логичког модела. Чак шта више, систем би требао да подржи задавање упита не само у виду SQL упита, већ и у виду исказа задатих формалним упитним језицима.

Како би систем био прилагођен потребама курсева посвећеним базама података, сви поступци провере сличности би требали да буду тако дефинисани да омогућавају проверу сваке од активности дизајна понаособ. Односно, систем треба да буде организован тако да свака активност у поступку дизајна може да буде независтан проблем који ће студент решавати. Интерактивност система би се састојала управо у употреби поступака за проверу сличности како би генерисао поруке које студенту указују на грешке и тиме га наводе ка тачном решењу.

Предложени поступци провере сличности концептуалних и логичких модела биће описани у трећој и четвртој глави, а њихова имплементација у систем који задовољава управо описане захтеве у глави пет. Пре тога, ће у наставку ове главе бити објашњени критеријуми који се могу користити у компаративним анализама различитих поступака провере сличности.

2.4. Критеријуми поступака за проверу сличности

Постоји велики број доступних прототипова решења за проверу сличности, и скоро сви објашњавају имплементационе детаље и дају неке експерименталне резултате, међутим сагледавање целокупне слике је тешко изводљиво. Један разлог је тај што је већина иако планирани да буду генерално примењива решења,

ипак развијена са неким специфичним претпоставкама у виду, па самим тим користи веома разнолике експерименте за процену ефикасности. Са друге стране, нека решења представљају композицију већег броја других решења, а у општем случају могу комбиновати стотине других, тако да није најјасније како их треба посматрати.

2.4.1. Димензије критеријума

Одабир критеријума за објективно упоређивање поступака за проверу сличности представља захтеван посао. Са једне стране, тешко је одабрати смислену карактеристику која би била применљива на сва постојећа решења, а са друге, лако је пропустити неку карактеристику битну за одређени подскуп решења. Међутим, постоје покушаји да се направи преглед приступа провери сличности попут [10]. Један од најбитнијих покушаја прављења прегледа експерименталних резултата разноврсних решења, дат је у [11]. Дефинисане су четири основне димензије за процену решења за проверу сличности:

- Улаз: врсте улазних информација које се користе приликом процеса провере сличности
- Излаз: информације које представљају додатак резултату провере сличности
- Квалитет: метрике одабране да се квантификује тачност и комплетност резултата провере.
- Напор: дефинише количину уштеђеног времена, односно ниво аутоматизације.

Свака од димензија биће детаљније размотрена у наставку, у складу са критеријумима од важности за компаративну анализу.

2.4.2. Критеријуми улаза

У домену провере сличности следеће информације су од важности приликом поређења различитих решења:

- Језик за опис шеме: улазне информације могу бити описане различитим језицима и форматима, нпр. XML, релациони модел, SQL, модел ентитета и односа, онтологија.
- Број шема приликом провере сличности: упоређивање двеју шема, нпр. изворне шеме и једне глобалне шеме, или провере међусобне сличности већег броја шема истовремено.
- Шема информације: најбитнији део се односи на број шема елемената доступних приликом провере. Веће улазне шеме, дефинишу већи простор претраге, што најчешће резултира лошијим квалитетом упарења. Са друге стране, уколико се приликом провере сличности искористи већа количина информација, али у смислу већег броја детаља, резултати провере могу бити бољи.
- Сличност шема: што су шеме које се пореде разноликије (тј. имају више шума у односу на корисне информације) утолико је проблем провере сличности захтевнији.
- Помоћне информације: доступност додатних информација, попут речника, ограничења и сл., може побољшати квалитет провере сличности.

2.4.3. Критеријуми излаза

Критеријуми који се односе на излаз, дефинишу да ли излаз поступка провере обезбеђује додатне информације на основу којих је могуће даље повећати квалитет резултата. Односно да ли провера дала већи број могућих решења за које корисник треба да одлучи који ће прихватити, а која одбацити. Мања количина додатних информација на излазу, обично резултира у мањој вероватноћи прављења грешке, али истовремено и у већем напору неопходном у анализи која следи након саме провере сличности:

- Кардиналност резултата: провера сличности као резултат може за сваки елемент шеме дати једно понуђено упарење, али исто тако може дати и већи број понуђених упарења.
- Ниво поверења: понуђена упарења могу бити на неки начин квантификована, нпр. одређеним нивоом сличности. Присуство информације о нивоу сличности омогућава кориснику лакше и поузданије доношење крајње одлуке, поготово у случају када је за сваки елемент понуђено више упарења.

2.4.4. Критеријуми квалитета

Процена квалитета резултата провере сличности дефинише се у односу на тачан резултат. Са тим у вези меру квалитета резултата дефинишу бројеви који говоре колико је било лажних и истинитих негатива и позитива. Лажни негативи (false negatives) представљају скуп елемента који нису аутоматски упарени тј. нису препознати као слични. Лажни позитиви (false positives) представљају скуп нетачно упарених елемената. Истинити позитиви (true positives) представљају

скуп тачно упарених елемената. Истинити негативи (true negatives) представљају скуп тачно неупарених елемената.

Постоје две уобичајено прихваћене мере квалитета, то су прецизност P (precision) и продорност R (recall). Прецизност описује однос тачних упарења према свим упарењима, док продорност описује однос тачних упарења према свим упарењима која су требала да буду пронађена. Уколико се ознакама A, B, C означе: A – true negatives, B – true positives, C – false positives, добија се да је

$$P = \frac{|B|}{|B| + |C|}$$

Односно:

$$R = \frac{|B|}{|B| + |A|}$$

У идеалном случају, када је скуп B максималан, а скупови A и C минимални, добија се да су прецизност и продорност једнаки 1. Међутим, ни прецизност ни продорност, самостално посматрани, не могу добро проценити квалитет упарења. На пример, може се догодити да је прецизност једнака 1, а да је продорност скоро једнака 0, што се догађа када постоји велики број не упарених елемената, а скоро да нема погрешно упарених. Управо из тог разлога постоји потреба да се дефинише јединствен критеријум који би обухватио и прецизност и продорност.

Постоји неколико предлога мера, које имају за циљ да обједине мере прецизности и продорности, а најзначајније су Ф-мера F (F-measure) и тачност A (accuracy, overall). Ф-мера користи тежински фактор α који је у опсегу [0, 1] и дефинише се као:

$$F(\alpha) = \frac{P \cdot R}{(1-\alpha) \cdot P + \alpha \cdot R}$$

Односно за случај када је α=0,5 добија се тзв. хармонијска Ф-мера:

$$F(\alpha) = 2 \cdot \frac{P \cdot R}{P + R}$$

Друго решење јесте комбинација прецизности и продорности која је названа тачност. Ова мера осликава идеју да се квантификује напор након процеса провере сличности потребан да се додају лажни негативи односно уклоне лажни позитиви:

$$\mathrm{A} = R \cdot \left(2 - \frac{1}{P}\right)$$

У случајевима када се обавља провера сличности већег броја шема истовремено, онда је значење скупова A, B, C може разликовати, међутим, генерално гледано семантика метрика остаје иста, с тим што се прецизност и продорност рачунају у односу на добро упарене моделе, односно могу се рачунати у односу на атрибуте који се појављују са одређеном фреквенцијом.

2.4.5. Критеријуми напора

Један од основних разлога за истраживање у области провере сличности, јесте смањивање обима посла приликом упаривања шема, самим тим процена потребног напора након примене алгоритама представља јако битан критеријум. Приликом процене потребног напора треба разликовати:

- Напор пре провере сличности: дефинисан је алгоритмом који се користи за процес провере сличности. Тако је на пример код провере засноване на

машинском учењу, неопходан тренинг система. Неки алгоритми за проверу захтевају детаљно конфигурисање разноврсних параметара попут прагова прихватања или тежинских коефицијената. Такође, неки алгоритми могу захтевати већу количину помоћних информација, попут речника, домен информација, ограничења итд.

- Напор после провере сличности: се пре свега односи на посао који корисник мора да уради пре коначног прихватања резултата. Конкретно, корисник треба да дода све лажне негативне резултате (погрешно елиминисане – false negatives), односно да уклони све лажне позитивне резултате (погрешно додате – false positives). Очигледно је да обим овог посла зависи доста и од кардиналности излаза процеса провере сличности.

На жалост, напор који особа треба да учини доста зависи од претходног знања које та особа поседује, сазнајних (когнитивних) способности те особе, нивоа упознатости те особе са конкретним проблемом, што све заједно доводи до тога да је јако тешко направити процену напора на један генерички начин.

3. Провера сличности концептуалних модела

Много тога је урађено у различитим областима, попут транслације и интеграције шема, представљања знања, машинског учења и приступа информацијама, са циљем да се аутоматизује поступак провере сличности шема. Главни циљ овог поглавља је преглед и класификација различитих приступа уз објашњење основних особина и могућности примене за сваки од њих. Критеријуми за класификацију су преузети из таксономије описане у [10]. Конкретно, дати су детаљи приступа који користе информације на нивоу шеме, на нивоу инстанци над шемама (подацима), помоћним информацијама, односно приступа који комбинују више различитих алгоритама за проверу сличности. Овакав преглед може да послужи не само при развоју нових приступа, већ и као генерална помоћ корисницима приликом одабира неког од постојећих решења [12].

3.1. Преглед постојећих решења и аспекти концептуалних модела

Приступи провери сличности се могу међусобно разликовати по питању улазних информација, начину обраде тих улазних информација и карактеристикама излазних резултата. Управо су ова три критеријума, која се могу сматрати

ортогоналним, узета за разликовање између појединих приступа које алгоритми за проверу сличности користе. На основу тих критеријума разликују се:

- Шема или инстанце: Провера сличности може да разматра информације на нивоу целокупне шеме (нпр. мета подаци као што су називи елемената, типови података, структурне особине) или информације на нивои инстанце (нпр. конкретни подаци).
- Елемент или структура: Алгоритам провере сличности може посматрати и упоређивати поједине елементе шеме (нпр. атрибуте) или комбинацију елемената који се заједно појављују у структури (нпр. шаблони).
- Језик или ограничења: Провера сличности може да користи лингвистички приступ (нпр. називе, текстуалне описе елемената), или ограничења дефинисана над шемом (нпр. ограничења попут типова података, ограничења јединствености, референцијалног интегритета).
- Са или без поновне употребе: Алгоритми за проверу сличности могу поред улазних шема и података над њима да користе и информације из помоћних извора (нпр. речника назива, глобалних шема, претходних резултата провере сличности, података које корисник уноси).
- Хибридни или композитни: У циљу веће применљивости и прецизности, алгоритам за проверу сличности може бити комбинација више појединачних приступа. Ово може бити постигнуто хибридним приступом (алгоритам је фиксна комбинација других алгоритама) или композитним (комбинују се резултати више других алгоритама).
- Једнострука или вишеструка кардиналност: Резултат провере сличности може бити да сваком елементу прве шеме одговара тачно један елемент

друге шеме (једнострука кардиналност) или више од једног елемента друге шеме (вишеструка кардиналност).

Већина наведених критеријума разматра улазне информације, док се не прави разлика по питању самог приступа за проверу сличности, односно на који начин се те улазне информације обрађују (обрада стрингова, машинско учење, итд.). Приступ провери сличности обично зависи од типа и карактеристика података који се обрађују. Слично, не прави се разлика по питању типа шема које се посматрају, односно да ли се ради о релационој, XML, објектно-орјентисаној или некој другој врсти шеме, као ни по питању интерне репрезентације шема, (листе, усмерени графови итд.) зато што алгоритми углавном зависе од типа информација које користе, а не од њихове репрезентације.

3.1.1. Аспект заснован на шеми

Шема засновани приступи разматрају само информације доступне у шеми. У зависности од изражајности употребљеног језика за дефинисање шеме, доступне информације могу да укључују разноврсне особине појединих елемената шеме, попут њиховог назива, описа, типа податка, ограничења, али такође и односа између појединих елемената, попут референцијалног интегритета, односа наслеђивања, или односа садржања. У наставку су најпре описани лингвистички (језички) приступ и приступ на основу ограничења, најчешћи приступи на нивоу елемената, који упоређују особине појединих елемената како би се одредило њихово поклапање. Након тога су размотрени приступи на нивоу структуре који користе односе између елемента како би се разматрало више елемената у исто време.

3.1.1.1. Лингвистички приступи

Приступи засновани на језику користе текстуалну обраду како би обрадили шема елементе и дошли до информација о називу и опису. Конкретно, називи елемената представљају најосновније конституанте шема и представљају први извор информација за процену сличности елемената. Сличност назива може бити процењена синтаксно упоређујући стрингове или семантички упоређујући њихова значења.

Синтаксна провера сличности назива израчунава се искључиво поредећи стрингове. Основни метод јесте одређивање идентичних назива, нпр. потпуно поклапање стрингова. У случају XML шема дефинисаних над истим простором имена, овај метод је довољан за одређивање парова одговарајућих елемената, с обзиром да простор имена гарантује јединственост назива. У осталим случајевима далеко је флексибилније користити приближну сличност стрингова, с обзиром да то омогућава одређивање сличности између назива и њихових скраћеница. Такви алгоритми су развијени и употребљују се и у другим областима, као што је обрада текста и аутоматска корекција правописних грешака (spelling corrections)[13], повезивање записа [14], уклапање секвенци у молекуларној биологији [16]. Неки од тих алгоритама, конкретно EditDistance, N-Gram и SoundEx алгоритми, су већ употребљени у области провере сличности шема, на пример у [16, 17, 18].

Алгоритам EditDistance користи технике динамичког програмирања, тако да се сличност стрингова израчунава на основу броја операција измена (убацивања, брисања, замена појединих карактера) неопходних за трансформацију једног стринга у други. Алгоритам N-Gram упоређује стрингове тако што се за сваки од њих израчунава скуп енграмова тј. секвенци од N карактера. У зависности од

вредности N, постоје различите варијације овог алгоритма, као што су Diagram (N=2), Trigram (N=3). На пример, скраћеница меди и реч медикамент имају следеће триграме {мед, еди} односно {мед, еди, дик, ика, кам, аме, мен, ент}, што значи да имају заједничка два триграма. SoundEx алгоритам посматра фонетску сличност између стрингова, односно између изговора тих стрингова. То се постиже израчунавањем одговарајућих SoundEx кодова. Ово може бити искоришћено када се жели провера да ли су два назива иста чак иако су другачије написани (тј. имају словну грешку). Ово је чест случај у језицима где не важи правило да једном гласу одговара једно слово, на пример у енглеском језику термини Licence, License и Licensing имају исте SoundEx кодове и према томе се може закључити да су слични.

Семантичка провера сличности, са друге стране, процењује сличност између стрингова на основу њиховог терминолошког односа, као што су синоними, хипоними и хипероними. На пример, речи дрво и стабло су синоними и због тога су слични. Односно, термин Плава је хипоним од термина Боја који представља хипероним. Овакав приступ захтева употребу помоћних извора, попут речника, онтологија или табела синонима које корисник обезбеди. Општи једнојезични или вишејезични речници, попут WordNet (http://wordnet.princeton.edu/), могу бити употребљени у ове сврхе [19, 20, 21]. Међутим, називи елемената у шемама обично не обезбеђују довољне информације о контексту како би се разрешиле потенцијалне недоумице. На пример, речи дрво и стабло могу бити синоними, а и не морају, уколико се посматра да стабло у рачунарској техници може означавати структуру података. Дакле, употреба општих речника може резултовати у многим погрешним упаривањима с обзиром на полисемију (вишезначност) речи. Управо

се из тог разлога, многи системи за проверу сличности, попут [22, 23, 24, 25], ослањају на специфичне, домен оријентисане, речнике и таксономије које садрже јединствене називе, синониме, скраћенице итд. Иако креирање таквих речника може захтевати поприличан напори и доста времена, показало се да је то вредно улагања уколико се жели да обезбеди подршка за многе послове приликом провере сличности у неком домену, поготово у случајевима када шеме имају равну структуру где речници могу дати јако добре резултате приликом упаривања.

Веома често, шеме поседују описе дате на обичном језику (природном, говорном језику) како би се документовала намена и жељена семантика одређених елемената шеме. Такви детаљни коментари могу бити лингвистички обрађени са циљем одређивања сличности елемената. На пример, описи могу бити обрађени користећи технике обраде природног језика (NLP – Natural Language Processing) како би се добио скуп карактеристичних речи за потребе поређења. Дуги описи појединих елемента такође могу бити посматрани као врста документације, као што је урађено у прототипу Delta [26]. Разматрање описа као контексних информација за разумевање значења појединих елемената шеме, технике разјашњавања значења речи [27] могу бити употребљене приликом идентификације који елементи имају исто или слично значење.

У шемама које се срећу у пракси, може се уочити неколико ортогоналних проблема, који утичу на квалитет упаривања назива и значења елемената у шеми:

- Називи састављени од више речи: Називи елемената шеме могу бити сачињени од више речи, попут АдресаПошиљаоца, АдресаОд или АдресаСаКојеЈеПослато. Директно поређење ових имена, веома ретко

доводи до коректних резултата одређивања сличности. Из тог разлога, користе се различите технике препроцесирања, као што је растављање на саставне делове тзв. токенизација (нпр. АдресаПошиљаоца даје {Адреса, Пошиљаоца}), уклањање везника и прилога (као што је Од, Са итд.), извођење речи (Пошиљаоца у Пошиљалац), све са циљем да се дође до карактеристичних речи које би биле употребљене за упоређивање [22, 24].

- Скраћене речи: Речи се веома ретко у називима појављују у свом пуном облику, најчешће због ограничења у простору за чување тих речи. Из тог разлога се веома често користе скраћенице, попут Адр уместо Адреса, које се углавном могу детектовати употребом поступка за одређивање приближне сличности стрингова. Међутим, уколико су називи сувише скраћени, као у случајевима када се користе само почетна слова речи као код акронима, на пример АП уместо АдресаПошиљаоца, онда одређивање сличности не даје добре резултате. У тим случајевима, неопходна је употреба речника акронима.

Хомоними: Одређивање сличности имена елемента шеме може дати погрешне резултате због постојања хомонима. Конкретно, називи елемента могу бити исти, али могу представљати потпуно различите концепте. На пример, стабло може представљати део биљке, а уједно представља у структуру података у рачунарским наукама. Да би се решио проблем хомонима, неопходна је употреба додатних информација у шемама, како би се они разликовали.

3.1.1.2. Приступи засновани на ограничењима

Шеме често садрже ограничења којима се за елементе тих шеме декларишу типови података, скупови дозвољених вредности, ограничења по питању

јединствености, опционалности, кардиналности итд. Уколико су такве информације доступне у шемама које се пореде, оне могу бити употребљене за одређивање сличности појединих елемената [28]. На пример, сличност може бити заснована на еквиваленцији типа податка, или битним карактеристикама (јединственост, могућност идентификације итд.).

Имајући у виду јединствену семантику појединих ограничења, уобичајен приступ је да се најпре обезбеди табела компатибилности ради аутоматске претраге за елементима који могу бити међусобно слични. На пример, табела компатибилности за типове података може покрити случајеве када се у шемама користе реални бројеви са различитом прецизношћу (једнострука и двострука прецизност) или различитим број дозвољених карактера у стринговима итд. Са друге стране, табела компатибилности у неким случајевима треба да буде толерантна према конфликтима типова, на пример између бројева и стрингова, како би се избегло потенцијално прерано елиминисање кандидата упаривања. Ово се може обезбедити додељивањем различитих вредности сличности различитим нивоима компатибилности.

Приступ провере сличности заснован на ограничењима, уколико се самостално користи, веома често доводи до несавршеног упаривања код ког сваком елементу из прве шеме може одговарати више елемента из друге шеме (тј. коресподенција n:m), с обзиром да је чест случај да више елемената у шеми има међусобно компатибилна ограничења. На пример, уколико се разматра само тип податка, онда сви елементи тог типа податка јесу међусобно слични. Међутим, ограничења ипак представљају помоћно средство које може ограничити простор претраге

приликом одређивање кандидата за упаривање, поготово уколико се овај приступ комбинује са другим ради повећања тачности.

3.1.1.3. Приступи засновани на структури

Приступи засновани на структури користе односе између елемената и упарују комбинације елемената који се појављују заједно у некој врсти структуре. У зависности од експресивности језика који се користи за опис шеме, различити односи између елемента могу постојати, на пример однос садржања, однос целине и дела, или референцијални интегритет. Обично су елементи шеме и њихови односи представљени у виду усмереног, или неусмереног, графа тако да различите врсте структурно повезаних елемента могу бити идентификовани као слични. Структура шеме може бити посматрана на два начина. Први начин дефинише околину елемента коју треба разматрати приликом израчунавања сличности, а други начин дефинише досег упаривања приликом итеративног поступка одређивања сличности.

Најчешћи разлог за посматрање структуре шеме јесте разматрање околине неких елемената приликом процене њихове сличности. Многе шеме имају структура која је хијерархијског типа, заснована на некој врсти односа садржања. Да би се проценила сличност између два елемента, могу се упоредити различите врсте елемента у њиховој околини, као што су родитељи, деца, обухваћени листови, као што је [22, 24, 29, 30]. У случају репрезентације структуре шеме у виду неусмереног графа, сви суседни елементи могу бити посматрани на равноправан начин [25, 31]. На пример, може се користити метрика заснована на раздаљини како би се идентификовали сви елементи који спадају у околину неког посматраног елемента. Тако би раздаљина 1 обухватала све елементе непосредно

повезане са посматраним елементом, док би раздаљина n > 1 обухватила све елементе до којих се може доћи обиласком графа у n -1 корака полазећи од посматраног елемента. Сличност између елемента који представљају околину посматраног елемента би могла бити одређена пре обиласка графа, користећи неки од приступа заснованим на елементима, а не на структури, на пример сличност на основу назива [22]. Софистицираније приступи, као код прототипа Dike [25] и SimilarityFlooding [31], за одређивање сличности између околних елемената користе рекурзивну пропагацију израчунатих сличности засновану на аритметици фиксне тачке. Када се фиксна тачка достигне, вредности добијене за сличност се сматрају стабилизованим и узимају се као структурна сличност елемената шеме.

Други разлог за посматрање структуре шеме је одређивање досега упаривања приликом итеративног поступка одређивања сличности. Ово се може постићи обиласком шеме у одозго-наниже или одоздо-навише маниру. Конкретно, слични елементи идентификовани на једном нивоу одређују досег посматрања за наредни ниво. Тај наредни ниво разматра или њихове наследника (код приступа одозго-наниже), или њихове претходнике (код приступа одоздо-навише). Алгоритам за приступ одозго-наниже је обично мање захтеван зато што сличности одређене на вишем нивоу апстракције ограничавају простор претраге на нижим нивоима апстракције на оне комбинације које су условљене сличношћу надређених елемената. Међутим, у случајевима приступ одозго-наниже може дати лоше резултате када се структуре на вишем нивоу доста разликују чак иако постоји добро поклапање елемената на нижим нивоима. Приступ одозго-наниже се може користити и само на појединим фрагментима шеме, као што је случај код система

COMA++[32] и код [33], како би се омогућила обрада веома великих шема и онтологија. Насупрот томе, приступ одоздо-навише упоређује све могуће комбинације елемената на нижим нивоима апстракције, и тиме налази све могуће парова елемената чак иако се структуре шема на вишим нивоима могу битно разликовати.

Уопштено посматрано, провера сличности на структурном нивоу може дати јако лоше резултате уколико постоји структурни конфликти, који настају због разлике у нивоу детаља представљених елементима шеме. На пример, посматрање елемента адреса који је представљен у виду једног стринга неће дати сличност приликом упоређивања са елементом који има структуру и при том се састоји од броја, улице и града. Овакве ситуације управо представљају мотив зашто треба комбиновати више приступа како би се обезбедила робусност приликом провере сличности. На пример, треба разматрати неколико врсте околних елемената, па уколико постоји сличност на вишем или на нижем нивоу, онда треба оставити могућност да то можда и јесу слични елементи.

3.1.2. Аспект заснован на инстанци

Провера сличности заснована на инстанци користи конкретне податке из инстанци креираних над шемама како би се утврдили који елементи тих шема међусобно одговарају једни другима. Оваква провера сличности представља користан избор у случајевима када су информације о шемама ограничене, као што је то обично случај са полу структурираним подацима. У екстремним ситуацијама када шема није доступна, она може бити реконструисана (до одређеног нивоа) на основу података доступних у инстанци, и то користећи неку од техника као што су [11, 34]. Чак и када је поприлично информација доступно за неку шему,

разматрање инстанци може бити комплементаран поступак који би поступцима заснованим на шемама обезбедити додатан увид у семантику и садржају појединих елемената шеме. Са друге стране, применљивост поступака заснованих на инстанцама зависи од доступности репрезентативних података у самој инстанци. Такође, тачност провере сличности у великој мери зависи од квалитета података у инстанци, што доводи до потребе за додатним напорима да би се обавило чишћење података. Количина података коју треба анализирати код поступака заснованих на инстанцама је обично доста већа него што је то случај код поступака заснованих на шемама. Управо зато, време извршавања постаје критично питање, поготово код интерактивне употребе.

3.2. Предлог поступка провера сличности концептуалних модела

Предложени поступак провере сличности концептуалних модела се састоји у обављању два корака. Најпре да се концептуални модели сведу на исту репрезентацију, а потом да се обави одређивање разлика. Први корак има за циљ да посматране концептуалне моделе преведе у облик погодан за даљу анализу. За ту сврху је у овом раду предложена употреба резултата из области формалне анализе концепата. инспирисано употребом у области анализе софтверских решења и реверзног инжењерства [35]. На тај начин се концептуални модели преводе у одговарајуће усмерене ацикличне графове, док се при томе чува информација о нивоу апстракције појединих елемената, што представља основу за интерактивну проверу сличности. Циљ другог корака јесте да одреди разлику између концептуалних модела чија се сличност проверава, и то кроз одређивање мере упарености чворова концептуалних решетки и детекцију разлике међу њима.

3.2.1. Репрезентација концептуалних модела

Формална анализа концепата (Formal Concept Analysis – FCA) је метода за представљање знања, управљање информацијама, и анализу података. FCA методом се проналазе и визуализују сви концепти и њихове међузависности из улазних података. Формална анализа концепата се примењује на различитим подручјима попут математике, биологије, социологије, психологије. Најзанимљивија је примена FCA на подручју рачунарства – користи се у претраживању података, анализи и одржавању програмског кода као и стварању таксономија и онтологија [36]. Исто тако примењује се као метода машинског учења и у подручју истраживања вештачке интелигенције.

Методу FCA је осмислио почетком осамдесетих година двадесетог века R.Wille, немачки математичар и професор емеритус на техничком универзитету у Дармштаду. Искористио је филозофско тумачење концепта (појма) као јединице мисли обликоване генерализацијом којом се обухвата неки скуп објеката и скуп њихових атрибута. Улазни подаци за методу FCA се приказују у облику матрице код које свака врста представља један објекат из домене од интереса, а свака колона један од дефинисаних атрибута. Ако неки објект има одређени атрибут то се евидентира у матрици стављањем ознаке "X" у пољу где врста тог објекта сече колону тог атрибута. Иначе ако неки објекат нема одређени атрибут онда поље у пресеку одговарајуће врсте и одговарајуће колоне остаје празно. Ова матрица се дефинише као формални контекст над којим се спроводи даља анализа. Дефиниције употребљене у овом раду дате су у [37, 38].

Формална анализа концепата резултује у два скупа излазних података. Први скуп даје хијерархијски однос свих идентификованих концепата у облику усмереног

ацикличног графа (диграфа) односно такозване концептуалне решетке (concept lattice). Други скуп представља везе свих установљених међузависности атрибута из формалног контекста.

Дефиниција 1: Формални контекст методе FCA је тројка (O ,A , I) где су O и A непразни скупови, а I бинарна релација између O и A.

За улазну матрицу са n врста и m колона формални контекст (O ,A , I) се састоји од скупова O ={o_1 , ... , o_n} и A={ a_1 , ... , a_m} као и релације I дефинисане као (o_i , a_j)∈I , ако и само ако поље улазне матрице које се налази у пресеку i-те врсте и j-те колоне није празно.

Дефиниција 2: За формални контекст (O ,A , I) дефинисани су оператори формирања концепата ↑:$2^O \rightarrow 2^A$ и ↓:$2^A \rightarrow 2^O$ за свако X⊆O и свако Y⊆A као:

X↑={a∈A |∀ o∈X: (o , a)∈I} и Y↓={o∈O |∀ a∈Y: (o ,a)∈I}.

Као што се може видети оператори ↑ и ↓ имају дуалне дефиниције. При томе, X↑ представља скуп свих атрибута које имају сви објекти из X, а Y↓представља скуп свих објеката који деле све атрибуте из Y. Формални концепт је сегмент формалног контекста у којем различити објекти деле исте атрибуте.

Дефиниција 3: Формални концепт у формалном контексту (O ,A , I) је пар (X, Y) такав да (X⊆O и Y⊆A) и за који важи X↑=Y и Y↓=X.

Дакле, (X, Y) је формални концепт ако и само ако се X састоји само од објеката који имају све атрибуте из Y, такозвана екстензија концепта (extent), а Y се састоји само од атрибута које деле сви објекти из A, такозвана интенција концепта (intent). Екстензија и интенција концепта може се и формално дефинисати и то на

следећи начин:

Ext (O ,A , I) ={$X \in 2^O$ |(X, Y)$\in$K(O, A , I) за неко Y},

Int (O ,A , I)={$Y \in 2^A$ |(X, Y)$\in$K(O, A, I) за неко X}.

Где K(O ,A , I) представља концептуалну решетку. Исто тако могу се исказати и помоћу оператора формирања концепата, екстензија као Ext (O ,A , I)={Y↓| Y$\subseteq$A}, а интенција концепта као Int(O ,A , I)={X↑| X$\subseteq$O}.

На други начин се формални концепти могу визуално дефинисати као максимални правоугаоници (над ознакама "X") које је могуће уписати у таблицу формалног контекста. Важно је нагласити да се формални концепти међусобно налазе у хијерархијском односу (подконцепт и надконцепт).

Дефиниција 4: За формалне концепте (X1 , Y1) и (X2 , Y2) из формалног контекста (O ,A , I) важи (X1 , Y1)$\leq$(X2 ,Y2) ако и само ако важи X1$\subseteq$X2 и Y2$\subseteq$Y1.

Оператор $\leq$ исказује однос подконцепт и надконцепт, аналогно принципу подкласе и надкласе. Сви формални концепти неког формалног контекста и њихови међусобни односи могу се трансформисати у структуру концептуалне решетке.

Дефиниција 5: Структура K=(O ,A , I) се дефинише као скуп свих формалних концепата, K=(O ,A , I)={(X, Y) $\in 2^O \times 2^A$ | X↑=Y, Y↓=X} . Ова структура с укљученим хијерархијским односом (K=(O ,A , I) , $\leq$) чини концептуалну решетку формалног контекста (O ,A , I).

Структура концептуалне решетке задовољава математичку дефиницију решетке јер представља парцијално уређени скуп у којем било која два елемента

(формална концепта) имају заједнички супремум тј. најмању горњу границу (join), у FCA контексту такозвани надконцепт, и заједнички инфимум тј. највећу доњу границу (meet), у FCA контексту такозвани подконцепт.

Метода FCA је математички строго дефинисана, а заснива се на Галоисовим везама (Galois connections) и операторима затварања (closure operators). Описани оператори формирања концепата $\uparrow:2^O \rightarrow 2^A$ и $\downarrow:2^A \rightarrow 2^O$ као и хијерархијски оператор $\leq$ чине Галоисову везу (веза између скупа атрибута и скупа објеката у формалном контексту). Оператори затварања се добијају композицијом оператора за формирање концепата: $\uparrow\downarrow:2^O \rightarrow 2^O$ и $\uparrow\downarrow:2^A \rightarrow 2^A$. Најважнија теорема у FCA, тзв. централна теорема концептуалних решетки, којом се исказује да је концептуална решетка не само решетка већ и потпуна решетка тј. парцијално уређени скуп чији сви подскупови имају и супремум (надконцепт) и инфимум (подконцепт).

Може се догодити да се у формалном контексту иста врста или колона појави више пута. Тада се формални контекст може очистити (кларифицирати) уклањањем редундантних врста и колона. Концептуалне решетке које се добију из почетног формалног контекста и очишћеног формалног контекста су изоморфне. Исто тако могуће је редуковати формални контекст избацивањем врста и редова према задатим правилима, а на крају добити концептуалну решетку изоморфну с оном која се добије из нередукованог формалног контекста.

Описано је више алгоритама за израчунавање концептуалне решетке. Такви алгоритми као улаз захтевају формални контекст (O, A, I) . Један од начина генерисања концептуалне решетке је се најпре израчуна екстензију $Ext(O, A, I)$, а потом за свако o , такво да је $o \in Ext(O, A, I)$ одреди $(o, o\uparrow)$.

Једноставнији алгоритми дају као излаз структуру K=(O ,A , I) из које се накнадно могу израчунати сви хијерархијски односи и добити концептуална решетка. Сложенији алгоритми су ефикаснији и директно као излаз дају концептуалну решетку (K=(O ,A , I) , ≤) . Један од алгоритама за одређивање структуре K=(O ,A , I), јесте алгоритам "Следећег затварача" (next closure), који као улазни податак користи формални контекст (O, A, I), а као излаз даје Int(O, A, I) – лексикографски сортирану листу свих интенција улазног формалног контекста. Из те листе може се реконструисати структура K=(O ,A , I) јер важи да је K=(O ,A , I)={(X↓, X)|X∈Int(O, A, I)}. На крају се, употребом дефиниције 4, могу одредити хијерархијски односи у структури K=(O ,A , I) како би се добила концептуална решетка.

Дефиниција 6: Атрибути X, Y ⊆A , i∈{1,... ,n} су лексички сортирани ($X <_i Y$) ако и само ако: i∈Y−X и X∩{1,... ,i−1}=Y∩{1,... , i−1}.

Дефиниција 7: За X⊆A ,i∈{1,... , n} дефинише се операција X⊕i :=((X∩{1,... , i−1}) ∪ {i})↑↓.

Теорема о лексичком следбенику каже да је најмања интенција Y+ већа од Y⊆A задата са Y+=Y⊕i гдје је i највећи елемент од A за који важи $Y <_i Y⊕i$.

Псеудокод алгоритма „Следећег затварача" може бити наведене у шест корака као на слици 3.1. Временска сложеност алгоритма "Следећег затварача" је дата са $O(|O|\cdot|A|^2\cdot|K=(O, A, I)|)$.

Величина концептуалне решетке може расти експоненцијално с повећањем формалног контекста и за велики улазни скуп података може постати нечитљива.

```
Корак 1.      Y:=⍉↑↓; (најмања интенција)
Корак 2.      store(Y);
Корак 3.      while not (Y=A) do
Корак 4.         Y:= Y+; (лексички следбеник)
Корак 5.         store(Y);
Корак 6.      endwhile.
```

Слика 3.1 Псеудокод алгоритма „Следећег затварача“

Зато се формални контекст може поделити груписањем атрибута у мање скупове. Након тога се могу израчунати концептуалне решетке за сваки део формалног контекста посебно, а такође је затим могуће добити и прегледнији запис читаве концептуалне решетке рачунањем продукта мањих концептуалних решетки. Велике концептуалне решетке могу се поједноставити и коришћењем технике груписања (clustering) концепата. У концептуалној решетки се приказују само они формални концепти чији атрибути имају подршку (support) већу од задате минималне вредности из интервала [0,1]. Подршка се рачуна као:

$$\mathrm{supp}(X \subseteq A) = \frac{|X|}{|O|}$$

Подршка неког скупа атрибута је број свих објеката који имају те атрибуте подељен с бројем свих објеката у формалном контексту. Тако се повећањем потребне минималне подршке може смањити концептуална решетка.

Из концептуалне решетке или формалног контекста могу се пронаћи и међузависности односно импликације атрибута. Импликација атрибута се може директно ишчитати из концептуалне решетке (сваки формални концепт обавезно садржи све атрибуте из свих својих надконцепата), али и формално дефинисати као веза између методе FCA и формалне логике. Утврђивање импликација атрибута је посебно корисно у случају великог формалног контекста и његове

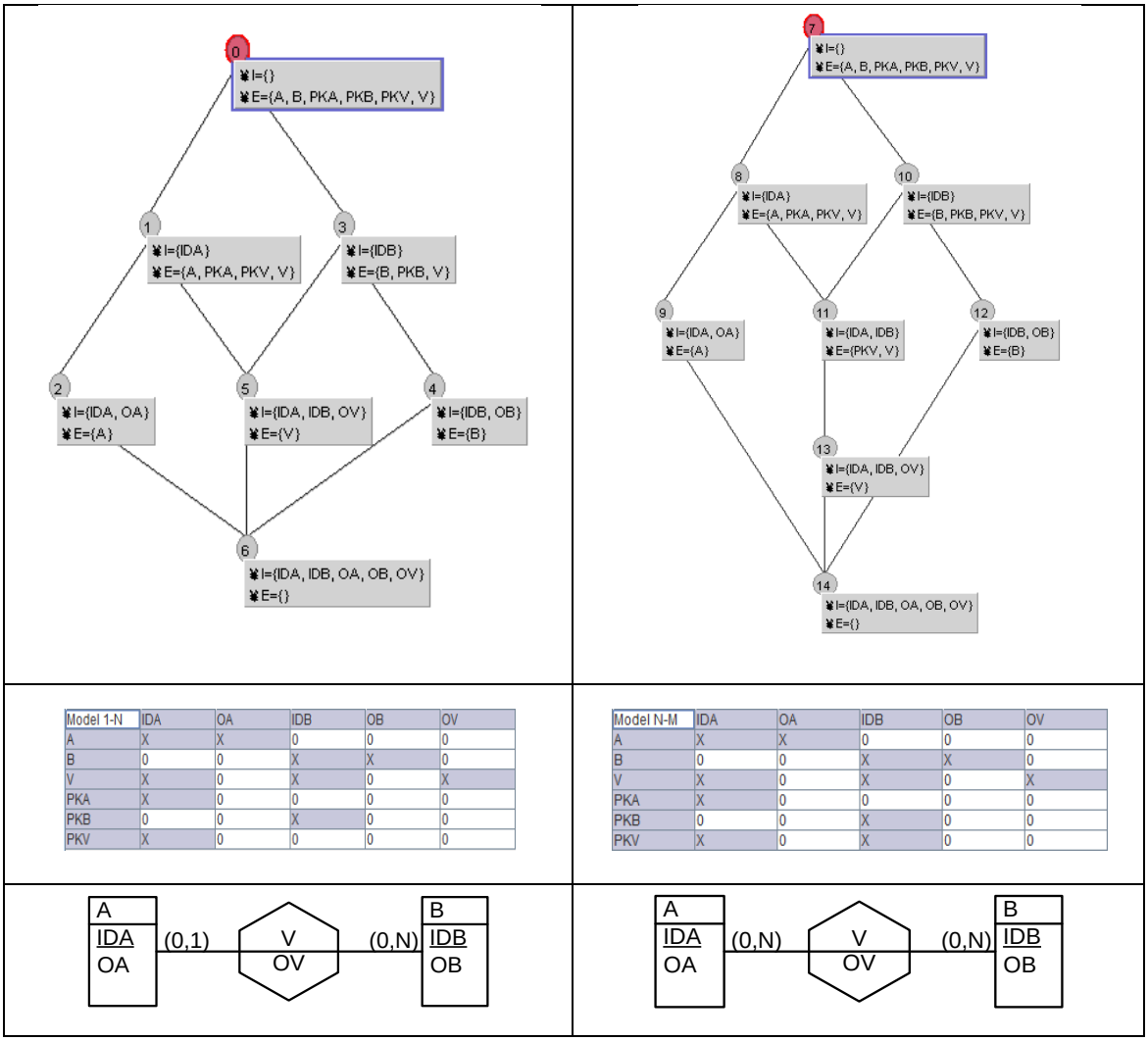
густе и непрегледне концептуалне решетке. Тада се може пронаћи минимални скуп свих импликација између атрибута које су истините у том формалном контексту.

Различите нотације које се користе приликом дефинисања концептуалних модела база података (Chen, IDEF1X, Crow's Feet) имају велику експресивност и потребан ниво апстракције за свакодневну употребу. Међутим, уколико се жели међусобно упоређивање концептуалних модела, ове нотације не обезбеђују довољан ниво апстракције. Управо је то један од разлога због чега је у овом раду предложена употреба формалне концептуалне анализе као вида репрезентације концептуалних модела база података. Овиме се такође омогућава и поређење концептуалних модела представљених различитим нотацијама. Други разлог за употребу формалне концептуалне анализе је строго дефинисан и добро развијен математички апарат који се може употребити приликом анализе концептуалних модела који се посматрају. Трећи разлог за употребу формалне концептуалне анализе се управо налази у чињеници да формални контекст и одговарајућа концептуална решетка обезбеђују информацију о парцијалној уређености скупа формалних концепата, што представља основу за правилно одлучивање приликом интерактивног поступка провере сличности концептуалних модела.

Генерално, алгоритам превођења концептуалног модела базе података у одговарајући формални контекст није јединствен. То значи да могу постојати различити алгоритми, који пре свега зависе од намене, количина и типа информација које се желе очувати из оригиналног концептуалног модела. Алгоритам предложен у овом раду се састоји од три корака, и има за циљ да отклони недостатке неких претходних решења [34].

У првом кораку се формира скуп свих објеката формалног контекста, односно скуп свих екстензија свих формалних концепата тог формалног контекста. Најпре се за сваки јаки ентитет из концептуалног модела формирају два нова објекта у формалном контексту, један који одговара самом ентитету и други који одговара идентификационом ограничењу (примарном кључу). Потом се, такође, и за сваки слаби ентитет из концептуалног модела формирају по два нова објекта, као и код јаких ентитета, али се и за сваки однос зависности (директни однос између ентитета) формира по један нови објекат који одговара референцијалном ограничењу (страном кључу) који тај слаби ентитет садржи. На крају се и за сваку везу (индиректни однос између ентитета) формирају по два нова објекта (један за саму везу и други за идентификационо ограничење), док се и за сваког од учесника у вези формира по један нови објекат који одговара референцијалном ограничењу.

У другом кораку се формира скуп посматраних атрибута формалног контекста, односно скуп свих интенција свих формалних концепата тог формалног контекста. Најпре се за сваки атрибут сваког од јаких ентитета из концептуалног модела формира по један нови атрибут. Потом се за сваки атрибут сваког од слабих ентитета из концептуалног модела формира по један нови атрибут, такође се и за сваки однос зависности, који ти слаби ентитети имају, формира по један нови атрибут који одговара референцијалном ограничењу. На крају се и за сваки атрибут сваке од веза формира по један нови атрибут, али се и за сваког од учесника у свакој од тих веза формира по један нови атрибут који одговара референцијалном ограничењу.



Слика 3.2 Пример употребе поступка репрезентације на концептуалним моделима са по два јака ентитета и са по једном везом: а)1-N б) N-M

У трећем кораку се формира бинарна релација између скупа свих објеката формираних у првом кораку алгоритма, и скупа свих атрибута формираних у другом кораку алгоритма. Сви атрибути који су настали на основу неког ентитета означавају се да су у релацији са објектом формираним на основу тог ентитета. Такође, атрибути који у неком ентитету учествују у неком идентификационом ограничењу, означавају се као да су у релацију са објектом насталим на основу тог идентификационог ограничења. Код атрибута и објеката насталим на основу слабих ентитета, поступак се понавља као код јаких ентитета, уз додатак да се сви

атрибути који одговарају неком од референцијалних ограничења означавају да су у релацији са одговарајућим ограничењем. Слично, у случају атрибута и објеката насталим на основу веза, релација се успоставља између свих атрибута насталих на основу неке везе и објекта насталог на основу те везе, али се такође као у релацији означавају и сви атрибути који одговарају идентификационом ограничењу односно референцијалним ограничењима учесника у вези. Специјално, код свих објеката који одговарају неком референцијалном ограничењу, да се у релацији означавају се оба атрибута, односно и реферишући и референсани атрибут.

Предложени алгоритам омогућава адекватно представљање не само јаких и слабих ентитета и индиректних релација међу њима, већ и директних релација , при том подржавајући кардиналности условљавања већих од један. Такође, алгоритам омогућава додавање нових типова ограничења која могу важити над једним или више атрибута. Типови подржаних ограничења зависе како од нивоа доступних информација у оригиналном концептуалном моделу тако и од нивоа апстракције који се жели представити формалним контекстом. Поступак проширивања посматраног скупа ограничења састојао би се у томе да се за свако ново ограничење које се жели посматрати, додаје одговарајући објекат у првом кораку алгоритма, потом додавање одговарајућег атрибута у другом кораку алгоритма, али само уколико то ново ограничење повлачи са собом креирање новог атрибута у оригиналном концептуалном моделу, у супротном није потребно додавање нових атрибута. У трећем кораку алгоритма, би се као у релацији означили сви атрибути који учествују у том новом ограничењу. Пример примене описаног алгоритам репрезентације концептуалних модела дат је на слици 3.2.

3.2.2. Одређивање разлика концептуалних модела

Одређивање разлика концептуалних модела почиње након што је сваки од модела чија се сличност проверава најпре репрезентован у виду концептуалне решетке. Поступак одређивања разлика се спроводи у два корака. Први корак има за циљ да успостави коресподенцију, односно да одреди меру упарености између појединих чворова концептуалних решетки. Циљ другог корака јесте да обави анализу упарених чворова и дефинише разлике између концептуалних решетки, односно одговарајућих елемената посматраних концептуалних модела.

3.2.2.1. Одређивање упарености концептуалних решетки

Одређивање коресподенције између појединих елемената модела, чија се сличност проверава, је активност која се често назива упаривање. У овом раду су за потребе проблема упаривања искоришћени резултати алгоритам за упаривање графова, под називом Поплава сличности (Similarity flooding) , предложеног за широк спектар различитих сценарија употребе [39].

Алгоритам се заснива на употреби усмерених означених графова, а то је управо репрезентација која је у овом раду предложена, с обзиром да се посматра концептуална решетка са хијерархијским односом. Свака грана у графу је представљена као уређена тројка (s, p, o), где су s и o чворови графа и представљају извор, односно одредиште, повезаних граном p. Упаривање се одређује итеративним поступком рачунања фиксне тачке. При овом рачунању основна идеја је да нека два чвора, која потичу из два графа који се упарују, јесу слична уколико се налазе у сличном окружењу, тј. уколико су слични њима инцидентни чворови. Другим речима, сличност два чвора се пропагира на друге чворове који су њихови суседи.

Граф пропагације је помоћна структура која се формира на основу полазних графова чије се упаривање обавља. Пре формирања графа пропагације, потребан је и граф повезаности парова. Граф повезаности парова (PCG – pairwise connectivity graph), за графове A и B се дефинише:

$((x, y), p, (x', y')) \in PCG(A, B) \Leftrightarrow (x, p, x') \in A \wedge (y, p, y') \in B$

Односно, сви чворови у графу повезаности парова, такозвани парови мапирања, представљају елементе из A×B. Другим речима, пар мапирања, говори да су чворови x и x' графа A, слични чворовима y и y' графа B, уколико постоји исти тип гране p, који повезује x и x', односно y и y', у њиховим графовима.

За сваку грану у графу повезаности парова, граф пропагације садржи и додатне гране које су усмерене у супротном смеру. Свака грана у графу пропагације говори колико добро сличност одређеног пара мапирања, пропагира на њихове суседе. За ту сврху се користе, такозвани, коефицијенти пропагације који су у опсегу од 0 до 1, инклузивно, и могу се израчунати на више начина. Уколико из једног чвора има већи број излазних грана истог типа, онда се коефицијент пропагације равномерно расподељује међу њима.

Нека је $\sigma(x, y) \geq 0$ мера сличности чвора $x \in A$ и чвора $y \in B$, која је дефинисана као тотална функција над A×B. Ова мера се скраћено назива упареност. Алгоритам је управо заснован на итеративном рачунању вредности упарености σ, па у складу са тим вредност σi представља упареност између чворова у итерацији i. Слично, упареност $\sigma 0$ представља почетну сличност која се може рачунати на различите начине, као на пример користећи упоређивање назива чворова (упоређивањем

стрингова). У најопштијем случају може се претпоставити да почетна упареност није доступна, односно да је σ0(x, y)=1 за све парове (x, y)∈A×B.

У свакој итерацији, упареност σ за пар мапирања (x, y) се увећава за вредност упарености σ сваког њима суседног пара мапирања, у графу пропагације, помножен одговарајућим пропагационим коефицијентом, тј. пропагационим коефицијентом на грани усмереној од суседних парова мапирања ка пару мапирања (x, y). Након тога се вредности свих упарености нормализују, тј. деле се са вредношћу максималне упарености (максималне међу тренутним вредностима упарености).

Генерално, упареност σi+1 се рачуна на основу вредности упарености σi на основу следећих израза:

$$\varphi_{\sigma_i}(x,y) = \sum_{(a_u,p,x)\in A,(b_u,p,y)\in B} \sigma_i(a_u,b_u)\cdot w((a_u,b_u),(x,y)) + \sum_{(a_v,p,x)\in A,(b_v,p,y)\in B} \sigma_i(a_v,b_v)\cdot w((a_v,b_v),(x,y))$$

$$\sigma_{i+1}(x,y) = \text{normalize}\left(\sigma_i(x,y) + \varphi_{\sigma_i}(x,y)\right) = \text{normalize}\left(t_{i+1}(x,y)\right) = \frac{t_{i+1}(x,y)}{\max\{t_{i+1}\}}$$

Описано рачунање се спроводи итеративно све док Еуклидско растојање вектора остатака Δ(σn, σn-1) не постане мање од ε за неко n>0. Уколико рачунање не конвергира, оно се обуставља након неког унапред дефинисаног броја итерација. Оно што је управо описано јесте функција за итеративно рачунање инкремената упарености за сваки пар мапирања на основу упарености његових суседа у графу

пропагације. Емпиријски је утврђено да се са аспекта квалитета упарења и брзине конвергенције најбоље показала функција:

$$\sigma_{i+1}(x,y) = \text{normalize}\left(\sigma_0(x,y) + \sigma_i(x,y) \ + \ \varphi_{\sigma_0+\sigma_i}(x,y)\right)$$

Применом описаног алгоритма на усмерене ацикличне графове, тј. концептуалне решетке које представљају репрезентацију концептуалних модела чија се сличност проверава, добија се матрица упарености. У матрици упарености се за сваки чвор из једне концептуалне решетке налази информација о упарености са чворовима из друге концептуалне решетке. На основу тога је могуће за сваки чвор из једне концептуалне решетке одредити да ли постоји кореспондентан чвор у другој концептуалној решетки.

3.2.2.2. Детекција разлика упарености концептуалних модела

Након што је обављено одређивање упарености између чворова концептуалних решетки које одговарају концептуалним моделима који се пореде, потребно је и одредити разлике међу тим моделима. За ту сврху се може искористити чињеница да концептуална решетка са хијерархијским односом (K=(O ,A , I) , ≤) представља усмерени ациклични граф (DAG - directed acyclic graph) у којем су формални концепти (чворови) повезани усмереним гранама (смер увек од надконцепта према подконцепту). Дакле, концептуална решетка се може анализирати коришћењем различитих алгоритама и техника из подручја теорије графова.

У овом раду је предложено да се при одређивању разлика два концептуална модела, најпре одреди линеарно сортирану листу формалних концепата који одговарају моделима који се пореде. Линеарно сортирана листа формалних концепата се формира узимајући у обзир све међусобне односе надконцепт-

подконцепт из концептуалне решетке. То се може решити применом постојећих алгоритама за тополошко сортирање (topological sort). Они дају линеарно сортирану листу чворова на начин да за сваку грану од чвора U ка чвору V из усмереног ацикличног графа G чвор U долази у сортирани поредак пре чвора V. Дакле, сваки чвор улази у линеарно сортирану листу тек ако су у листи већ сви његови чворови родитељи или ако уопште нема чвор родитељ. Исказано формалније, алгоритми за тополошко сортирање дају линеарно уређен скуп који одговара неком парцијалном уређеном скупу (нпр. концептуална решетка).

Постоји више алгоритама за тополошко сортирање, али се могу поделити у два скупа - први скуп су алгоритми засновани на уклањању извора (чвора без улазних грана), а други скуп чине алгоритми који су засновани на претраживању графа по дубину (DFS - depth first search). Основни кораци оба типа алгоритама: Алгоритам за уклањање извора на почетку тражи чвор(ове) без улазних грана, брише их (заједно с њиховим излазним гранама) из графа и убацује у листу. Ова два корака понавља све док граф не буде потпуно празан, а тада је попуњена линеарно сортирана листа чворова. Алгоритам заснован на претраживању по дубину врши DFS пролазак кроз граф и бележи поредак по којем су чворови потпуно истражени. На крају се обртањем забележеног поретка добије тражена линеарно сортирана листа чворова. Имплементације ових алгоритама су сложеније јер проверавају да ли у графу постоји циклус, и ако постоји, сортирање се прекида јер се такав граф (DCG - directed cyclic graph) не може тополошки сортирати. Временска сложеност ових алгоритам је O(|V|+|E|), односно линеарно је зависна од броја чворова и грана. Ови алгоритми могу да дају више различитих исправних коначних поредака чворова, другим речима решење није јединствено. Ипак, ако у

графу постоји усмерени Хамилтонов пут (пут који пролази кроз сваки чвор графа тачно једанпут) сваки алгоритам за тополошко сортирање ће дати исто решење.

Након што се одреде линеарно сортиране листу формалних концепата који одговарају концептуалним моделима који се пореде, потребно је одредити разлике међу њима. Разлике се одређују на тај начин што се једна од листа узима као основ поређења (тзв. основна листа), што значи да у том случају друга листа представља предмет поређења (тзв. предметна листа). Најпре се обавља пролазак кроз основну листу. Елементи се из основне листе узимају редом којим су сортирани. Уколико се најпре дохватају елементи са почетка листе, то значи да се поређење обавља по принципу одозго наниже, односно, полазећи од генералнијих концепата. Могуће је и прво дохватати елементе са краја листе, што онда значи да се поређење обавља по принципу одоздо навише, односно полазећи од конкретнијих концепата па даље ка општијим. Без обзира на одабрани редослед након дохватања елемента из основне листе, потребно је одредити да ли постоји њему одговарајући елемент у предметној листи, тј. да ли постоји њему кореспондентан елемент са којим је упарен. Уколико постоји, потребно га је уклонити из предметне листе и прећи на дохватање следећег елемента из основне листе. Поступак се понавља све док се не обраде сви елементи из основне листе. Уколико се приликом провере дође у ситуацију да за посматрани елемент, из основне листе, не постоји одговарајући елемент у предметној листи, онда је то прва детектована разлика између концептуалних модела који се пореде. Процес провере и одлучивања који елемент из предметне листе, одговара одређеном елементу из основне листе, састоји се у консултовању матрице пресликавања

формиране у поступку упоређивања репрезентација концептуалних модела, односно њима припадајућих формалних контекста.

Све детектоване разлике, између концептуалних модела који се пореде, је могуће обрађивати на два начина, пакетни и интерактивни. Пакетни начин се састоји у томе да се све детектоване разлике наведу истовремено, док се интерактивни начин састоји у томе да се поређење зауставља у тренутку детектовања прве разлике међу моделима који се пореде. Оба начина имају својих предности, пакетни омогућава да се све разлике целовитије сагледају и тако одабере најбоље решење, међутим у случају великих модела и великог броја разлика то може бити не тривијалан задатак. Са друге стране, интерактивни начин, омогућава постепено разрешавање разлика, чиме се избегава проблем приликом поређења великих модела, међутим, овај начин не обезбеђује увид и све разлике. Треба имати у виду да се и код пакетног начина, уколико га примењује крајњи корисник, обрада разлика обавља секвенцијално, тј. једна по једна. Другим речима, из угла корисника, идеално би било да му се омогући увид у све разлике, али да се обезбеди подршка тако да се те разлике презентују у интерактивном маниру, и да се након сваке обрађене разлике, тј. разрешеног конфликта, сагледа какав је то ефекат оставило на целину. Без обзира на то што се ради о комбинацији два начина, овај комбиновани приступ, је по својој природи, интерактиван. Уколико се у овом интерактивном поступку наиђе на ситуацију да и након корисникове интервенције и покушаја да разреши детектовану разлику, нема одговарајућих резултата, може се интерактивним системом обезбедити подршка да се понови поступак одређивања разлика међу концептуалним моделима. Суштина поновљеног поступка је у томе што би се искористила чињеница да за један

формални контекст може постојати више различитих, а исправних, линеарно сортирана листа формалних концепата.

4. Провера сличности логичких модела

Дизајн базе података почиње са креирањем концептуалног модела посматраног система. То је поступак који је потпуно независтан од имплементационих детаља као што су апликативни програми, програмски језици, хардверска платформа, па чак независтан и од циљног система за управљање базом података. Превођење креираног концептуалног модела на одређени конкретни модел података, попут релационог модела података, представља креирање логичког модела посматраног система [40].

Поступак креирања логичког модела релационих база података почиње превођењем елемената концептуалног модела у одговарајуће релације (тј. релационе шеме). Поступак превођења се састоји у примени правила којима је дефинисано шта настаје као резултат превођења одговарајућих јаких и слабих ентитета као и превођења свих врста директних и индиректних односа међу ентитетима. Исправност структуре добијених релационих шема се најпре проверава нормализацијом. Након тога се проверава да ли су сва потребна ограничења подржана креираним логичким моделом, и по потреби обавља допуњавање декларативним исказима које треба да обезбеде постојање тих ограничења. На крају је потребно потврдити да ли креирани логички модел може

да подржи све врсте манипулативних акција предвиђених спецификацијом корисничких захтева. У наставку рада ће најпре бити објашњени аспекти логичких модела а потом и предлог поступака провере сличности логичких модела. Када је реч о постојећим решењима, она су дата у делу који се односи на упите, с обзиром да је то аспект коме је посвећено највише пажње у истраживачким круговима.

4.1. Преглед постојећих решења и аспекти логичких модела

Уколико се жели провера сличности логичких модела релационих база података, неопходно је дефинисати поступке којима би се проверили сви потребни аспекти. Конкретно то значи да је потребно посматрати како манипулативне тако и декларативне аспекте логичког модела. Манипулативни аспект покрива све потребне начине за дохватање података и пре свега се односи на упите над базом података. Манипулативни аспект такође обухвата и начине за одржавање података, конкретно начине за додавање нових, измену постојећих и уклањање старих података у бази [41]. Као универзални начин за манипулисање подацима користи се структурни упитни језик SQL, али се, када је реч о дохватању података, посебна пажња посвећује и формалним упитним језицима који се као такви посматрају као посебан аспект. Декларативни аспект покрива ограничења над релационим шемама добијених креирањем логичког модела. С обзиром да структура релационих шема представља најбитнији део декларативног аспекта, посебна пажња се посвећује нормализацији која се као таква веома често посматра као посебан аспект приликом провере сличности логичких модела.

У наставку овог поглавља најпре ће, у првом делу, бити обрађен аспект заснован на структурно-интегритетској компоненти који обједињује ограничења над

релационим шемама, дакле декларативни аспект логичког модела и аспект нормализације. Након тога ће, у другом делу, бити обрађен манипулативни аспекти провере сличности логичких модела. У оквиру овог дела биће приказан манипулативни аспект ажурирања, али ће већа пажња бити посвећена манипулативном аспекту упита. Упити ће покрити не само аспект формалних упитних језика већ и упите дате SQL исказима.

4.1.1. Аспект заснован на структурно-интегритетској компоненти

Структурну компоненту логичког модела релационе базе података чине појмови атрибута и домена, шеме релације, релације, шеме релационе базе података, релационе базе података и NULL вредности.

Атрибут A_i је именовано својство које има значење, има вредност и не може се растављати на делове без губитка значења. Домен $D_i = dom(A_i)$ је скуп свих могућих вредности неког атрибута A_i. Активни домен D_i је скуп стварних вредности неког атрибута A_i и као такав је подскуп домена D_i.

Шема релације $R = \{A_i\}$ је коначни непразан скуп атрибута различитих назива и коначан скуп O ограничења над тим атрибутима. Шема релације се представља у облику $R(A_1, ..., A_i, ..., A_n)$. Број атрибута n представља кардиналност шеме релације.

За шему релације $R(A_1, ..., A_i, ..., A_n)$ торка вредности (скраћено: торка) представља један низ вредности атрибута: $t = <a_1, ..., a_i, ..., a_n >$. Релација r над шемом релације R је коначан скуп торки, при чему важи ($\times$ је симбол за Декартов производ, а D_i је домен): $r \subseteq D_1 \times , ..., \times D_i \times, ..., \times D_n$.

Шема релационе базе података S је коначан скуп шема релација $\{R_j\}$ и коначан скуп ограничења U која важе између њих. Релациона база података B је коначан скуп релација $\{r_j\}$ над шемом релационе базе података $S=\{R_j\}$.

Постоје ситуације када за неки ентитет у окружењу неко његово својство нема дефинисану вредност. С обзиром да шема релације и релација у оквиру модела података представљају класу ентитета, а торка у релацији инстанцу те класе, следи да се недефинисана вредност може појавити и у торци неке релације. Недефинисана вредност атрибута у торци релације може се јавити у два случаја:

- вредност постоји, али није била позната у тренутку настанка торке,
- вредност за конкретну торку нема смисла; ово се дешава када се при састављању модела података неког система спроводи компромисно реструктурирање.

С обзиром на претходно, у саставу структурне компоненте релационог модела података је и универзална вредност NULL са значењем "недефинисано" која се може применити на атрибут било ког домена.

4.1.1.1. Интегритет података

Интегритетска компонента релационог модела података служи за представљање ограничења која важе над вредностима појединих атрибута у торкама релација. Из разлога прегледности, у објашњењима која следе биће употребљавани термини дефинисани у оквиру SQL. Та ограничења се јављају у три вида:

- идентификациони интегритет: ограничења која произилазе из уникатности торки у релацијама,

- референцијални интегритет: ограничења која укључују атрибуте који се могу налазити у различитим шемама релација,
- зависности: ограничења која укључују атрибуте унутар исте шеме релације.

Ограничења идентификационог интегритета произилазе из скуповне природе дефиниције релације која подразумева уникатност торки. У оквиру тога уводе се појмови супер-кључа, кандидат-кључа, примарног кључа и интегритета примарног кључа.

Супер-кључ S шеме релације R је сваки њен подскуп атрибута X који једнозначно одређује торке у релацији r, односно за који важи да се једна вредност низа атрибута x може појавити у p највише једанпут. Формално изражено: $S = \{X \mid X \subseteq R \wedge X \rightarrow R\}$. Свака шема релације има бар један супер-кључ.

Кандидат кључ K шеме релације R је сваки супер-кључ који има особину да ни један његов прави подскуп није супер-кључ. Формално изражено: $S = \{X \mid X \subseteq R \wedge X \rightarrow R \wedge \neg\exists Y(Y \subset X \wedge Y \rightarrow R)\}$.

Примарни кључ P шеме релације R је кандидат-кључ који је одабран за потребу идентификације торки у релацији r.

Интегритет примарног кључа дефинише да ни један атрибут шеме релације R који је у саставу њеног примарног кључа P не може имати NULL вредност у релацији r. Формално изражено: $\neg\exists A_i(A_i \subset P \wedge A_i = NULL)$.

Са друге стране, референцијални интегритет каже да у релационој бази података једини начин успостављања везе између торки из различитих релација је путем вредности атрибута.

Страни кључ шеме релације R је сваки њен подскуп атрибута F за који важи да у релацији r његова дефинисана вредност мора имати вредност кандидат-кључа у некој релацији. Могуће је да се и страни кључ и реферисани кандидат-кључ налазе у истој шеми релације.

Страни кључ у некој релацији r може имати једну од две вредности:

- вредност кандидат-кључа у некој релацији,
- вредност NULL, ако није забрањена у скупу ограничења O шеме релације R.

Нека су r и s релације над шемама релација R и S и нека се у S налази страни кључ F који реферише примарни кључ P у R. При иницирању уклањања једне вредности атрибута P у релацији r (операцијом уклањања торке из релације r или измене над атрибутима P у некој торки релације r), у случају да та вредност постоји као вредност F у релацији s, могућ је један од три исхода:

- RESTRICT: операција се не извршава,
- CASCADE: операција се извршава, а при томе се из релације s уклањају све торке код којих F има уклоњену вредност P,
- SET NULL: операција се извршава, а при томе се у релацији s у свим торкама код којих F има уклоњену вредност P та вредност замењује са вредношћу NULL.

У имплементацији релационих база података уведен је и четврти исход, а то је да се уместо NULL вредности (као у трећем исходу) као нова вредност за F узима DEFAULT вредност задата у дефиницији шеме релације S.

4.1.1.2. Нормалне форме

Ограничења која су разматрана у претходном делу била су уникатност атрибута у шеми релације, односно уникатност торки у релацији, потом услов идентификационог интегритета за примарни кључ, као и услов референцијалног интегритета за страни кључ, док је садржај саме релације у погледу вредности осталих атрибута био потпуно произвољан. Међутим, између атрибута шеме релације могу постојати одређене зависности које ограничавају вредности тих атрибута у торкама релације. Постојање одређених зависности између атрибута неке шеме релације, може довести до низа непогодности, такозваних аномалија, па се таква шема релације сматра да има лошу структуру. У оквиру теорије која се бави проучавањем различитих врста зависности, дефинисане су нормалне форме које представљају критеријум ваљаности шеме релације, као и поступци нормализације, односно декомпозиције шеме релације са лошом структуром на две или више шема релација које су све у складу са жељеном нормалном формом.

Основни критеријум јесте да се једна информација треба чувати само на једном месту у релационој бази података, односно не сме бити редундантног чувања информација. На пример, уколико над неком шемом релације постоје ситуације да једној вредности неког атрибута који није примарни кључ одговара тачно једна вредност неког другог атрибута, долази до вишеструког појављивања комбинације вредности ових два атрибута, односно долази до појаве редундансе. Овај недостатак шеме релације се огледа у проблемима вишеструког уношења (исти информација се редундантно уноси), вишеструког мењања (уколико се жели промена неке информације, неопходно је то учинити на свим местима где се редундантно чува)и вишеструког уклањања (уколико се жели уклањање неке

информације, неопходно је то учинити на свим местима где се редундантно чува). Уз наведене недостатке ажурирања, присутна су и два драстична недостатка, аномалија уношења и аномалија уклањања, чији је карактер егзистенцијалне природе. Наиме, постојање редундансе доводи до тога да уношење једне информације захтева претходни унос неке друге информације, или да уклањање једне информације доводи до уклањања и друге информације, иако то није било жељено понашање.

Решење недостатака изазваних редундантним чувањем информација, постиже се декомпозицијом шеме релације, односно растављањем шеме релације на две или више нових шема релација. Сам процес декомпозиције је двојак, односно декомпонују се и шема и релација над њом. Уколико су R и r шема релације и релација које желимо да декомпонујемо на шеме релација R_1 и R_2, односно релације r_1 и r_2, онда декомпозиција треба да испуни три услова. Декомпозиција, најпре, треба да обезбеди очување атрибута односно треба да важи $R_1 \cup R_2 = R$, потом да обезбеди очување података односно треба да важи $\pi_{R1}(r) \bowtie \pi_{R2}(r) = r$, и на крају декомпозиција треба да је реверзибилна односно скуп заједничких атрибута насталих релација треба да буде кандидат-кључ у бар једној од насталих релација, $R_1 \cap R_2 \rightarrow R_1 \vee R_1 \cap R_2 \rightarrow R_2$.

Међутим, декомпозиција једне шеме релације може се спровести на више начина. Да ли су све могуће декомпозиције равноправне или не, зависи од тога које критеријуме квалитета испуњавају, односно у којој се нормалној форми налазе релационе шеме након декомпозиције, и да ли су приликом декомпозиције очуване све зависности које су постојале у полазној шеми релације. Дакле,

декомпозиција се спроводи у складу са зависностима које важе међу атрибутима, и у том смислу разликују се три основне врсте зависности, функцијске (functional) и вишезначне (multivalued) зависности, као и зависности спајања (join dependency). Функцијска зависност неког атрибута Y од неког скупа атрибута X, говори да једној комбинацији вредности атрибута у скупу X одговара тачно једна вредност атрибута Y. За функцијску зависност се каже да је тривијална уколико она дефинише зависност неког атрибута Y од неког скупа атрибута X који укључује тај атрибут, односно уколико је Y подскуп X. Вишезначна зависност неког атрибута B од неког скупа атрибута A, говори да једној комбинацији вредности атрибута у скупу A одговара скуп вредности атрибута B. За вишезначну зависност се каже да је тривијална уколико скуп атрибута A и од њега вишезначно зависган скуп атрибута B заједно обухватају све атрибуте релационе шеме над којом је та вишезначна зависност дефинисана, или уколико је B подскуп A. Зависност спајања ограничава скуп могућих релација над посматраном релационом шемом. За релацију r над шемом R се каже да задовољава зависност спајања дефинисану скупом релационих шема R_1, ..., R_n, уколико се може увек реконструисати спајањем релација r_1, ..., r_n насталих декомпозицијом. За зависност спајања се каже да је тривијална уколико бар једна од релација r_i којом је дефинисана зависност обухвата све атрибуте релационе шеме над којом је та зависност спајања дефинисана.

Дакле, циљ декомпозиције јесте да се постигне одређена нормална форма, односно да се испуне одређени критеријуми квалитета. Свака нормална форма дефинише критеријуме који говоре о степену рањивости релационе шеме по питању логичких неконзистентности и аномалија. Што је релациона шема у већој

нормалној форми, утолико је њена рањивост мања. Нормална форма се посматра на нивоу сваке релационе шеме, али се за читаву релациону базу података каже да је у некој нормалној форми, уколико су све релационе шеме те базе у тој нормалној форми. У табели 4.1 је дат преглед нормалних форми дефинисаних за релационе шеме, односно наведен је назив сваке од њих, аутор и дефиниција која наводи критеријуме која посматрана форма испуњава.

Табела 4.1 Преглед нормалних форми, ауторства и значења

Нормална форма	**Први пут дефинисано**	**Значење**
Прва нормална форма (1NF)	E.F.Codd (1970) [42]	Атрибути су примитивног типа (немају структуру или понављајуће групе)
Друга нормална форма (2NF)	E.F.Codd (1971) [43]	Ниједан не-кључни атрибут није функцијски зависан од дела кандидат-кључа (скупа атрибута који је прави подскуп неког кандидат-кључа)
Трећа нормална форма (3NF)	E.F.Codd (1971) [43]	Сваки не-кључни атрибут је директно (не транзитивно) функцијски зависан од сваког кандидат-кључа у шеми.
Нормална форма елементарног кључа (EKNF)	C.Zaniolo (1982) [44]	Свака не тривијална функцијска зависност у шеми је или зависност од

Нормална форма	Први пут дефинисано	Значење
		супер-кључа (супер-кључна) или је зависност од елементарног кључног атрибута (један атрибут који је кандидат кључ)
Бојс-Кодова нормална форма (BCNF)	R.F.Boyce, E.F.Codd (1974) [45]	Свака не тривијална функцијска зависност у шеми је зависност од супер-кључа (супер-кључна).
Четврта нормална форма (4NF)	R. Fagin (1977) [46]	Свака не тривијална вишезначна зависност у шеми је зависност од супер-кључа.
Пета нормална форма (5NF - PJNF)	R. Fagin (1979) [47]	Свака не тривијална зависност спајања у шеми је имплицирана супер-кључем те шеме.
Домен-кључ нормална форма (DKNF)	R. Fagin (1981) [48]	Свако ограничење над шемом је логичка последица доменских ограничења шеме и кључних ограничења.
Шеста нормална форма (6NF)	C.J.Date, H.Darwen, N.Lorentzos (2004) [49]	Шема не поседује не тривијалне зависности спајања.

4.1.2. Аспект заснован на манипулативној компоненти

Манипулативну компоненту релационог модела података чине два дела. Први, манипулативна компонента ажурирања, намењен је одржавању садржаја релација у релационој бази података. Други део, манипулативна компонента упита, служи за увид у садржај релација у релационој бази података.

4.1.2.1. Манипулативна компонента ажурирања

У оквиру релационог модела података постоје три операције ажурирања: унос торке у релацију, измена торки у релацији и уклањање торки из релације. У даљем разматрању сматра се да се те операције примењују на релацију r над шемом релације $R(A_1, ..., A_i, ..., A_n)$.

За унос једне торке у релацију r користи се операција уноса INSERT дефинисана са $Ins<v_1, ..., v_i, ..., v_n> (r)$, где је v_i константа типа који одговара домену колоне или NULL. Уместо задавања NULL вредности, она може да се изостави с тим да се зарез задржава, због правилне интерпретације позиције осталих вредности.

Измена торки у релацији r се остварује путем операције измене UPDATE дефинисана са $Upd_{P(X)}<a_j=v>(r)$, где су a_j атрибут, v константа а P(X) предикат над подскупом атрибута R. При томе атрибут a_j добија вредност v у оним торкама релације за које је задовољен предикат P.

За уклањање торки из релације r користи се операција уклањања DELETE дефинисана са $Del_{P(X)}(r)$, где је P(X) предикат над подскупом атрибута R. Уклањају се оне торке релације за које је задовољен предикат P.

4.1.2.2. Манипулативна компонента упита

Формални упитни језици, представљају формалну основу манипулативног компоненте упита релационог модела података [42] и могу се поделити и две основне групе: релациона алгебра и релациони рачун. Релациона алгебра представља процедуралне поступке за спецификацију упита над базом података, док релациони рачун дефинише различите декларативне поступке за спецификацију упита над базом података. Оригинални релациони рачун [50] дозвољава да променљиве узимају вредности над датом релацијом, чиме је дефинисано да вредности могу да долазе само из торки присутних у тој релацији. Ова врста релационог рачуна је позната под називом релациони рачун торки. Друга варијанта релационог рачуна је релациони рачун домена, код кога променљиве могу да узимају вредности над посматраним доменом [51].

Област еквиваленције упита има велики значај у примени релационих база података, посебно у домену побољшања перформанси рада релационих база података. Комплексност бата података расте како у погледу броја релација, тако и у погледу количине података у њима. Могућност одређивања еквивалентних упита који се оптималније извршавају одувек је привлачио велику пажњу истраживача. Ранија истраживања су била усмерена на проверу сличности посебне групе упита, назване коњуктивни упити. Иако се ради о ограниченом облику упита, велики број упита у свакодневној примени може се сврстати у ову групу упита. Коњуктивним упитима одговарају SQL упити који су у форми SELECT-FROM-WHERE и при чему WHERE клаузула може да садржи само атомске услове једнакости који се искључивао састоје од назива колоне изједначене са константом, а међусобно се комбинују употребом AND логичком

операцијом. Дакле, искључени су упити који садрже употребу агрегатних функција и подупита. Наиме, реч је ограниченом облику упита који одговара предикатском рачуну првог реда који се могу описати скупом формула конструисани на основу атомских формула користећи конјункцију и егзистенцијални квантификатор, док су дисјункција, негација и универзални квантификатор забрањени. Коњуктивни упити имају генералну форму

$$(x_1, x_2, \dots, x_k)\exists(x_{k+1}, \dots, x_n)A_1 \vee A_2 \dots \vee A_n$$

Где $X_1 \dots X_k$ представљају слободне променљиве, $X_{k+1} \dots X_n$ везане променљиве, а $A_1 \dots A_n$ атомске формуле. Уколико коњуктивни упит нема везаних променљивих онда се назива логички коњуктивни упит.

Са циљем повећања изражајне моћи, могућа су проширења коњуктивних упита употребом операције уније, што одговара позитивној релационој алгебри, или чак и употребом негације чиме се по питању изражајне моћи постиже еквивалент релационој алгебри, као што је дато Кодовом теоремом [52]. Међутим, поменута проширења доводе до појаве тешко израчунљивих проблема и неодлучивих ситуација, који су у случају коњуктивних упита израчунљиви [53].

Проучавање комплексности израчунљивости и евалуације коњуктивних упита обухвата два проблема. Први проблем се односи на евалуацију коњуктивних упита када се као улазне информације посматрају не само упит, већ и релациона база података над којом се упит извршава, што се назива комбинована комплексност. Док се други проблем назива комплексност података, и односи се на евалуацију упита над релационом базом података, када је упит фиксан и непроменљив. Коњуктивни упити су НП-потпуни са аспекта комбиноване

комплексности, док им је сложеност мала са аспекта комплексности података и израчунљиви су у полиномијалном времену [54].

Основни проблем у одређивању еквиваленције упита јесте провера садржаности упита. Сматра се да је упит садржан у неком другом упиту уколико, посматрано над неком базом података, увек враћа резултат који је садржан у резултату тог другог упита. Проблем садржаности упита представља НП-потпун проблем када је реч о коњуктивним упитима. Показано је да овај проблем еквивалентан проблему евалуације упита [55].

Важну класу коњуктивних упита који имају полиномијалну комбиновану сложеност представљају ациклични коњуктивни упити [56]. Ацикличност коњуктивних упита је структурна особина упита која се дефинише користећи хипергграф посматраног упита. Упит је ацикличан уколико је уколико за његов хипергграф важи да је ширина одговарајућег хиперстабла једнака један. У специјалним случајевима када се посматрају упити над релацијама које су бинарне, онда је коњуктивни упит ацикличан ако и само ако је ацикличан и граф зависности променљивих које се користе у упиту. Битна генерализација ацикличности постоји и у смислу ограничене ширине хиперстабла, аналогно ограниченој ширини стабла код графова. Показано је да коњуктивни упити са ограниченом ширином стабла имају логаритамску комбиновану комплексност [57], док неограничени коњуктивни упити над подацима у стаблу имају полиномијалну комбиновану сложеност [58].

Истраживања у области еквиваленције коњуктивних упита обухватају резултате у случајевима релационих шема са дефинисаним кључевима [59]. Такође, постоје

резултати и када је реч о коњуктивним упитима са агрегатним функцијама. Конкретно резултати који се односе на проблем одређивања реформулација минималне величине у присуству уграђених ограничења [60], односно максимално садржаних скупова реформулација коњуктивних упита са агрегатном функцијом пребројавања [61]. Допринос ових радова је такође и у приближавању теоретских разматрања конкретним имплементацијама у виду SQL, између осталог и због тога што се поред скуповне семантике посматра и вишескуповна семантика (семантика торбе – bag, multiset).

Генерално посматрано, манипулативна компонента упита постоји у виду процедуралног и декларативног дела. Суштина процедуралне манипулативне компоненте упита је формулација како се долази до решења упита. У ту сврху се користи релациона алгебра са својим скупом основних и додатних операција. Основе релационе алгебре су изложене у наставку овог поглавља. Суштина декларативне манипулативне компоненте упита је формулација шта је решење упита и за то се користе две варијанте релационог рачуна, релациони рачун домена и релациони рачун торки. Велики број доступних системи за управљање базама података засновани су на релационом моделу, међутим језик који се у њима користи је структурирани упитни језик (SQL), који почива на формалним принципима релационог модела, али их не задовољава у потпуности. Другим речима, иако формалну основу за SQL представљају формални упитни језици, они нису директно подржани у доступним системи за управљање базама података.

4.1.2.2.1. Релациона алгебра

Основне операције релационе алгебре чини скуп од 8 операција од којих су 5 елементарне а преостале 3 су изведене из њих, који је касније проширен додатном

операцијом преименовања. Као што је приказано у табели 4.2, а на основу [41] основне операције релационе алгебре могу се класификовати:

- Према сложености на елементарне и изведене,
- Према броју операнада на унарне и бинарне,
- Према врсти на посебне и скуповне.

Табела 4.2 Преглед операција релационе алгебре

Симбол	Назив	Сложеност	Број операнада	Врста
σ	рестрикција	елементарна	унарна	посебна
π	пројекција	елементарна	унарна	посебна
∪	унија	елементарна	бинарна	скуповна
−	разлика	елементарна	бинарна	скуповна
×	Декартов производ	елементарна	бинарна	посебна
∩	пресек	изведена	бинарна	скуповна
$\times_{args}$	спајање	изведена	бинарна	посебна
/	дељење	изведена	бинарна	посебна
ρ	преименовање	елементарна	унарна	посебна

Рестрикција (симбол σ) је елементарна, унарна и посебна операција која из полазне релације по задатом критеријуму издваја подскуп торки. Критеријум је неки логички израз који је израчунљив над сваком торком. Добијена релација има исту структуру као и полазна. Ако је p релација над шемом R(X), а P(X) услов рестрикције, формална дефиниција операције рестрикције гласи:

$$\sigma_{P(X)}(r) = t(X) = \{t | x \in r \wedge P(X)\}$$

Структура релације се не мења - шема резултата одговара шеми операнада. За кардиналност релација N, односно број торки у релацијама, важи: N(t) ≤ N(r).

Пројекција (симбол π) је елементарна, унарна и посебна операција која из полазне релације по задатом скупу атрибута формира нову релацију као скуп торки над тим атрибутима. Задати скуп атрибута је најчешће подскуп скупа атрибута полазне релације, а вредности атрибута у торкама настале релације одговарају онима у торкама полазне релације. Ако је r релација над шемом R(X), Y задати скуп атрибута, а x и y торке над X и Y, формална дефиниција операције пројекције гласи:

$$\pi_Y\,(r) = t(Y) = \{t | Y \subseteq X \wedge y \in x\}$$

при чему су одговарајуће шеме релација R(X) и T(Y). Структура релације се мења и одређује је задати скуп атрибута Y. За кардиналности релација важи N(t) ≤ N(r).

Унија (симбол ∪) је елементарна, бинарна и скуповна операција која из две полазне релације формира нову која садржи све торке које се налазе у било којој или евентуално у обе полазне релације. Ова операција је могућа само између релација које су унијски еквивалентне, односно имају исте атрибуте по броју, редоследу, називу и домену. Блаже формулисан услов је да су шеме релација унијски компатибилне, односно да важи да шеме релација имају исти број атрибута и да атрибути шема релација редом одговарају једни другим по доменима.

Ако је су r и s релације над шемама R(X) и S(X), где X означава унијски компатибилни скуп атрибута у обе релације, формална дефиниција операције уније гласи:

$$r \cup s = t(X) = \{x | x \in r \vee x \in s\}$$

Сагласно особини скупова, свака торка која је присутна у обе полазне релације јавља се само једном у резултантној релацији. За кардиналности релација важи:

max (N(r),N(s)) ≤ N(t) ≤ N(r) + N(s).

Разлика (симбол –) је елементарна, бинарна и скуповна операција која из две полазне релације формира нову која садржи све торке прве релације које се не налазе у другој релацији. Ова операција је могућа само између унијски еквивалентних или бар компатибилних релација. Ако је су r и s релације над шемама R(X) и S(X), где X означава унијски компатибилни скуп атрибута у обе релације, формална дефиниција операције разлике гласи:

$$r - s = t(X) = \{x | x \in r \wedge x \notin s\}$$

За кардиналности релација важи: 0 ≤ N(t) ≤ N(r).

Декартов производ (симбол x) је елементарна, бинарна и скуповна операција која из две полазне релације формира нову, са торкама добијеним тако што се свака торка из прве релације редом “спој” са сваком торком друге релације, при чему шема настале релације садржи редом све атрибуте полазних релација.

Ако су r и s релације над шемама R(X) и S(Y), где су X и Y скупови атрибута у шемама релација, формална дефиниција операције Декартовог производа гласи:

$$r \times s = t(XY) = \{xy | x \in r \wedge y \in s\}$$

Стриктно посматрано, Декартов производ није могућ ако у релацијама постоји бар један атрибут истог назива, али ако се за пуни назив атрибута усвоји назив

релације и назив атрибута раздвојени тачком, тада је Декартов производ могућ увек осим када се примењује између једне исте релације. У тој, а и неким другим ситуацијама конфликта назива примењује се операција преименовања.

Шема резултантне релације садржи све атрибуте полазних релација. За кардиналности релација важи: $N(t) = N(r) \cdot N(s)$.

Пресек (симбол ∩) је изведена, бинарна и скуповна операција која из две полазне релације формира нову која садржи све торке прве релације које се налазе и у другој релацији. Ова операција је могућа само између унијски еквивалентних или бар компатибилних релација. Ако је су r и s релације над шемама R(X) и S(X), где X означава унијски компатибилни скуп атрибута у обе релације, формална дефиниција операције пресека гласи:

$$r \cap s = t(X) = \{x | x \in r \wedge x \in s\}$$

Пресек је као изведена операција дефинисан преко операције разлике

$$r \cap s = r - (r - s)$$

За кардиналности релација важи: $0 \leq N(t) \leq \min(N(r), N(s))$.

Спајање (симбол x_{args}) је изведена, бинарна и посебна операција која из две полазне релације формира нову, са торкама добијеним у два корака. Најпре се свака торка из прве релације редом се спаја са свим торкама из друге релације, а потом из тако добијених торки издвајају се оне које задовољавају задати услов (args).

Нека су r и s релације над шемама R(X) и S(Y), где су X и Y скупови атрибута у шемама релација. Формална дефиниција операције спајања гласи:

$$r \times_{P(XY)} s = \sigma_{P(XY)}(r \times s) = t(XY) = \{xy | x \in r \wedge y \in s \wedge P(XY)\}$$

Оваква дефиниција операције спајања дозвољава произвољни услов P, под условом да је израчунљив за сваку торку релације настале Декартовим производом полазних релација. У пракси се најчешће јавља одређени број специјалних случајева, у смислу форме услова P.

Под тета-спајањем, у ознаци r $x_{(Xi\ \Theta\ Yk)}$ s, се подразумева спајање код кога симбол Θ означава било који од оператора поређења =, ≠, ≤, ≥ . При том су r и s релације над шемама R(X) и S(Y), а Xi и Yk атрибути за које важи Xi∈X и Yk∈Y. Ако уместо по једног атрибута из обе релације у услову спајања учествују подскупови са истим бројем атрибута, оператор поређења се примењују између парова атрибута одговарајући број пута и у логичкој конјункцији. У пракси је најчешћи случај да се употребљава услов једнакости атрибута по којима се врши спајање, то је такозвано екви-спајање, у ознаци r $x_{(Xi=Yk)}$ s. Релација настала екви-спајањем садржи сувишне атрибуте. Наиме, вредности атрибута из једне и друге релације по којима се врши спајање увек су исте, па се додатном операцијом пројекције елиминишу непотребни атрибути. Ово је толико чест случај у пракси да је уведена специјална операција природног спајања, у ознаци ⋈, која подразумева секвенцу три елементарне операције релационе алгебре. Најпре се обавља Декартов производ релација, потом рестрикција по услову једнакости спојних атрибута у обе релације, а на крају и пројекција по разлици уније свих атрибута и скупа спојних атрибута из било које од релација.

Дељење (симбол /) је изведена, бинарна и посебна операција. Нека је r релација шеме R(XZ), а s релација шеме S(Y), где су X и Y дисјунктни скупови атрибута, а Y и Z унијски компатибилни скупови атрибута у смислу типа и значења. Дељење релације r са релацијом s се дефинише као

$$r/s = \pi_X(r) - \pi_X((\pi_X(r) \times s) - r) = t(X)$$

Као резултат операције дељења добијају се оне вредности у r јављају у комбинацији са свим вредностима из релације s. Другим речима, операција дељења нам даје оне вредности у релацији r које "покривају" скуп вредности задат релацијом s.

У случају неколико основних операција релационе алгебре указано је на ситуације конфликта у вези назива релација и назива атрибута који чине поједине операције немогућим. То намеће допуну релационе алгебре операцијом преименовања, што је и учињено увођењем додатне елементарне, унарне и посебне операције преименовања, у ознаци ρ, назива атрибута релације над којом врши преименовање са очувањем садржаја, односно

$$\rho_{Y1/A1\ldots Yn/An}(r) \rightarrow (Y1, \ldots, Yn)$$

Операнд преименовања може бити осим релације и неки израз релационе алгебре. За кардиналности релација важи: N(s) = N(r).

4.1.2.2.2. Релациони рачун домена

Релациони рачун домена је прва од две варијанте релационог рачуна који је изведен из предикатског рачуна првог реда. Основу тог рачуна чине појмови варијабле, израза, формуле и атома, као и правила за њихово формирање.

Варијабле у изразу релационог рачуна имају вредности атрибута релација и могу бити у два вида. У виду слободне варијабле, које добијају вредности на основу израза и као такве заједно представљају резултат упита, и у виду везане варијабле, од којих свака варира тако што узима вредности из активног домена (постојећег скупа вредности) једног атрибута једне релације, и које се у изразу јављају искључиво са универзалним и егзистенцијалним квантификаторима {∀, ∃}.

Израз релационог рачуна домена је општег облика $\{<d_1,\ldots, d_i, \ldots d_n>|P(d_1, \ldots, d_i, \ldots, d_m, s_1, \ldots, s_j, \ldots, s_n)\}$

где d_i представља слободну варијаблу која учествује у резултату упита, s_j везану варијаблу, а P је формула типа предиката. Формула релационог рачуна, P, домена F је дефинисана правилима формирања:

- сваки атом је формула;
- ако је F формула, онда су ¬F и (F) формуле;
- ако су F1 и F2 формуле, онда су F1∨F2 и F1∧F2 формуле;
- ако је F(s) формула, где је s везана варијабла, онда су ∃s(F(s)) и ∀s(F(s)) формуле.

Ако се са Θ означи оператор поређења из скупа оператора {=, ≠, ≤, ≥, <, > } тада су могуће форме атома релационог рачуна домена:

- <x1, ..., xi, ..., xk> ∈r , где је r релација са k атрибута, а варијабле xi су редом над активним доменима тих атрибута, при чему варијабле не варирају независно над одговарајућим доменима, него у целини, над вредностима које чине једну торку у r,

- x Θ y , где су x и y варијабле на које је примењив оператор Θ,
- x Θ C , где је x варијабла, а C константа, на које је примењив оператор Θ.

Ради концизности и прегледности, уместо израза $\exists x_1(\exists x_2(... \exists x_k (P(x_1, x_2, ...,x_k)...)))$, користи се скраћена форма $\exists x_1, x_2, ..., x_k(P(x_1,x_2, ... ,x_k))$. Структура резултата упита одређена конструкцијом наведених слободних променљивих $<d_1,\ldots, d_i, \ldots d_n>$ у упиту релационог рачуна домена.

4.1.2.2.3. Релациони рачун торки

Релациони рачун торки је друга од две варијанте релационог рачуна који је изведен из предикатског рачуна првог реда. Основу тог рачуна чине појмови варијабле, израза, формуле и атома, као и правила за њихово формирање.

Варијабле у изразу релационог рачуна имају вредности целих торки и могу се јављати у два вида, слободне и везане. Само једна од варијабли је слободна, добија вредност на основу израза, и као таква представља резултат упита. Све остале варијабле су везане и свака од њих варира тако што узима вредности из активног, постојећег, скупа вредности торки из једне релације, и у изразу се јављају искључиво са универзалним и егзистенцијалним квантификаторима {∀, ∃}.

Израз релационог рачуна торки је општег облика $\{ t|P(t, s_1, \ldots, s_i, \ldots,s_n)\}$ где t представља слободну варијаблу која је резултат упита, s_i везану варијаблу, а P је формула типа предиката. Формула релационог рачуна торки, P, је дефинисана правилима формирања:

- сваки атом је формула;
- ако је F формула, онда су ¬F и (F) формуле;

- ако су F1 и F2 формуле, онда су F1∨F2 и F1∧F2 формуле;
- ако је F(s) формула, где је s везана варијабла, онда су ∃s(F(s)) и ∀s(F(s)) формуле.

Ако се са Θ означи оператор поређења из скупа оператора {=, ≠, ≤, ≥, <, > } тада су могуће форме атома релационог рачуна торки:

- u ∈ r , где је u везана варијабла, а r релација,
- v[x] Θ w[y] , где су v и w везане варијабле, а x и y атрибути садржани у торкама v и w, а на које је примењив оператор Θ,
- t[x] = u[y] , где је t слободна, а u везана варијабла; за разлику од осталих атома који се вреднују на истинито или неистинито, овај атом је увек истинит и додељује вредност атрибуту x слободне варијабле t на основу вредности атрибута y везане варијабле u,
- v[x] Θ C , где је v везана варијабла, x атрибут садржан у торки v, C константа, при чему је над x и C примењив оператор Θ.

Структура резултата упита, односно скупа атрибута торке t, одређен је као унија свих атрибута који се појављују у атомима облика “t[x]=…”.

4.1.2.2.4. Структурни упитни језик – SQL

У структурном упитном језику, наредба SELECT за упите представља најзначајнију и најчешће коришћену наредбу за манипулацију подацима. Код сваког упита задајемо које податке тражимо као резултат, потом из којих табела то тражимо и који услов треба да задовоље подаци да би били укључени у резултат. Резултат упита не мора бити релација у смислу уникатности редова који улазе у резултат. То се испољава када за резултат упита бирамо само неке од

колона, када може доћи до појаве истоветних редова у резултату. Стога приликом формулације упита треба да постоји могућност спецификације да ли желимо елиминацију вишеструког појављивања истих редова у резултату или не.

Наредба упита се састоји од пет клаузула, SELECT, FROM, WHERE, GROUP BY и HAVING. Иза сваке од клаузула долази одговарајућа листа аргумената. Код упита се најчешће тражи приказ само одређених колона, или приказ свих колона по редоследу који је другачији од стварног. То се постиже наредбом упита код које иза SELECT клаузуле наводимо жељене колоне. Оваква наредба одговара операцији пројекције у релационој алгебри, али постоји једна битна разлика: ако то не нагласимо, у табели која настаје као приказ неће бити елиминисана вишеструка појављивања истих вредности.

У клаузули FROM специфицира се одакле долази подаци. Упити над више табела подразумевају спајање табела по неком услову и реализују се тако што се у FROM клаузули наведу више назива извора података, одвојених зарезима. Извор података може бити табела, поглед или нова угнеждена SELECT наредба. Од табела наведених у FROM клаузули формира се Декартов производ и тако се као међу резултат добија фиктивна табела која садржи све колоне свих наведених табела. Уколико постоје колоне са истим називом, онда је приликом навођења, у FROM клаузули, потребно дати надимке изворима података.

Услов укључивања редова у формирање резултата наводи се иза клаузуле WHERE. Тај услов је логички израз који је израчунљив над сваким појединим редом фиктивне табеле настале на основу FROM клаузуле. У формирање резултата упита улазе само они редови за које тај израз даје истинит резултат. Уколико је

WHERE клаузула изостављена онда се посматра као да сви редови задовољавају услов. У општем случају, услов може бити у виду простог или сложеног предиката. Прост предикат је елементарни логички израз који је израчуњив над сваким појединим редом фиктивне табеле и који се не може растављати на једноставније логичке изразе. Прост предикат је најчешће релацијски израз, али може и бити формиран на основу неких специфичних функција за проверу NULL, за проверу сличности знаковних константи (LIKE), или неког од осталих SQL специфичних функција. Сложен предикат је логички израз који је формиран из једног или више једноставнијих логичких израза применом логичких оператора AND, OR и NOT.

Понекада је за израчунавање резултата упита потребно да се редови који задовоље WHERE клаузулу, додатно групишу. У те сврхе користе се GROPU BY клаузула, иза које је потребно навести колоне по којим се врши груписање. Док се услов за укључивање сводних редова у резултат задаје у HAVING клаузули. Унутар група се могу тражити резултати посебних SQL функција које се називају сводним или агрегатним функцијама (MIN, MAX, AVG, COUNT itd). Уколико се обавља груписање, онда је у SELECT клаузули дозвољено појављивање само подскупа колона употребљених за груписање, док све остале колоне смеју бити употребљено једино као аргументи агрегатних функција, или у општем случају то могу бити изрази састављени из више таквих функција, оператора и константи.

Логички посматрано, резултат упита се израчунава тако што најпре се формира прва помоћна табела, као Декартов производ свих извора података наведених у FROM клаузули. Потом се у тако креираној првој помоћној табели, задржавају само редови који задовољавају услов наведен у WHERE клаузули. Након тога се

формира друга помоћна табела тако што се редови прве помоћне табеле сакупљају у групе са једнаким вредностима над колонама наведеним у GROUP BY клаузули. Потом се на основу сваке групе из друге помоћне табеле, формира по један нови ред који ће бити смештен у трећу помоћну табелу. Трећа помоћна табела има колоне које се јављају у GROUP BY клаузули уз додате нове колоне за које се вредност израчунава на основу агрегатних функција, и то израчунавањем над сваком од група. Ово ради за оне агрегатне функције које се јављају у HAVING или SELECT клаузули. Из треће помоћне табеле се уклањају сви они сводни редови који не задовољавају HAVING клаузулу. У резултату се из треће помоћне табеле приказују само оне колоне које су наведене у SELECT клаузули. Описани поступак објашњава логичко извршавање упита, док је реална имплементација далеко ефикаснија, у смислу редоследа спајања табеле, елиминације непотребних колона, привремено срачунатих података и осталих оптимизација.

Израчунавање и разумевање упита је утолико компликованије што се унутар упита могу користити подупити. Подупитом се назива SELECT наредбу која се налази у склопу WHERE или HAVING клаузуле и чије извршење претходи вредновању предиката у тим клаузулама. Подупити се могу класификовати по резултату којег дају и по начину извршавања у односу на "спољну" SELECT наредбу. Класификацију подупита по начину извршавања спроводимо по томе да ли његово извршавање зависи од извршавања спољног упита (упита у коме се налази) или не. По томе разликујемо две врсте подупита, некорелисани и корелисани. Некорелисани подупит чије извршавање ни на који начин не зависи од извршавања спољног упита. Овакав подупит се извршава само једном, и то на

почетку извршавања упита у коме се налази. Корелисани подупит чије извршавање зависи од извршавања спољног упита. Овакав подупит се извршава за сваки ред табеле коју обрађује спољни упит.

4.2. Предлог поступка провера сличности логичких модела

Логички модели представљају трансформацију одговарајућих концептуалних модела на релациони модел података. Управо зато се провера сличности логичких модела спроводи након обављене провере сличности концептуалних модела, односно након упаривања одговарајућих елемената, тј. након успостављеног мапирања. Предлог поступка провере сличности логичких модела се састоји у обављању два корака. Најпре да се логички модели сведу на исту репрезентацију, а потом да се обави одређивање разлика. У сваком од ова два корака, посебна пажња се посвећује сваком од аспеката, односно, структурно-интегритетском и манипулативном.

4.2.1. Репрезентација логичких модела

Велики број доступних системи за управљање базама података засновани су на релационом моделу, међутим језик који се у њима користи је структурирани упитни језик (SQL), који почива на формалним принципима релационог модела, али у одређеним ситуацијама и проширује семантику релационог модела. Управо из разлога што SQL обухвата све потребне концепте релационог модела, у овим раду је он одабран као основна за репрезентацију логичких модела.

4.2.1.1. Репрезентација структура

Логички модел настаје трансформисањем одговарајућег концептуалног модела. Сам процес трансформације се обавља по правилима која су дефинисана још при

појави релационог модела и приликом увођења првих нотација за потребе концептуалних модела [62]. Иако могу постојати различите репрезентације логичког модела, за разлику од концептуалног модела, логички модел најчешће нема већи број репрезентација.

Другим речима када се посматра логички модел релационог модела података, он је најчешће одмах дат у форми декларативних SQL исказа. Међутим, могу постојати и другачије репрезентације, које не само да су јако ређе, већ и често не поседују адекватан ниво изражајности. Управо из разлога разликовања по питању изражајности, није могуће обезбедити проверу сличности логичких модела исказаних различитим репрезентацијама.

Поред самог SQL као вида репрезентације, могући начин записа јесте у прегледнијој форми код које се поред назива релационих шема, набрајају атрибути и њихови типови, потом у виду скупа набрајају најпре функцијске зависности, а потом и вишезначне зависности које важе над тим релационим шемама. Трансформација овог вида записа је поступак који се своди на два корака. Први у коме се на основу релационих шема и њихових атрибута, дефинишу одговарајуће CREATE TABLE декларације. У овом кораку се одмах за сваки од атрибута дефинишу њихови типови, али се истовремену и декларишу примарни и страни кључеви као и одговарајућа правила динамичког референцијалног интегритета. У другом кораку се на основу скупа функцијских и вишезначних зависности дефинишу ограничења. Та ограничења могу бити на нивоу атрибута у случају елементарних суперкључних функцијских зависности, односно ограничења на нивоу табеле у свим осталим случајевима. С обзиром да је управо описани вид репрезентације далеко од ређи од оног код кога је логички

модел одмах дефинисан у форми SQL исказа, у овом раду неће бити навођени детаљи ове трансформације.

4.2.1.2. Репрезентација упита

Иако формалну основу за SQL представљају формални упитни језици, они нису директно подржани доступним системима за управљање базама података. Управо су из тог разлога у овим раду су дефинисана правила превођења исказа датих формалним упитним језицима у одговарајуће SQL исказе. За потребе превођења, обе групе формалних упитних језика, могу бити описане користећи без контекстне граматике. Другим речима, граматика код којих се сва правила превођења могу искористити без обзира на специфични контекст у коме се променљива налази, тј. сва правила превођења могу да буду дефинисана искључиво помоћу терминалних симбола на десној страни. Превођењем исказа формалних упитних језика у одговарајуће SQL исказе, омогућено је да се провера сличности исказа датих формалним упитним језицима сведе на проблем провере сличности њима одговарајућих SQL исказа.

Правила превођења исказа релационе алгебре дата су у табели 4.3. Правила превођења су дата за све елементарне операције, док су све остале изводиве из овог скупа. Уколико се у исказу комбинује већи број операција, као што је веома често случај, онда се SELECT наредба унутрашње операције јавља у FROM клаузули одговарајуће SELECT наредбе спољашње операције.

Табела 4.3 Преглед правила превођења операција релационе алгебре

Симбол	**SQL**
$\sigma_{a>0}$**(Tabela)**	SELECT * FROM Tabela WHERE a>0;

$\pi_{a,b}$(Tabela)	SELECT DISTINCT a, b FROM Tabela;
TabelaA ∪ TabelaB	SELECT * FROM TabelaA UNION SELECT * FROM TabelaB;
TabelaA − TabelaB	SELECT * FROM TabelaA WHERE NOT EXISTS(SELECT * FROM TabelaB);
TabelaA × TabelaB	SELECT * FROM TabelaA, TabelaB;
$\rho_{a/b,c/d}$(Tabela)	SELECT b as a, d as c FROM Tabela;

Основна БНФ граматика дефинисана за потребе креирање парсера за упите релационе алгебре дата је следећим правилима:

```
lista_upita ::= lista_upita upit | upit ;
upit ::= izraz SEMICOLON  | izraz ASSIGN IDENT SEMICOLON ;
izraz ::= izraz_unija | izraz_razlika | izraz_presek
                    | izraz_prirodno_spajanje | izraz_spajanje
                    | izraz_projekcija | izraz_izbor
                    | izraz_preimenovanje | IDENT ;
izraz_projekcija ::= PROJECTION SUB_START lista_atributa RCURLY LPARENT izraz
                       RPARENT;
izraz_preimenovanje ::= RENAME SUB_START pair_list RCURLY LPARENT izraz
                          RPARENT ;
pair_list ::= par | pair_list:l COMMA par ;
par ::= IDENT DIV IDENT ;
lista_atributa ::= IDENT | lista_atributa COMMA IDENT ;
izraz_unija ::= LPARENT izraz UNION izraz RPARENT ;
izraz_razlika::= LPARENT izraz MINUS izraz RPARENT ;
izraz_presek ::= LPARENT izraz INTERSECTION izraz RPARENT ;
izraz_prirodno_spajanje ::= LPARENT izraz NATURAL_JOIN izraz RPARENT ;
izraz_spajanje ::= LPARENT izraz CROSS_JOIN izraz RPARENT ;
izraz_izbor ::= SELECTION SUB_START uslov RCURLY LPARENT izraz RPARENT ;
uslov ::= prost_uslov AND uslov | prost_uslov OR uslov | prost_uslov ;
prost_uslov ::= operand operator_poredjenja operand ;
operand ::= IDENT | STRING | INTCONST ;
operator_poredjenja ::= LESS | LESS_OR_EQUAL | EQUAL | NOT_EQUAL
                    | GREATER | GREATER_OR_EQUAL ;
```

Правила превођења релационог рачуна домена састоје се у томе да се све слободне променљиве наводе у SELECT листи главног упита као листа атрибута,

док се у његовој FROM клаузули у виду листе налазе све табеле које дефинишу домене тих слободних променљивих. За сваку везану променљиву се дефинише угнеждени подупити. Подупит је оператором EXISTS, односно NOT EXISTS, везан за спољни упит, што одговара употреби егзистенцијалног оператора. Који је то конкретан спољни упит зависи од нивоа угнеждења на коме је дефинисана везана променљива. Сви додатни услови дефинисани на променљивама, било везаним или слободним, смештају се у WHERE клаузули одговарајућег упита, било главног или неког од угнеждених. Основна БНФ граматика дефинисана за потребе креирање парсера за упите релационог рачуна домена дата је следећим правилима:

```
lista_upita ::= lista_upita upit | upit ;
upit ::= LCURLY LESS lista_varijabli GREATER BAR formula RCURLY;
lista_varijabli ::= IDENT | lista_varijabli COMMA IDENT;
formula ::= atomska_formula |formula AND formula |formula OR formula
             |NOT LPARENT formula RPARENT
             |LPARENT EXISTS lista_varijabli RPARENT LPARENT formula RPARENT
             |LPARENT FORALL lista_varijabli RPARENT LPARENT formula RPARENT;
atomska_formula ::=IDENT LPARENT lista_argumenata RPARENT
                    | argument operator_poredjenja argument;
lista_argumenata ::= argument | lista_argumenata COMMA argument;
argument ::= IDENT | STRING | NUMBER;
operator_poredjenja ::= LESS | LESS_OR_EQUAL | EQUAL | NOT_EQUAL | GREATER
                        | GREATER_OR_EQUAL ;
```

На еквивалентан начин се обавља и превођење релационог рачуна торки. Једина разлика се састоји у томе на који начин се одређује шта ће се налазити у SELECT листи главног упита. Основна БНФ граматика дефинисана за потребе креирање парсера за упите релационог рачуна торки дата је следећим правилима:

```
lista_upita ::= lista_upita upit | upit ;
upit ::= LCURLY n_torka BAR formula RCURLY;
```

```
n_torka ::= IDENT;
formula ::= atomska_formula |formula AND formula |formula OR formula
             |NOT LPARENT formula RPARENT
             |LPARENT EXISTS n_torka RPARENT LPARENT formula RPARENT
             |LPARENT FORALL n_torka RPARENT LPARENT formula RPARENT;
atomska_formula ::=IDENT LPARENT lista_argumenata RPARENT
                    | argument operator_poredjenja argument;
lista_argumenata ::= argument | lista_argumenata COMMA argument;
argument ::= IDENT | STRING | NUMBER;
operator_poredjenja ::= LESS | LESS_OR_EQUAL | EQUAL | NOT_EQUAL | GREATER
                        | GREATER_OR_EQUAL ;
```

4.2.2. Одређивање разлика логичких модела

Одређивање разлика логичких модела као претпоставку има да је провера сличности на концептуалном нивоу спроведена и да је обављено упаривање одговарајућих елемената. Процес одређивања разлика посебно посматра разлике у структурном аспекту модела, а посебно у манипулативном аспекту модела.

4.2.2.1. Одређивање разлика између структура

Полазна претпоставка је да је провера сличности на концептуалном нивоу обављена пре почетка провере сличности на логичком нивоу. Без смањена општости претпоставља се, такође, да је након провере сличности упаривање елемента обављено тако да постоји пресликавање између одговарајућих елемената логичког модела, и да се трансформација једног у други може обавити одговарајућим преименовањем. Самим тим, провера постојања разлике у структурама логичких модела, се своди на проверу интегритетске компоненте логичког модела. Другим речима, приликом провере разлика у структури логичких модела, та провера своди на проверу интегритетске компоненте логичког нивоа.

Провера разлика између логичких модела по питању интегритетске компоненте, обавља се тако што се посматра понашања модела, односно одзив модела на побуду. Као побуда, у овом случају, се користи манипулативна компонента ажурирања, односно провера се обавља дефинисањем одговарајућих група исказа које садрже комбинације INSERT, UPDATE и DELETE наредби.

Дефиниција 3.0: Нека су дати логички модели M и M'. Модели се сматрају еквивалентним, односно M≡M', уколико за сваки скуп исказа S, састављен од наредби убацивања, ажурирања и брисања $S=\{s_1,\ldots s_n\}$ где је $n>0$ и $s_i \in \{Ins,Upd,Del\}$, оба модела(тј. базе података дефинисане над њима) враћају исти резултат на сваки могући упит над њима $\{\forall S, Q|Q^M \equiv Q^{M'}\}$.

У складу са дефиницијом, потребно је проверити да ли је након извршеног скупа манипулативних исказа садржај обе базе података идентичан. Тако се ради провере:

- важења идентификационог интегритета, користи скуп који се састоји од низа исказа убацивања, како би се проверило да ли је за идентификацију употребљен исти скуп атрибута,
- важења референцијалног интегритета, користи скуп који се састоји од низа исказа убацивања, ажурирања и брисања; искази убацивања ради провере статичког референцијалног интегритета, а искази ажурирања и брисања ради провере акција динамичког референцијалног интегритета,
- важења различитих ограничења која могу да представљају не само функцијске, већи друге типове зависности, али и неких других типова

ограничења која су проистекла из природе модела (доменска ограничења), користи скуп који се састоји од низа исказа убацивања.

Овим проверама су обухваћени и специјални случајеви који нису експлицитно побројани. На пример, постојање ограничења за кандидат кључеве се може посматрати као специјални случај идентификационог интегритета, а потпуно еквивалентно и као специјални случај ограничења с обзиром да постојање кандидат кључева проистиче из постојања функцијских зависности. Такође, ограничења по питању дозвољавања да се појави NULL у неком атрибуту, се проверавају на исти начин као и сва остала ограничења, користећи исказ убацивања, али са изостављеном вредношћу за тај атрибут.

4.2.2.2. Одређивање разлика између упита

Досадашњи резултати у области еквиваленције упита, као што је показано, немају решење за најгенералнији облик SQL упита. Конкретно, упити са груписањем и агрегатним функцијама, као и упити са подупитима, како некорлисаним тако и корелисаним подупитима, још увек нису адекватно подржани формалним проверама еквиваленције. Управо из тог разлога, еквиваленција упита у овом раду посматра се аспекта датог помоћу следеће две дефиниције.

Дефиниција 3.1: Нека су дати упити Q и Q'. Каже се да је упит Q садржан у упиту Q', у ознаци $Q \subseteq Q'$, уколико над сваком могућом базом D, скуп резултата који враћа упит Q представља подскуп резултата које враћа упит Q', тј. важи $\{\forall D | Q^D \subseteq Q'^D\}$.

Дефиниција 3.2: Еквивалентним се сматрају они упити Q и Q' који су међусобно садржани, односно $Q \equiv Q'$ уколико важи да је $\{\forall D | Q^D = Q'^D\}$.

Провера једнакости резултата упита над конкретном релационом базом података може се спровести примењујући алгоритам са четири провере:

- Поређење броја колона – ово је први корак при поређењу. Уколико се број колона у резултату упита Q разликује од броја колона у резултату упита Q', даље провере се не извршавају и враћа порука о грешци да има премало/превише колона.
- Поређење типова колона – други корак покушава да упари све типове колона из резултата упита Q са типовима колона из резултату упита Q'. Уколико се не може наћи упаривање за сваку од колона, поређење се прекида и враћа се порука о грешци да нису присутне све колоне односно да упити немају идентичну енарност, односно структуру торки.
- Поређење броја редова – слично као и поређење броја колона, пореди се укупан број редова. Уколико резултати које посматраних упита не садрже идентичан број редова, поређење се прекида и враћа се порука о грешци да резултати упита немају исту кардиналност, односно не садрже неке од очекиваних торки.
- Поређење садржаја редова – ако су сви претходни критеријуми задовољени, преостало је да се упореде садржаји појединачних редова. Уколико не могу да се упаре сви редови из резултата упита Q са редовима из резултата упита Q', враћа се порука о грешци да неки од очекиваних редова нису присутни.

Провера еквиваленције упита у општем облику се према наведеним проверама своди на проблем одређивања репрезентативног попуњавања, односно репрезентативне инстанце релационе базе података.

5. Систем за интерактивну проверу сличности

Постоји више начина како би предложени поступци провере сличности концептуалних и логичких модела релационих база података могли да буду проверени и њихова исправност потврђена. Један од начина био би формална верификација, који је у овом случају не прихватљив, пре свега због тога што је већи број проблема којима се овај рад бави класификован као НП-потпун. Други начин јесте провера у виду имплементације конкретног система. С обзиром на то да је област проблема, којим се овај ради бави, јако широка, није могуће развити један систем који би одговарао свим могућим применама. Управо је из тог разлога развијен систем за интерактивну проверу сличности са идејом да буде употребљен у настави, односно као помоћно образовно средство на курсевима који се баве изучавањем релационих база података.

У наставку рада ће најпре бити објашњена мотивација и проблеми код којих би систем био од значаја, а потом ће бити дат преглед и упоредна анализа постојећих система. Потом ће бити описана нови система, предложен у овом раду. Нови систем у својој основи користи поступке за проверу сличности концептуалних и логичких модела описане у претходном делу рада, а њихова имплементација у новом предложеном систему ће бити описана у делу рада који следи. Примери

употребе система биће дати кроз опис конкретних лабораторијских вежби. На крају ће бити дата критичка евалуација проистекла на основу искустава из трогодишње употребе система.

5.1. Мотивација за развој система

Курсеви који се баве изучавањем релационих база података имају значајну улогу у области рачунарских наука и софтверског инжењерства. Један такав курс, Базе података, је први курс посвећен системима за управљање базама података, а који се налази у плану и програму Електротехничког факултета на Универзитету у Београду. Циљеви курса су упознавање се релационим базама података и управљањем трансакцијама, уз посебан нагласак на SQL. Међутим, развијени систем је имао за циљ да покрије ову област на начин који би одговарао што већем броју потенцијалних корисника, а то је могуће само уколико се поштују препоруке светских организација на основу којих су планови и програми курсева углавном и формирани.

Радна група, састављена од удружених представника IEEE и ACM удружења, је идентификовала скуп основних и изборних тема за курсеве који се баве изучавањем база података [63]. У табели 5.1 је дата листа препоручених тема и њихова покривеност курсом Базе података 1 на Електротехничком факултету. Теме које нису покривене овим курсом, обрађују се у друга три курса: напредни курс из база података (Базе података 2), Софтверски алати база података и Информациони системи (Информациони системи 1).

Препоруке IEEE/ACM такође осликавају потребу да студенти треба да поседују не само теоретско већ и практично разумевање обрађиваних тема. Треба имати у

виду и то да су ове препоруке представљају само једну од могућих перспектива на садржај курсева. Друга перспектива долази од оних који учествују у настави на курсевима из база података. Управо су они заступници идеје да су релационе базе података, SQL, концептуално моделовање, нормализација и релациона алгебра најчешће изучаване теме у области база података [64, 65]. Управо из тог разлога постоји потреба не само за скупом лабораторијских вежби, које би покриле те најзаступљеније теме, већ и за системом који би био подршка спровођену вежби.

Развијени систем за интерактивну проверу сличности концептуалних и логичких модела релационих база података, назван ADVICE (Automated Database Verification with Interactive Counter Example), представља интегрисан систем који обезбеђује подршку скупу лабораторијских вежби. Основна функционалност система ADVICE јесте подршка аутоматизованој провери исправности како би се помогло студентима приликом самосталног рада и провери знања, односно предавачима приликом оцењивања [66]. Међутим, у суштини ADVICE омогућава студентима интерактиван рад са базама података на конкретним примерима, а у исто време омогућава и предавачима да прате напредак студената. Све функционалности система су доступне и преко Интернета, односно могу послужити у различитим врстама учења на даљину. У табели 5.1 је такође приказана и колона која означава теме које су покривене системом. Као што се може видети ADVICE покрива 78% основних тема, 30% изборних тема, и тиме све најчешће изучаване теме. Могућности које нуди ADVICE биће описане касније у раду, кроз опис скупа лабораторијских вежби које су развијене да науче студенте како да праве концептуалне моделе и како да их преводе у релационе логичке моделе, како да раде са дефиниционим (DDL) и манипулативним (DML)

деловима SQL, како да раде са формалним упитним језицима и како да користе алгоритме за нормализацију база података. Међутим, пре детаља који се односе на ADVICE у наставку раде ће бити дат преглед постојећих система у овој области.

Табела 5.1 Скуп тема препоручених од стране IEEE/ACM за курсеве посвећене базама података

Модули	Теме	БП курс*	ADVICE
Системи база података [основни]	Компоненте система база података	+	
	Системи за управљање базама података	+	+
	Архитектуре база података	+	+
	Упитни језици база података	+	+
Моделирање података [основни]	Моделирање података	+	+
	Основни концепти (релације, интегритет)	+	+
	Концептуални модели	+	+
	Објектно оријентисани модели		
	Релациони модел података	+	+
Релационе базе података	Концепти шема и релација	+	+
	Референцијални интегритет	+	+
	Релациона алгебра и релациони рачун	+	+
Упитни језици база података	Структурирани упитни језик (SQL)	+	+
	Стратегије обраде упита	+	+
	Упит примерима (QBE)		
	Уграђени упитни језици		
	Објектни упитни језици		
Дизајн релационих база података	Дизајн база података	+	+
	Функцијске зависности	+	+
	Нормалне форме	+	+
	Вишевредносне зависности	+	
Управљање трансакцијама	Трансакције	+	
	Опоравак од квара	+	
	Обезбеђивање серијализованости	+	
Дистрибуиране базе података	Дистрибуирано чување података		
	Обрада дистрибуираних упита		
	Модел дистрибуираних трансакција		
	Хомогена и хетерогена решења		
	Клијент-сервер приступ		
Физички дизајн база података	Захтеви за чување података		
	Карактеристике медијума за чување података		
	Записи и типови записа		
	Структура датотека		
	Б-стабла		
	Компресија података		
	Ефикасност и оптимизација база података		
Покривеност основних тема:		**89%**	**78%**
Покривеност додатних тема:		**44%**	**30%**

***БП курс означава курс из База података на Електротехничком факултету**
Знак (+) значи да је тема покривена БП курсом, односно системом ADVICE

5.2.Преглед постојећих система

Изради система ADVICE претходио је преглед и компаративна анализа постојећих система који се употребљавају као помоћно средство на курсевима из база података. Преглед је обухватио јавно доступне системе, а за потребе њихове анализе успостављен је скуп критеријума. Критеријуми су подељени у две групе, оне који се односе на покривеност тема и оне који се односе на подршку образовном процесу. Критеријум покривености тема говори да ли одређени систем обухвата концептуално моделовање, рад са DDL и DML деловима SQL, формалне упитне језике и нормализацију релационих база података. Критеријум подршке образовном процесу говори да ли одређени систем поседује функционалности за проверу исправности у случају питања са отвореним одговором, функционалност за праћење напретка и постигнућа студената, као и функционалности које би обезбедиле подршку у случајевима временске и/или физичке удаљености између студената и предавача.

Преглед анализираних система и њихових карактеристика у складу са предложеним критеријумима, дат је у табели 5.2. Анализирани системи се значајно разликују у смислу покривања тема као и у нивоу детаља на који то чине. На пример, SQL Trainer [70] и Gradiance SQL [71] покривају само аспект SQL DML, док SQLCourse [72] покрива не само SQL DML већ и SQL DDL аспект. Међутим, прва два споменута поседују проверу исправности и обезбеђују студентима подршку у току рада на основу резултата и исправности њихових одговора.

Један систем који покрива више тема од било ког другог система је ADbC [76, 77]. Он обухвата поред осталих и теме које се односе на управљање трансакцијама,

безбедност у базама података, али му недостају теме које везане за формалне упитне језике. Међутим, главни недостатак система ADbC јесте што он ставња акценат на материјале са курса и њихову анимирану презентацију, док је интеракција са корисником предефинисана и не имплементира проверу исправности, изузев у виду статичних квизова, односно питања са понуђеним одговорима.

Табела 5.2 Преглед и карактеристике система који се користе на курсевима из база података

Систем	Концептуални дизајн	SQL DDL	SQL DML	Формални упитни језици	Нормализација	Провера исправности	Учење на дальину	Праћење напретка
SQL-Tutor [67]	Не	Не	Да	Не	Не	Да	Да	Да
NORMIT [68]	Не	Не	Не	Не	Да	Да	Да	Не
KERMIT [69]	Да	Не	Не	Не	Не	Да	Да	Не
SQL Trainer [70]	Не	Не	Да	Не	Не	Да	Да	Да
Gradiance SQL [71]	Не	Не	Да	Не	Не	Да	Да	Да
SQLCourse [72]	Не	Да	Да	Не	Не	Не	Да	Не
WinRDBI [73]	Не	Не	Да	Да	Не	Не	Не	Не
Web-based Normalization Tool [74]	Не	Не	Не	Не	Да	Не	Да	Не
LDBN [75]	Не	Не	Не	Не	Да	Да	Да	Не
ADbC [76], [77]	Да	Да	Да	Не	Да	Не	Да	Не

Међу анализираним системима, SQL Tutor [67], NORMIT [68] и KERMIT [69] се разликују по томе што заједно чине међусобно комплементаран скуп алата обједињен у виду DB-suite пакета [78]. Иако овај пакет алата задовољава већину постављених критеријума, теме које се односе на SQL DDL и формалне упитне језике нису обухваћене. Други потенцијални недостатак овог пакета јесте што прихвата искључиво SQL упите са фиксном структуром.

Скоро сви системи имају подршку за учење на даљину, изузев WinRDBI [73], који је развијен као Java апликација, па га је самим тим лако прилагодити и за такве примене. Са друге стране, WinRDBI је једини систем, међу анализираним, који има потпуно подршку за формалне упитне језике. Слично томе, алати који такође покривају само једну тему, прецизније тему нормализације релационих база података јесу Web-based Normalization Tool [74] и LDBN [75].

Резултат компаративне анализе је да ниједан од евалуираних система не задовољава у потпуности успостављене критеријуме покривања тема и подршке образовном процесу. Управо та чињеница даје објашњење зашто је развијен ADVICE система. Дакле, то је покушај да се задовоље сви постављени критеријуми, а уз то и потврде поступци провере сличности описани у овом раду. Битна функционалност система ADVICE је провера исправности која омогућава активно учење студената засновано на интерактивним контра примерима. Ова функционалност користи поступке провере сличности описане у овом раду, како би се упоредило решење које је дао студент са званичним решењем које је дао предавач. Након провере сличности, уколико је студентово решење погрешно, систем генерише поруку о првој разлици коју је уочио. Та порука је управо контра пример због чега студентово решење није добро. На тај начин, кроз више

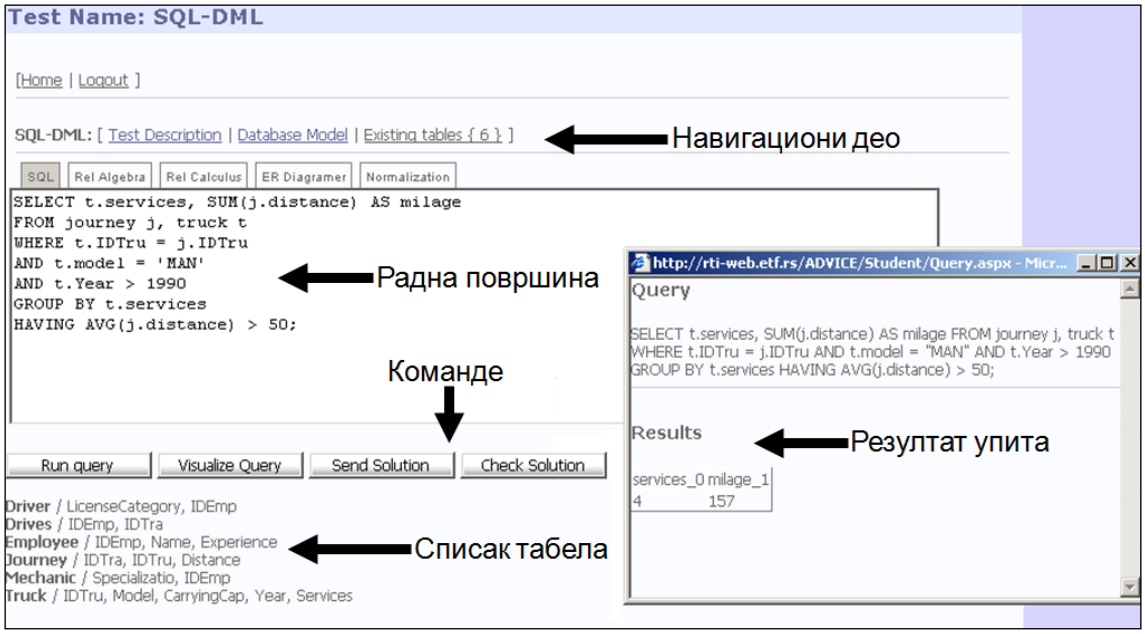
итерација, студент активно учествује у откривању решења, што је у складу са начинима доказаним у пракси [79]. Наравно, у општем случају, иако може постојати више тачних одговора, довољно је да предавач дефинише само једно од њих. У случају SQL, међутим, предавач треба да обезбеди, поред једног тачног решења, скрипт за иницијализацију базе података, као и скриптове за тестирање како би се осигурало да само тачна решења дају исти резултат као и решење које је предавач предвидео. Такође, још једна интересантна функционалност систем ADVICE система је визуелизација упита, која може бити употребљена како би се студентима појаснила не-процедурална семантика SQL упита. Детаљи имплементације система, укључујући и визуализацију, биће дати у наставку рада.

5.3.Архитектура предложеног система

Систем ADVICE се састоји од скупа алата од којих је сваки намењен одређеној теми из области база података. Основна идеја је да се систем користи као подршка приликом спровођења лабораторијских вежби, тако да је свака од тема покривена одговарајућим практичним радом и конкретним примерима из праксе. Алати у оквиру система пружају подршку различитим типовима вежби, конкретно за концептуално моделовање, SQL, релациону алгебру, релациони рачун и нормализацију. Сви алати су интегрисани, а за две основне улоге у систему, студентску и професорску, систем обезбеђује два графичка интерфејса за све подржане функционалности. Интерфејс за студенте, који омогућава студентима да користе систем и приступају конкретним примерима база података, и интерфејс за предаваче, који им омогућава да додају нове вежбе и прате напредак студената.

Сваки студент користи систем да би приступио својој инстанци базе података, која може бити празна или попуњена подацима, што зависи од конкретне вежбе.

Интерфејс за студенте се састоји од три целине и типичан изглед је приказан на слици 5.1. Прва целина је навигациони део који садржи везе ка детаљима тренутно активне вежбе, друга целина је радна површина са свим алатима и скупом командних дугмића, док трећа целина садржи листу табела које су доступне у студентовој инстанци базе података. У зависности од тренутно одабраног алата, могу постојати и додатни прозори, на пример, прилико задавања SQL упита појављује се додатни прозор са резултатом које је упит вратио.



Слика 5.1 Типичан изглед екрана система ADVICE из перспективе студената

Навигациони део садржи назив тренутно активне вежбе, везу ка страни са описом вежбе (Test Description), везу ка страни на којој се налази концептуални модел (Database model) за конкретну вежбу и везу ка тренутним табелама (Existing Tables) на основу које се укључује, односно искључује приказ листе доступних табела. Навигациони део има исту структуру и значење без обзира на тип вежбе.

Радна површина се састоји од посебног дела, тј. посебног таба за сваки од алата који је доступан студенту. Студент може да користи више од једног алата у истом тренутку. На слици 5.1 се може видети да постоје табови за сваки од алата, односно за SQL, релациону алгебру, релациони рачун, концептуално моделовање и нормализацију. Изглед радне површине зависи од тренутно одабраног алата. На слици 5.1 је приказан алат за SQL који се састоји од текстуалног поља који омогућава студенту да пише произвољне SQL исказе. У случају да студент зада упит, приказује се и додатни прозор који садржи резултате извршавања тог упита. Изглед радне површине за остале алате биће описан касније у раду у делу који описује примере употребе система уз опис конкретних лабораторијских вежби.

Скуп командних дугмића омогућава студенту интеракцију са системом. У случају алата за SQL, ови дугмићу су Покрени упит (Run Query), Визуализуј упит (Visualize Query), Пошаљи решење (Send Solution) и Провери решење (Check Solution). Дугме Покрени упит омогућава студенту да покрене произвољан SQL исказ, не само упит, над инстанцом базе података креиране управо за њега. У случају упита, као резултат покретања упита, отвара се нови прозор са резултатом (Query). Дугме Визуализуј упит омогућава студенту да види графичку репрезентацију упита. Поступак визуелизације се односи искључиво на SQL упите, и детаљи тог поступка су описани касније у раду. Дугме Пошаљи одговор омогућава студенту да сачува у систему одговор који је дао за тренутну вежбу. Одговор који је студент послао биће на располагању предавачу за ручно или аутоматизовано оцењивање. Дугме Провери одговор омогућава студенту да консултује систем за аутоматску проверу исправности одговора. У случају нетачног одговора, систем ће студенту вратити поруку која садржи контра пример

који би требао да наведе студента на исправно решење. Дугме Провери одговор је доступно само уколико предавач то жели, односно може бити искључено у одређеним вежбама.

Систем омогућава предавачу да прави нове вежбе, прави групе студената и да дефинише њихова права приступа над вежбама. Све набројане акције се спроводе кроз интерфејс за предавача и при томе свака акција има засебан екран.

Да би направио нову вежбу, предавач треба да дефинише њен назив, дефинише проблем који се решава, одабере тип вежбе, напише скрипт за иницијализацију базе података, направи решење и по потреби напише додатне скриптове за тестирање исправности решења. Доступни типови вежби су SQL DDL, SQL DML, релациона алгебра, релациони рачун, концептуално моделовање и нормализација.

Предавач пише скрипт за иницијализацију базе података који ће систем извршити аутоматски за сваког студента на почетку вежбе. Скрипт треба да инстанцу базе података сваког од студената попуни почетним садржајем. Почетни садржај је обавезан само у случају вежби које се односе на SQL DML, релациону алгебру и релациони рачун, док је за остале типове вежби почетни садржај опциони.

Предавач треба да направи решење уколико жели да омогући аутоматизовану проверу исправности. У току прављења решења предавач може по потреби да користи исте алате који ће касније бити доступни студентима током вежбе. У случају SQL DDL вежби, осим тачног решења, предавач мора да дефинише и додатне скрипте за тестирање који се користе приликом провере исправности. Скрипте за иницијализацију и тестирање морају да буду написани тако да у потпуности покривају све грешке које предавач жели да детектује. Алгоритми за

аутоматизовану проверу исправности засновани су на поступцима за проверу сличности концептуалних и логичких модела релационих база података који су раније описани у раду. Провера исправности се у суштини ослања на провери сличности између модела, којег је задао предавач као тачног, са једне стране, и модела, којег је направио студент као предлог решења, са друге стране. Примери резултата рада ових алгоритама као и примери иницијализационих скрипти, решења и додатних скрипти за тестирање биће приказани касније у раду код описа конкретних лабораторијских вежби.

Уколико провера исправности није потребна, предавач не мора да припрема вежбу. Систем се може користити чак и без припремљене вежбе. У том случају, систем се понаша као скуп алата који омогућава студенту да направи произвољан концептуални модел, да тај модел преведе и да на основу тога аутоматски направи шему релационе базе података. Над тако добијеном шемом базе података, студент може да извршава произвољне SQL исказе, исказе релационе алгебре и релационог рачуна.

Предавач може да направи групе студената, додавањем једног по једног или увођењем комплетне листе студената од једном. Сваки студент добија јединствено корисничко име којим ће се пријављивати на систем, и основу кога ће бити праћен његов напредак. Систем омогућава предавачу да помоћу механизма за контролу приступа дефинише да само одређене групе студената могу да приступе одређене вежбе. Овај механизам се може искористити у случајевима великог броја студената, или у случајевима када се истовремено организују лабораторијске вежбе за већи број различитих курсева посвећених базама података. Изгледи екрана за неке од описаних опција дати су у прилогу А.

5.4. Детаљи имплементације предложеног система

Систем ADVICE има модуларну структуру и састоји се од модула за SQL, формалне упитне језике, нормализацију, концептуално моделовање, визуализацију упита, администрацију студената и управљање вежбама. Систем за управљање базама података је независтан од модула, и може бити развијен од различитих произвођача. Захваљујући модуларној структури, систем је јако флексибилан и може бити проширен додатним модулима. Нови модули могу бити развијени у различитим технологијама и међусобно повезани одговарајућим интерфејсима. Имплементација модула је стандардна, па ће због тога у наставку рада бити описани само најбитнији имплементациони детаљи.

Алгоритми описани у наставку односе се на проверу исправности модела, проверу исправности упита и визуализацију упита. Циљ алгоритма за проверу исправности модела је да омогући предавачу да направи само решење и скриптове за тестирање, док ће систем аутоматски проверити исправност студентовог решења и на основу тога генерисати сугестије које ће у итеративном поступку наводити студента ка тачном решењу. Систем генерише сугестије тако што проверава сличност између студентовог одговора и решења које је задао предавач. Циљ визуелизације упита је да помогне студенту да разуме SQL као не-процедурални језик. Визуелизација се састоји у увођењу графичких елемената за најбитније делове упита, чиме се поједностављује целокупан сазнајни (когнитивни) процес.

5.4.1. Провера исправности модела

Резултат алгоритма за провере исправности модела су сугестије које говоре о томе који ентитети и/или атрибути недостају у одговору студента. Сугестије настају

када алгоритам открије прву разлику између одговора студента и решења коју не може да упари. Алгоритам се ослања на поступке провере сличности модела описане у претходном делу рада, и може се класификовати као приступ заснован на шеми који комбинује приступ на нивоу елемената и приступ на нивоу структуре [80].

```
         match(answer, solution){
Korak 1.     schema1 = toSchema(answer);
             schema2 = toSchema(solution);

Korak 2.     reduce(schema1);
             reduce(schema2);

Korak 3.     for each (newSchema1, newSchema2) in eliminate(schema1, schema2){

Korak 4.            for each renamedSchema1 in rename(newSchema1, newSchema2){

Korak 5.                   if(compare(renamedSchema1, newSchema2)){
                                  return success;
                           }
                    }
             }
             return failure;
          }
```

Слика 5.2 Алгоритам за проверу исправности концептуалних модела

Алгоритам користећи поступак провере сличности анализира информацију о броју и називима ентитета у моделима, затим броју, типовима и називима атрибута унутар ентитета, али такође и ограничењима у виду примарних, страних и кандидат кључева. Алгоритам је у својој основи сличан приступу описаном у [81] с обзиром да се заснива на алгоритму Поплаве сличности, али се разликује у томе што ADVICE обавља додатне трансформације. Трансформације се огледају на употреби методама формалне анализе концепата као и у конкретним операцијама редукције шема и елиминацији елемената који не могу да буду упарени.

Алгоритам за проверу исправности приказан на слици 5.2 састоји се од пет корака, и као улаз прихвата концептуалне моделе. Концептуални модели представљају студентов одговор и решење које је дао предавач. Први корак је трансформација концептуалних модела у интерну репрезентацију засновану на формалној анализи концепата и чува информацију о целокупним шемама, тј. табелама, атрибутима, ограничењима али и уређености међу њима. Други корак је редукција шема, која обухвата компакцију веза са кардиналношћу 0..1, егзистенцијалних односа са кардиналношћу 0..1. Трећи корак је елиминација, која се ослања на одређивање разлике међу посматраним моделима и уклањању табела и атрибута који се не могу упарити. Табеле, односно атрибути, у једној шеми су они који немају своју одговарајућу табелу, односно атрибут, у другој шеми. Две табеле су упарене, уколико имају исти број упарених атрибута са одговарајућим типовима. Упарени атрибути имају иста ограничења и компатибилне типове. Табеле и атрибуте које није могуће упарити биће елиминисани из обеју шема. Корак елиминације као резултат враћа скуп парова шема, с обзиром да може бити више од једног могућег упарења између елемента анализираних шема. За сваки пар шема спроводе се четврти и пети корак алгоритма. Четврти корак је преименовање шема, табела, атрибута и ограничења у једној шеми, како би се проверила сличност са одговарајућим елементима у другој шеми. Преименовање се у суштини обавља над шемом која представља одговор студента, како би се утврдило шта у њој недостаје. Информација о мапирању обављеном приликом сваког од преименовања се чува како би касније могла да се генерише сугестија која треба да укаже студенту шта је погрешно у његовом решењу. Корак преименовања као резултат враћа скуп шема са могућим преименовањима табела и атрибута. За

свако од могућих преименовања спроводи се корак пети корак. Пети корак обавља упоређивање шема једноставним упоређивањем стрингова.

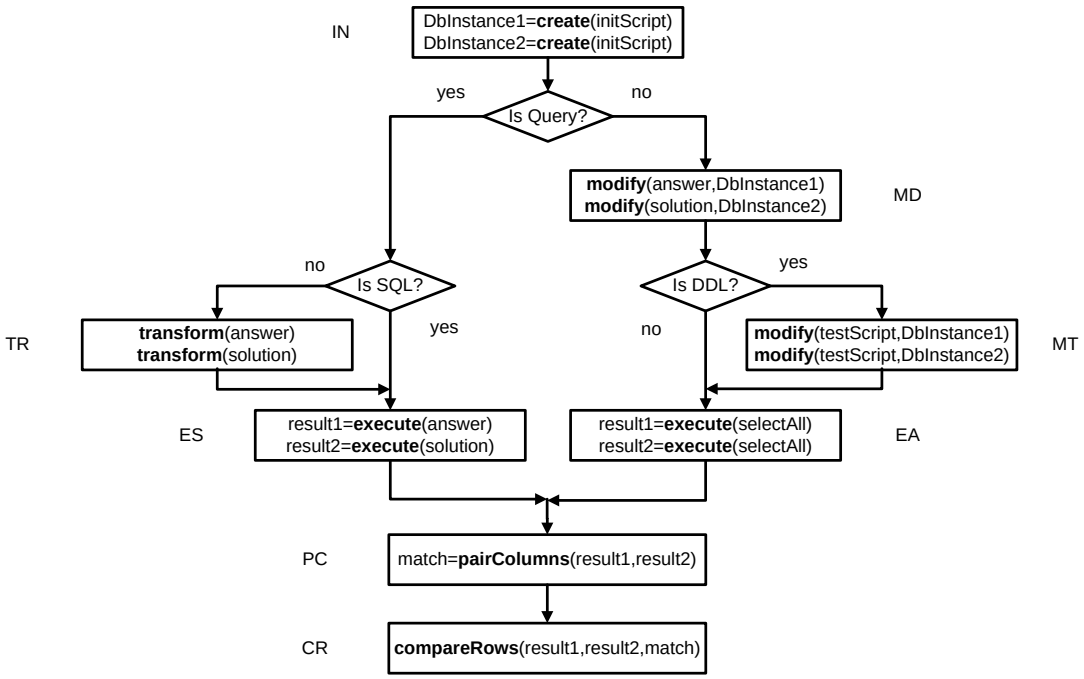
Имајући у виду да је провера сличности модела проблем који је НП-потпун [82], наивна имплементација описаног алгоритма може да има слабе перформансе. Из тог разлога имплементација корака елиминације и преименовања користе одређена правила. Прво правило дефинише број атрибута које је дозвољено елиминисати из шема. Сврха овог правила је да се избегну случајеви који би могли да одузму много времена за проверу, чиме се смањује комплексност алгоритма. Друго правило захтева да корак преименовања не користи само број и тип атрибута већ и њихова имена и ограничења. Из практичних разлога, дозвољени типови ограничења су примарни, кандидат и страни кључеви. Ограничење је то што алгоритам, у најгорем случају, обавља исцрпну претрагу. Међутим, претпоставка је, а и препорука, да предавач треба да даје проблеме, односно решења, мање комплексности. Систем је успешно тестиран са примерима са до десет табела и са укупно двадесет атрибута.

5.4.2. Провера исправности исказа

Резултат провере исправности исказа су сугестије које говоре да у резултату добијеном на основу студентовог одговора, одређени ред недостаје или да је сувишан. Тумачење недостајућег реда, односно сувишног реда пре свега зависи од тога да ли се посматра исправност исказа који представља упит, или неког исказа којим се декларише нова табела. Алгоритам за проверу исправности исказа је дат на слици 5.3. Алгоритам разматра еквивалентност два исказа на основу тога да ли остављају базу података у истом стању, за разлику од алгоритама који се заснивају на техникама трансформација упита како би се утврдила логичка

еквиваленција [83], без обзира на садржај базе података. Описани алгоритам је сличан алгоритмима за проверу исправности употребљеним у алатима SQL Trainer [70] и Gradiance SQL [71], али се разликује по томе што је општији и као такав применљив и при провери исправности SQL DDL исказа, као и исказа задатих формалним упитним језицима.

Поступак провере исправности зависи од типа исказа. Алгоритам разликује четири типа исказа, SQL DML, SQL DDL, релациону алгебру и релациони рачун.



Слика 5.3 Алгоритам провере исправности исказа

У случају SQL Select, поступак провере исправности упоређује два скупа података и њихове мета податке. У овом контексту термин скуп података означава резултат извршавања упита, који по својој природи може садржати дупликате, али ће у наставку бити коришћен због еквиваленције са термином који се користи у

енглеском језику (data set). За потребе провере исправности, први скуп података се добија као резултат извршавања (први део корака ES) упита који представља одговор студента, над првом копијом референтне инстанце базе података креиране на основу скрипта за иницијализацију базе података (први део корака IN). Други скуп података је резултат је резултат извршавања решења које је предавач дао (други део корака ES) над другом копијом референте инстанце базе података(други део корака IN). Овако добијена два скупа података се упоређују као што је описани раније у раду у оквиру предлога поступака за проверу сличности манипулативног дела логичких модела. Описани поступак се спроводи у два корака. Први корак је упаривање колона користећи мета податке о скупу података (корак PC), док је други корак упоређивање самих података уз игнорисање уређеност података (корак CR). Поступак провере исправности у случају исказа релационе алгебре и релационог рачуна захтева додатни корак (корак TR) који обавља трансформацију датог исказа у еквивалентан SQL Select исказ, по правилима описаним у делу који описује репрезентацију упита као првог корака у предлогу поступака за проверу сличности логичких модела.

У случају SQL DDL исказа, поступак провере исправности се спроводи само за Create Table исказе. Разлика у односу на проверу исправности SQL Select исказа је у томе, што су поред скрипте за иницијализацију базе података, потребне и скрипте за тестирање које садрже додатне SQL тест исказе. Поступак се понавља за сваки тест скрипт понаособ.

Тест скрипте могу садржати комбинацију Insert, Update и Delete исказа који треба да провере важење ограничења. Као и раније, и сада се упоређују два скупа података. Први је резултат извршавања тест SQL исказа (први део корака MT) над

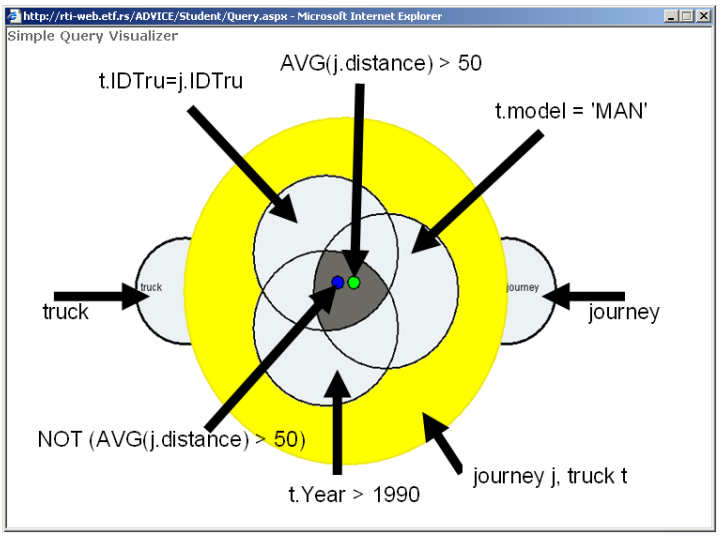
копијом референтне инстанце базе података (први део корака IN) која је модификована користећи Create Table исказ који је студент дао у свом одговору (први део корака MD). Други скуп података је резултат извршавања тест SQL исказа (други део корака MT) над копијом референтне инстанце базе података (други део корака IN) која је модификована користећи Create Table исказ који је дао предавач у оквиру решења (други део корака MD). Два скупа података који ће бити упоређивани се добијају дохватањем свих новододатих редова у новокреираним табелама (корак EA). Остатак поступка провере исправности је еквивалентан претходно описаном случају (кораци PC и CR). У случају SQL Insert, Update, или Delete исказа поступак провере исправности се разликује од управо описаног поступка SQL DDL по томе што додатне тест скрипте са SQL исказима нису потребне (прескаче се корак MT).

Описани алгоритам има ограничења, тј. претпоставке које морају бити испуњене. Односно исправност алгоритма зависи не само од решења које је предавач дао, већ и од скрипте за иницијализацију базе података, а у случају SQL DDL исказа и од додатних тест скрипти. Дато решење мора бити тачно, док скрипт за иницијализацију базе података и додатне тест скрипте морају да буду написане тако да покривају све могуће грешке које предавач жели да открије у одговорима студената. У случају Select исказа, додатно ограничење је да колоне које су резултат агрегатних функција морају да имају називе које је предавач специфицирао у поставци проблема. Слично, у случају SQL DDL исказа, додатно ограничење је да именовање колона и редослед дефинисања колона мора такође да буде онакав како је предавач специфицирао у поставци проблема.

5.4.3. Визуелизација упита

Техника визуелизације упита се састоји у формирању графичких еквивалената за најбитније делове SQL Select исказа. Постоји велики број случајева у којима је визуелизација упита потребна, на пример, у вишедимензионој обради података [84], оптимизацији рада база података [85] и слично. За потребе ADVICE система развијена је техника визуелизације упита која се заснива на скуповима, како би се помогло студентима да лакше разумеју не-процедуралну семантику SQL упита. Један пример упита и одговарајућег дијаграма приказан је на слици 5.4. Визуелизација упита се састоји од два корака, синтаксне анализе и динамичког извршавања.

```
SELECT t.services, SUM(j.distance) AS milage
FROM journey j, truck t
WHERE t.IDTru = j.IDTru
AND t.model = 'MAN'
AND t.Year > 1990
GROUP BY t.services
HAVING AVG(j.distance) > 50;
```



Слика 5.4 Пример визуализације упита са две табеле и агрегатном функцијом

Циљ корака синтаксне анализе је да идентификује најбитније делове упита за које се могу дефинисати једноставна правила визуелизације. Најбитнији елементи упита су управо клаузуле SQL упита. Свака табела, поглед или упит наведен у From клаузули се представља засебним полукругом на самом ободу дијаграма.

Ови полукрузи се повезују главним кругом који представља Декартов производ свих елемента у From клаузули. За сваки услов специфициран у Where клаузули креира се засебан унутрашњи круг (круг унутар главног круга). У случају комплексних логичких услова формираних употребом AND и OR операторима за повезивање услова, сваки од тих услова се визуализује засебно, уз приказ свих пресека међу њима. У случају када упит садржи Having клаузулу, два мала круга различите боје се приказују у самом центру дијаграма тј. у пресеку свих услова. Један од та два мала круга, представља редове, тј. групе, које задовољавају услов Having клаузуле, док други представља оне који не задовољавају тај услов.

Циљ корака динамичког извршавања је да омогући студенту да погледа податке који одговарају одређеним деловима визуализованог упита. У случају одабира неког од полукругова, којима је визуализована From клаузула, садржај основних табела или погледа се приказује у засебном прозору. Слично, у случају одабира неког од делова логичког услова из Where клаузуле, приказују се редови који задовољавају одабрани део услова. Такође, одабиром неког од два централна круга приказују се подаци који одговарају Having клаузули.

Визуелизација упита може да прикаже само главни упит, односно угнеждени подупити не могу бити визуализовани засебно. Додатно ограничење је да није подржана ортогоналност SQL упита у смислу да не може бити визуализован упит који у Select листи садржи други Select исказ.

5.5. Примери употребе предложеног система

У циљу илустрације рада развијеног система ADVICE у наставку ће бити описани примери његове употребе. Као примери употребе одабрани су делови конкретних

лабораторијских вежби на којима је систем био коришћен у оквиру наставе предмета Базе података 1 на Електротехничком факултету у Београду. Најпре ће бити дат увид у организацију лабораторијских вежби, а потом и описане конкретне лабораторијске вежбе.

5.5.1. Организација лабораторијских вежби

Свака лабораторијска сесија се састоји од четири дела: припрема за лабораторију, прелиминарни тест процене спремности студента за рад у лабораторији, лабораторијска вежба и извештај о урађеној вежби. У току припреме за лабораторијску вежбу студент може приступити ADVICE систему од своје куће, путем Интернета. Процена припремљености се спроводи кроз прелиминарни тест, који се састоји од задатака сличних онима који се раде на лабораторијској вежби, али је за време трајања теста помоћ система ADVICE, односно опција за проверу исправности, недоступна студенту. Свака лабораторијска вежба је састављена од неколико задатака које студент треба да уради. Помоћ система ADVICE, заснована на итеративној провери исправности која користи претходно описане алгоритме за проверу исправности модела и исказа, је доступна студенту у току трајања лабораторијске вежбе. Након завршетка лабораторијске вежбе, извештај се аутоматски формира на основу одговора које је студент давао у току свог рада.

Предавач посматра рад студента све време трајања лабораторијске вежбе, не само на крају, и на тај начин стиче утисак о томе како студенти користе систем. Предавач може да види последњи одговор који је студент сачувао у систему, али и број итерација које је студент направио са системом. На овај начин предавач може да усмери своју пажњу и помогне оним студентима који у изради заостају у односу на остале студенте. Када се лабораторијска вежба заврши, предавач има

извештаје о свим студентима, као и статистику о њиховој интеракцији са системом. Извештај о студенту се састоји од одговора које је дао у току трајања лабораторијске вежбе, као и одговора које је дао у току прелиминарног теста. У току оцењивања студената, предавач може, а не мора, користити систем за проверу исправности.

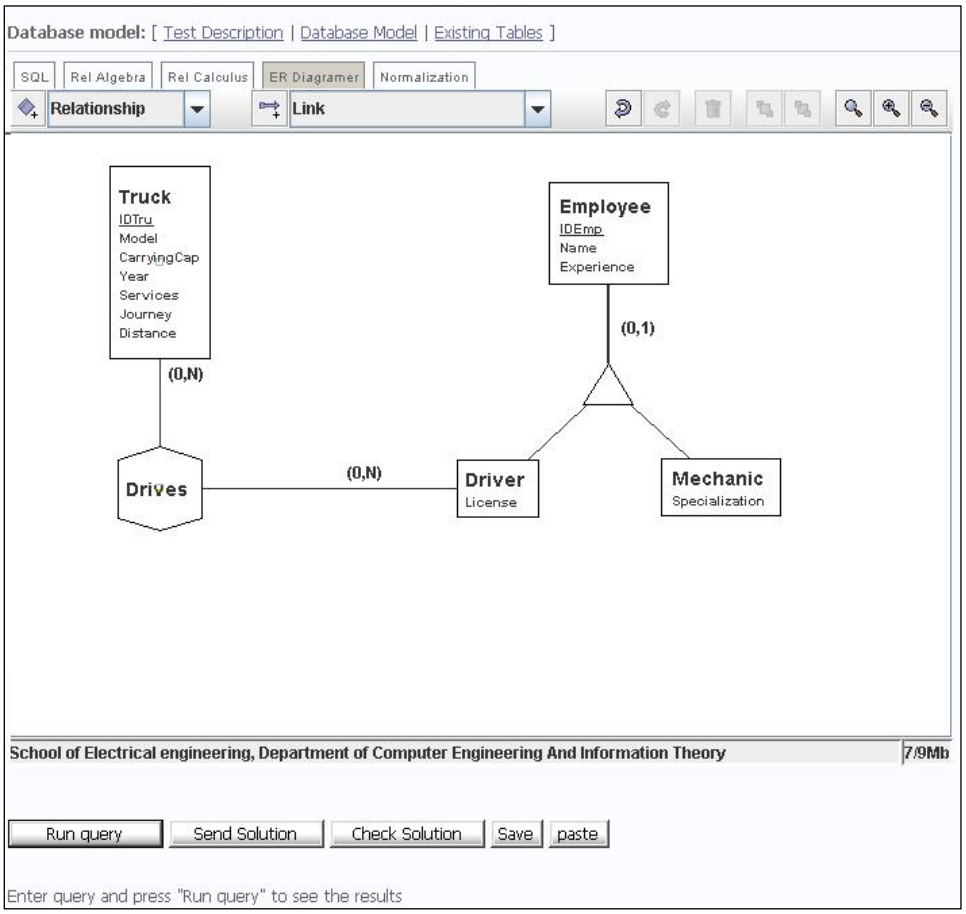
У наставку ће бити описани поједини задаци у оквиру лабораторијских вежби, како би се демонстрирао рад система ADVICE. Лабораторијске вежбе обухватају модел ентитета и односа, SQL DDL, SQL DML, релациону алгебру и нормализацију. Ради прегледности, опис сваке од вежби има следећу структуру: циљ вежбе, опис проблема, досег проблема, пример провере исправности студентовог решења.

5.5.2. Вежба посвећена моделу ентитета и односа

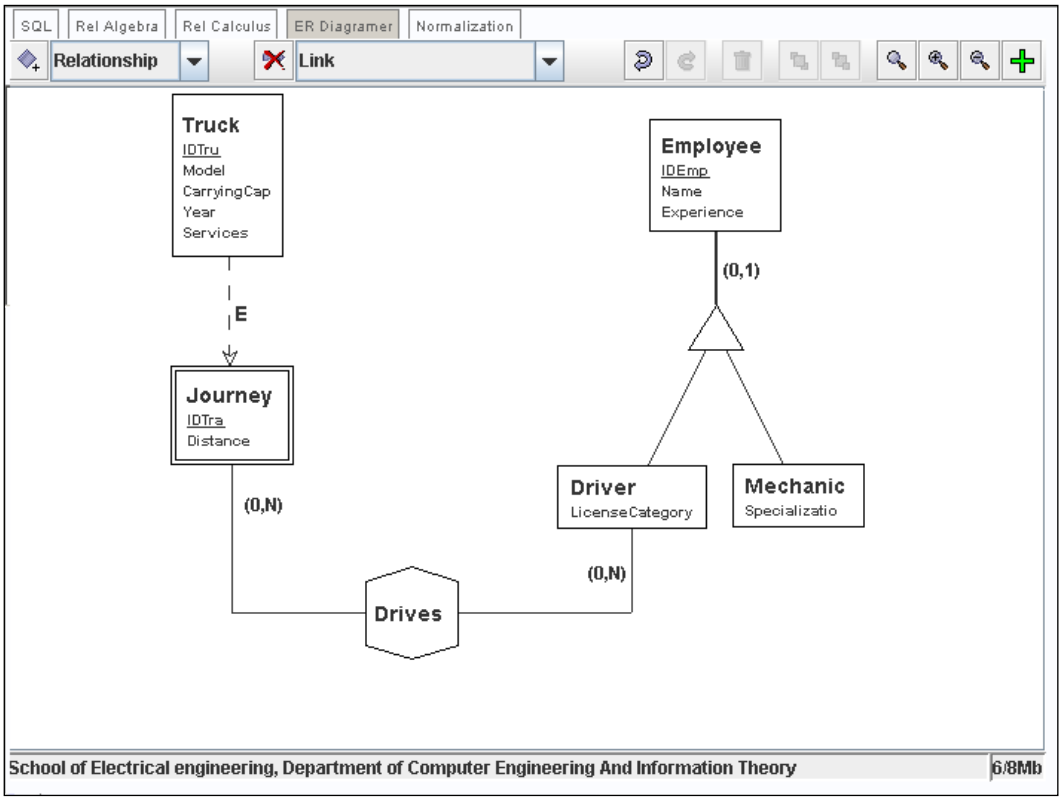
Циљ вежбе: Овај тип вежбе обухвата концептуално моделовање, логичко моделовање, концепте примарног кључа, кандидат кључа, страног кључа и типова података.

Опис проблема: Посматрани систем је компанија за превоз која поседује известан број камиона. За сваки камион прати се информација о типу, капацитету, години производње и број сервиса. Има, презиме и радно искуство је потребно чувати за сваког запосленог. За неке запослене треба чувати и додатне информације, у случају возача то је возачка дозвола, а у случају механичара информација о његовој специјализацији. Компанија мора да чува информацију и о свим путовањима које је организовала. За свако путовање бележи се пређено растојање као и информације о томе који је камион коришћен и који возачи су учествовали.

Досег проблема: У овој вежби, студент користи ADVICE, односно његов алат под називом ER Diagramer. Овај алат помаже студентима да разумеју основне графичке елементе ER нотације у складу са [86]. У овом конкретном случају, студенту се демонстрира концепт ентитета, као и директни и индиректни односи међу њима. Концепт егзистенцијалног (не-идентификујућег) односа се користи између јаких и слабих ентитета, камион-путовање (truck-journey). У моделу постоји и директни однос наслеђивања, који се користи у проширеној ER нотацији, радник-возач-механичар (employee-driver-mechanic). Концепт индиректног односа се користи како би се успоставио однос "вози" ("drives") између ентитета путовање и возач.



Слика 5.5 Пример одговора студента за вежбу модела ентитета и односа



Слика 5.6 Тачно решење за вежбу модела ентитета и односа

Пример провере исправности студентовог решења: Слике 5.5 и 5.6 показују студентов одговор и тачно решење, респективно. Намера студента је била да за потребе чувања података о путовању употреби атрибут уместо ентитета. Након покретања опције за проверу исправности, студент добија поруку “Недостаје ентитет: Путовање” (“An entity is missing: Journey”). Док год студент не упари све ентитете, систем генерише поруке да одређени ентитети недостају или да су сувишни. Тек када студент упари све ентитете, систем генерише поруке које се односе на атрибуте.

5.5.3. Вежба посвећена SQL – DDL

Циљ вежбе: Овај тип лабораторијских вежби покрива све аспекте Create Table исказа. Вежба обухвата декларацију примарног кључа, кандидат кључа, као и страног кључа, типова података, ограничења на нивоу атрибута и нивоу табеле.

Опис проблема: Написати SQL DDL исказ који ће у претходно описани модел компаније за превоз додати нову табелу Сервис са атрибутима шифра запосленог, шифра камиона, број сервиса (“Service(IDEmp, IDTru, NumOfServices)”). Уколико је механичар сервисирао камион, у тој табели се чува информација о томе колико пута је то урадио. Сваки ред табеле Сервис треба да реферише постојећег механичара и постојећи камион. У случају брисања и/или мењања шифре механичара или шифре камиона, одговарајући редови у табели Сервис треба такође да буду обрисани, односно промењени.

```
CREATE TABLE Service
(IDEmp INT REFERENCES Mechanic(IDEmp)
           ON DELETE CASCADE
           ON UPDATE CASCADE,
IDTru INT REFERENCES Truck(IDTru)
           ON DELETE CASCADE
           ON UPDATE CASCADE,
NumOfServices INT);
```

Слика 5.7 Пример одговора студента за вежбу SQL-DDL

```
CREATE TABLE Service
(IDEmp INT NOT NULL REFERENCES Mechanic(IDEmp)
           ON DELETE CASCADE
           ON UPDATE CASCADE,
IDTru INT NOT NULL REFERENCES Truck(IDTru)
           ON DELETE CASCADE
           ON UPDATE CASCADE,
NumOfServices INT NOT NULL CHECK(NumOfServices>0) DEFAULT 1,
PRIMARY KEY(IDEmp, IDTru));
```

Слика 5.8 Тачно решење за вежбу SQL-DDL

Truck

IDTru	Model	CarryingCap	Year	Services
0	'Mercedes'	6	1994	1
1	'Volvo'	8	2005	0
2	'MAN'	12	2004	4

Mechanic

IDEmp	Specialization
0	'Brakes'
1	'Clutch'
2	'Suspension'

Слика 5.9 Почетно стање табела вежбе SQL-DDL постављених скриптом за иницијализацију

```
INSERT INTO Service VALUES (0,0,1);     -- test normalnog ponašanja

INSERT INTO Service VALUES (NULL,0,1);  -- test not NULL ograničenja
                                        -- atributa IDEmp

INSERT INTO Service VALUES (0,NULL,1);  -- test not NULL ograničenja
                                        -- atributa IDTru

INSERT INTO Service VALUES (0,1,NULL);  -- test not NULL ograničenja
                                        -- atributa NumOfServices

INSERT INTO Service VALUES (4,1,5);     -- test loše vrednosti IDEmp
                                        --(ne postoji IDEmp = 4)

INSERT INTO Service VALUES (0,7,3);     -- test loše vrednosti IDTru
                                        --(ne postoji IDTru = 7)

INSERT INTO Service VALUES (2,2,-1);    -- test loše vrednosti NumOfServices
                                        -- (NumOfServices mora biti > 0)

DELETE FROM Mechanic WHERE IDEmp=0;     -- test referencijalnog integriteta
                                        -- (Mechanic, Service)

DELETE FROM Truck WHERE IDTru=0;        -- test referencijalnog integriteta
                                        -- (Truck, Service)
```

Слика 5.10 Пример тест скрипте за вежбу SQL-DDL

Досег проблема: У овој вежби, студент користи SQL алат у оквиру ADVICE система. Алат омогућава студенту да за потребе вежбе пише SQL DDL исказе и да на тај начин научи концепте идентификационог интегритета (дефинисање примарног кључа), референцијалног интегритета (дефиниција страног кључа), као и једноставна ограничења на нивоу атрибута (јер подупити нису подржани) за потребе исправности података (атрибут број сервисирања је увек већи од нуле).

Пример провере исправности студентовог решења: Слике 5.7 и 5.8 садрже одговор студента и решење, респективно. Студент је заборавио да страни кључ не имплицира ограничење о забрањеном NULL (“not null”), а исто важи и са број сервиса који не сме бити изостављен. Шта више, студент није обратио пажњу на чињеницу да број сервиса треба да буде позитивна вредност, и да та информација не треба да буде чувана више пута за сваки пар механичар-камион. Како би све ове грешке биле откривене, предавач мора да обезбеди одговарајуће почетно стање базе података као и тест скрипте. Почетно стање базе је приказано на слици 5.9 , док је на слици 5.10 приказан тест скрипт. Намена првог Insert исказа у тест скрипту је да провери регуларно понашање. Следећа три исказа проверавају ограничења недостајућих вредности (“not null”), три исказа која следе проверавају не регуларне вредности за сваки од атрибута у табели, док последња два исказа проверавају да ли је успостављен динамички референцијални интегритет. Након покретања опције за проверу исправности, студент добија поруку “Овај ред не треба да буде прихваћен у табели: [, 0 , 1]” (“This row should not be accepted by your table: [, 0 , 1]”). Све док студент не напише тачан одговор, систем генерише поруке које указују на откривене грешке.

5.5.4. Вежба посвећена SQL – DML

Циљ вежбе: Овај тип лабораторијске вежбе обухвата Select, Insert, Update и Delete исказе. Представљени проблем се односи на Select исказ. У овој вежби, студент као помоћ такође може да користи и модул који му омогућава визуализацију упита.

Опис проблема: За претходно описани проблем компаније за превоз, написати упит који враћа растојање и тип камиона за свако путовање на коме је учествовало најмање два возача.

Досег проблема: У овој вежби, студент користи SQL алат у оквиру ADVICE система. Алат му омогућава да напише SQL DML исказ за потребе вежбе и да на тај начин научи како да спаја табеле (спољна спајања су подржана), користи агрегатне функције, дефинише корелисане подупите и пише комплексне услове.

Truck

IDTru	Model	CarryingCap	Year	Services
0	'Mercedes'	6	1994	1
1	'Volvo'	8	2005	0
2	'MAN'	12	2004	4

Journey

IDJou	Distance	IDTru
0	157	2
1	78	0
2	189	1

Drives

IDEmp	IDJou
0	0
1	0
1	1
2	0
2	1
2	2

Слика 5.11 Почетно стање табела вежбе SQL-DML постављених скриптом за иницијализацију

Пример провере исправности студентовог решења: Слика 5.11 приказује садржај табела релевантних за овај проблем. На сликама 5.12 и 5.13 приказани су одговор који је студент дао, решење, као и резултати које су ти упити вратили. Студентов одговор није тачан зато што обухвата само путовања на којима је учествовало тачно два возача. Након покретања опције за проверу исправности, студент добија поруку “Имате један ред мање. Недостајући ред: (157,MAN)” (“You have 1 row(s) less. Missing row: (157,MAN)”). Алгоритам за проверу исправности генерише поруке које се односе на редове који недостају, као и редове који су сувишни у

студентовом одговору. У случају да студенту недостају и колоне, он би о томе био обавештен одговарајућом поруком.

```
SELECT J.Distance, T.Model
FROM Journey J, Truck T
WHERE T.IDTru=J.IDTru
AND (SELECT COUNT(*)
     FROM Drives D
     WHERE D.IDJou=J.IDJou)=2;
```

```
Одговор система на дати упит:
78, 'Mercedes'
```

Слика 5.12 Пример одговора студента за вежбу SQL-DML

```
SELECT J.Distance, T.Model
FROM Journey J, Truck T
WHERE T.IDTru=J.IDTru
AND (SELECT COUNT(*)
     FROM Drives D
     WHERE D.IDJou=J.IDJou)>=2;
```

```
Одговор система на дати упит:
78, 'Mercedes'
157, 'MAN'
```

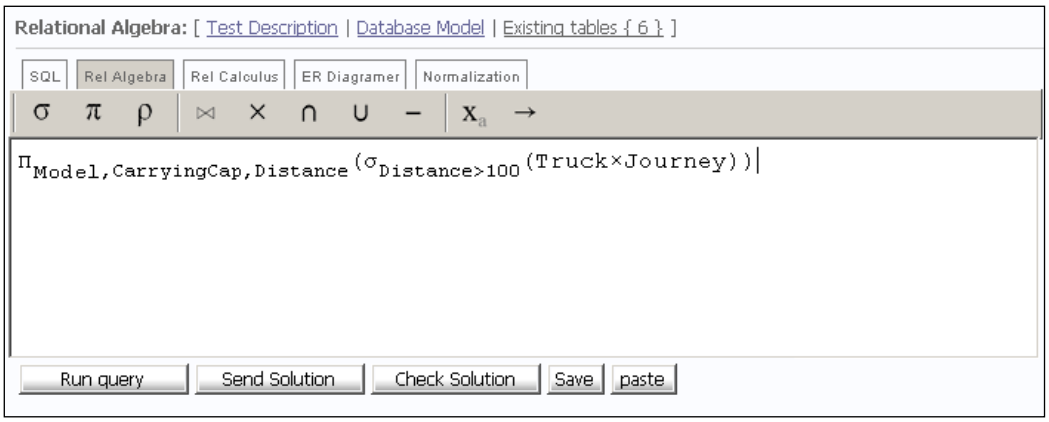
Слика 5.13 Тачно решење за вежбу SQL-DML

5.5.5. Вежба посвећена релационој алгебри

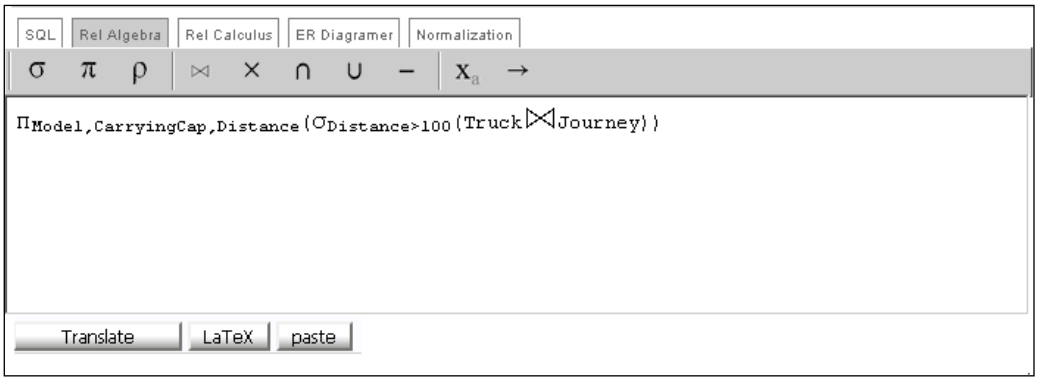
Циљ вежбе: Овај тип лабораторијских вежби треба да обради исказе релационе алгебре. У поређењу са DML, циљ ових вежби је да нагласи процедурални аспект долажења до резултата. Кроз већи број мањих задатака, вежба треба да покрије употребу основних и долажење до изведених операција релационе алгебре.

Опис проблема: За претходно описани проблем компаније за превоз, написати исказ релационе алгебре, који за свако путовање дуже од 100 километара, као резултат даје пређено растојање, модел и носивост камиона који је учествовао на путовању.

Досег проблема:У овој вежби, студент користи алат за Релациони алгебру у оквиру ADVICE система. Алат помаже студенту да разуме употребу посебних операција рестрикције, пројекције, Декартовог производа, природног спајања, као и основне скуповне операције попут уније и разлике, у складу са [87].



Слика 5.14 Пример одговора студента за вежбу релациона алгебра



Слика 5.15 Тачан одговор за вежбу релациона алгебра

Пример провере исправности студентовог решења: Садржаји табела релевантних за дати проблем су приказани на слици 5.11, док су на сликама 5.14 и 5.15 су приказани студентов одговор и решење. Сваки исказ релационе алгебре, ADVICE

најпре трансформише у одговарајући SQL упит, без да то приказује студенту, а онда га извршава. Исказима релационе алгебре са слика 5.14 и 5.15, одговарају SQL упити приказани на сликама 5.16 и 5.17. Студентов одговор није тачан зато што користи Декартов производ уместо природног спајања. Након покретања опције за проверу исправности студент добија поруку “Имате 4 реда више. Сувишни редови су: ('Mercedes',6,157), ('Mercedes',6,189), ('Volvo',8,157), ('MAN',12,189)” (“You have 4 row(s) more. Superfluous row: ('Mercedes',6,157), ('Mercedes',6,189), ('Volvo',8,157), ('MAN',12,189)”). Алгоритам за проверу исправности враћа поруку о недостајућим и сувишним редовима откривеним у одговору студента. У случају да студенту недостају и колоне, он би о томе био обавештен одговарајућом поруком.

```
SELECT DISTINCT T.Model, T.CarryingCap, J.Distance
FROM Truck T, Journey J
WHERE J.Distance>100;
```

Слика 5.16 Упит настао превођењем исказа релационе алгебре са слике 5.14

```
SELECT DISTINCT T.Model, T.CarryingCap, J.Distance
FROM Truck T, Journey J
WHERE T.IDTru=J.IDTru
AND J.Distance>100;
```

Слика 5.17 Упит настао превођењем исказа релационе алгебре са слике 5.15

5.5.6. Вежба посвећена нормализацији

Циљ вежбе: Овај тип лабораторијских вежби обрађује теорију нормализације. За разлику од вежби посвећених моделу ентитета и односа, циљ је да се објасни поступак дизајна базе података по принципу одоздо навише.

Опис проблема: Спровести декомпозицију релационе шеме D(IDJou, Distance, IDTru, Model, AxleRatio) до 3NF, без губитака при спајању и уз очување

функцијских зависности, уколико над шемом важе функцијске зависности F={IDJou→Distance,IDTru; IDTru→Model; Model→AxleRatio}.

Досег проблема: У овој вежби, студент користи алат за Нормализацију у оквиру система ADVICE. Овај алат помаже студенту да разуме нормализацију кроз примере који обухватају поступке одређивања кандидат кључева, каноничног покривача, декомпозиције до различитих нормалних форми (2NF, 3NF, и BCNF), као и поступак провере очувања функцијских зависности.

```
nf=3;
r=D(IDJou, Distance, IDTru, Model, AxleRatio);
F={IDJou->Distance,IDTru IDTru->Model Model->AxleRatio };
d=D1(IDJou,Distance,IDTru)|D2(IDTru,Model,AxleRatio);
```

Слика 5.18 Пример одговора студента за вежбу нормализација

```
nf=3;
r=D(IDJou, Distance, IDTru, Model, AxleRatio);
F={IDJou->Distance,IDTru IDTru->Model Model->AxleRatio};
```

Слика 5.19 Тачан одговор за вежбу нормализација

Пример провере исправности студентовог решења: Одговор студента је приказан на слици 5.18, а решење на слици 5.19. Алат за нормализацију не захтева да у решењу буде наведене декомпозиција, већ само поставка проблема, јер је алат у стању да сам провери исправност декомпозиције коју је студент дао. У датом примеру, студентов одговор није тачан зато што занемарује чињеницу функцијска зависност Model→AxleRatio нарушава услове за 3NF. Након покретања опције за проверу исправности студент добија поруку “Погрешан одговор. Постоји грешка у следећој релационој шеми D2(IDTru, Model, AxleRatio)“ (“Wrong answer. Errors in the following relational schemas: D2(IDTru, Model, AxleRatio)“) . Односно,

алгоритам за проверу исправности враћа поруку о релационим шемама које не задовољавају задату нормалну форму.

5.6. Евалуација предложеног система

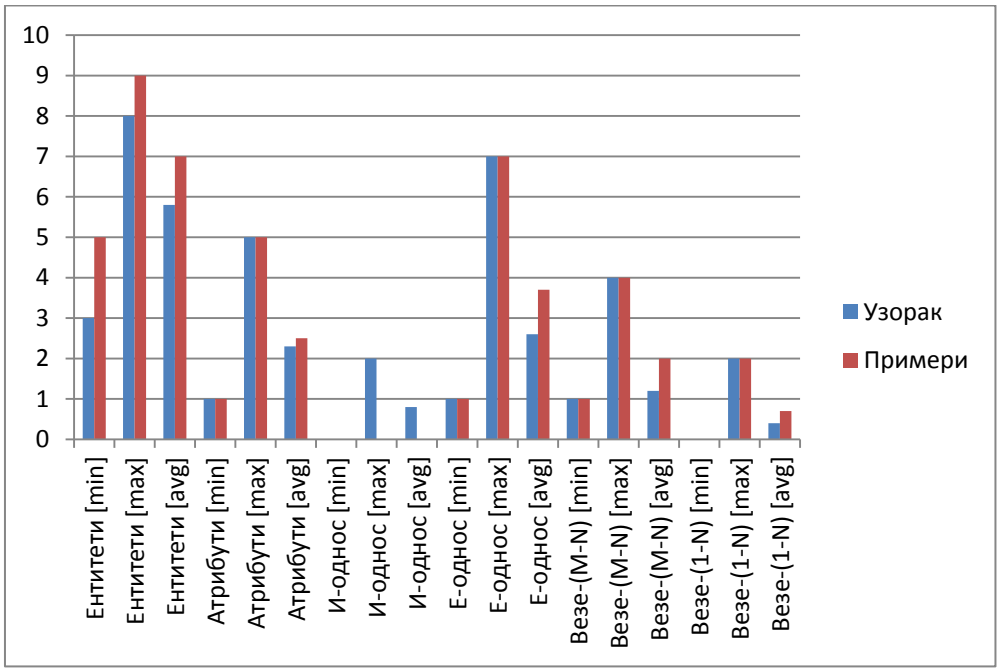
У овом делу рада биће дата евалуација предложених поступака провере сличности као и евалуација практичних искустава у коришћењу целокупног ADVICE система. Евалуација предложених поступака обухвата анализу провере сличности концептуалних модела на основу спроведених експеримената. Анализа поступака провере сличности логичких модела није приказана посебно, због тога што је природа предложених поступака таква да су они потпуно егзактни и за проверу њихове исправности коришћен је скуп примера чији је један део дат у прилогу Ц. Међутим, у наставку рада ће бити дата, евалуације система као целине, и то кроз анализу резултата приликом практичне примене система. Управо посматрањем система као целине, обухваћени су не само ефекти примене провере сличности концептуалних модела, већ и поступака провере сличности логичких модела.

5.6.1. Евалуација предложених поступака провере сличности

Предложени поступци провере сличности су искоришћени при имплементацији система за интерактивну проверу сличности који има за циљ да служи као подршка на курсевима из база података и да помаже студентима да савладају разлике између теорије и праксе. Евалуација предложених поступака провере сличности обављена је спровођењем експеримената и статистичком анализом добијених резултата.

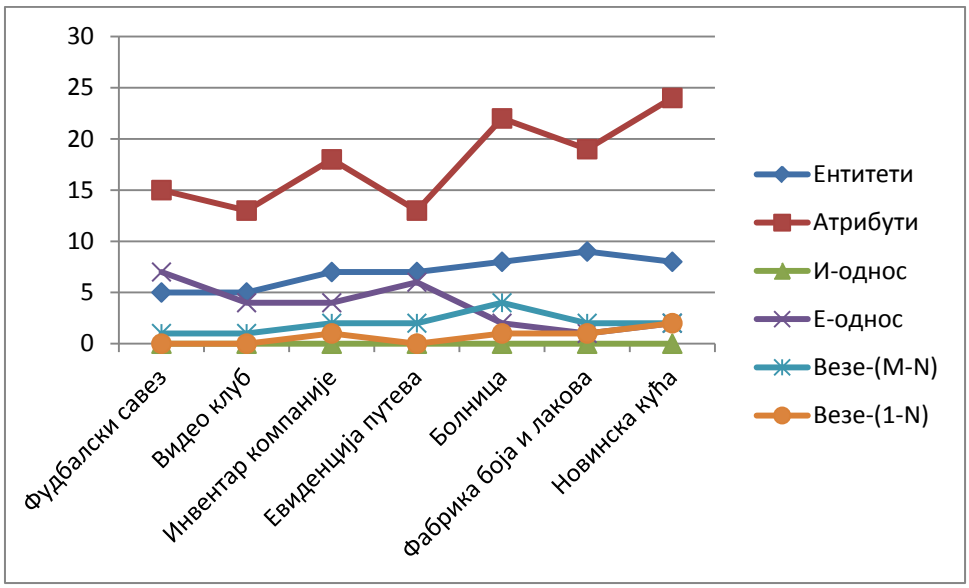
Како би биле процењене карактеристичне ситуације у којима ће система, а самим тим и предложени поступци провере сличности, бити коришћени, спроведена је

анализа одређеног броја курсева из база података на универзитетима широм света, а за које је било могуће наћи примере испитних задатака. Универзитети су одабрани међу првих педесет универзитета са листе Академског рангирања светских универзитета [88], и при том су посматрани њихови курсеви из база података у периоду од 2004 до 2011. Како би била процењена сложеност посматраних примера, у направљеном узорку сакупљена је статистика о броју појединих елемента модела ентитета и односа. Тако су, на пример, посматрани број ентитета и број атрибута у тим ентитетима, али су такође посматрани и број идентификационих и егзистенцијалних односа који се појављују у тим моделима, као и број веза, и то веза више-на-више и веза један-на-више. Статистика анализираног узорка дата је слици 5.20. На слици означено као узорак, јесу примери прикупљени са анализираних универзитета, док је примерима назван скуп примера чији су детаљи дати у прилогу Б.



Слика 5.20 Карактеристике посматраних примера концептуалних модела

Примера из прилога Б су коришћени као основа за анализу која следи у наставку рада. Са слике се може видети да су примери из прилога одабрани тако да по својим средњим вредностима одговарају направљеном узорку, али да по максималним вредностима буду изнад узорка, како би се што боље проверило понашање предложених поступака у граничним случајевима. Једино одступање постоји по питању идентификационих односа којих у предложеним примерима уопште нема. Како би се то решило, посматрани примери су за потребе анализе били модификовани тако да су све везе у примерима биле модификоване тако да буду представљене користећи идентификационе и егзистенцијалне зависности. Карактеристике примера из прилога Б дату су на слици 5.21.



Слика 5.21 Карактеристике примера за евалуацију поступака провере сличности

Да би била спроведена евалуација могућности предложених поступака да на адекватан начин детектују грешке студената, и да на њих одговоре, направљена је листа најчешћих грешака. Листа најчешћих грешака на курсу из база података као

и кратак опис тих грешака дати су у табели 5.3. Вероватноће су израчунате на основу грешака пронађених у радовима студената, како на лабораторијским грешкама, тако и на испитима. Вероватноћа за неку грешку у табели, означава очекиван просечан број студената који праве ту грешку док раде задатке у којима се тај тип грешке може појавити.

Табела 5.3 Најчешће грешке студената при изради концептуалног модела

Тип	Опис грешке	Вероватноћа	Бр. конц. 1	Бр. конц. 2
Тип 1	Ентитет и Е-однос уместо атрибута	0,31	2	8
Тип 2	Ентитет и веза уместо атрибута	0,15	2	11
Тип 3	Инвертована условљеност	0,27	8	8
Тип 4	Веза уместо условљености	0,22	8	11
Тип 5	Тернарна веза уместо агрегације и везе	0,19	15	18
Тип 6	Однос између ентитета специјализације	0,12	12	17

С обзиром да се приликом провере сличности концептуалних модела, користи репрезентација заснована формалној анализи и концепата, у табели 5.3 су такође дате и колоне које представљају број формалних концепата који се појављују у тим грешкама. При том, треба напоменути да је наведени број формалних за случај када сваки од ентитета има само по два атрибута од којих је један одабран за примарни кључ, док су све везе без атрибута.

Способност предложених поступака да детектују ове грешке анализирана је на примерима из прилога Б. Вероватноће детектовања најчешћих грешака дата је у табели 5.4. Мерења су спроведена тако што је сваки од примера више пута мењан

на основу сваке од карактеристичних грешака, наравно промене су сваки пут биле међусобно различите.

Табела 5.4 Вероватноћа детектовања одређеног типа грешке у анализираним примерима

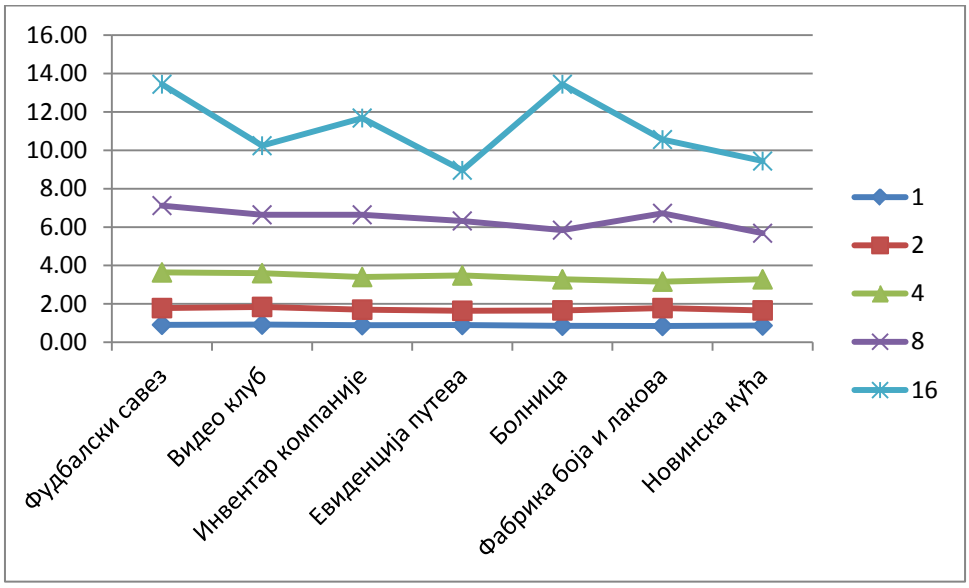
	Фудбалски савез	**Видео клуб**	**Инвентар компаније**	**Евиденција путева**	**Болница**	**Фабрика боја и лакова**	**Новинска кућа**
Тип 1	0,92	0,85	0,90	0,84	0,81	0,88	0,71
Тип 2	0,88	0,73	0,65	0,81	0,74	0,79	0,62
Тип 3	0,81	0,85	0,62	0,87	0,78	0,81	0,82
Тип 4	0,75	0,70	0,72	0,85	0,80	0,83	0,80
Тип 5	0,00	0,00	0,70	0,00	0,69	0,00	0,00
Тип 6	0,00	0,00	0,85	0,32	0,71	0,80	0,83

За вредности приказане у табели 5.4 треба напоменути да поља у којима је уписана вредност вероватноће 0,0 означава да тај тип грешке није било могуће направити у посматраном примеру. То је случај са типовима грешака који захтевају да у примеру постоји специјализација, односно агрегација или тернарна веза. Вероватноћа грешака детектованих је скоро у свим случајевима јако висока, поготово ако се узме у обзир да грешке могу захтевати број формалних концепата у распону од два до осамнаест, као што је приказано у табели 5.3. Са тим у вези у табели 5.5 дат је број формалних концепата који су настали на основу датих примера.

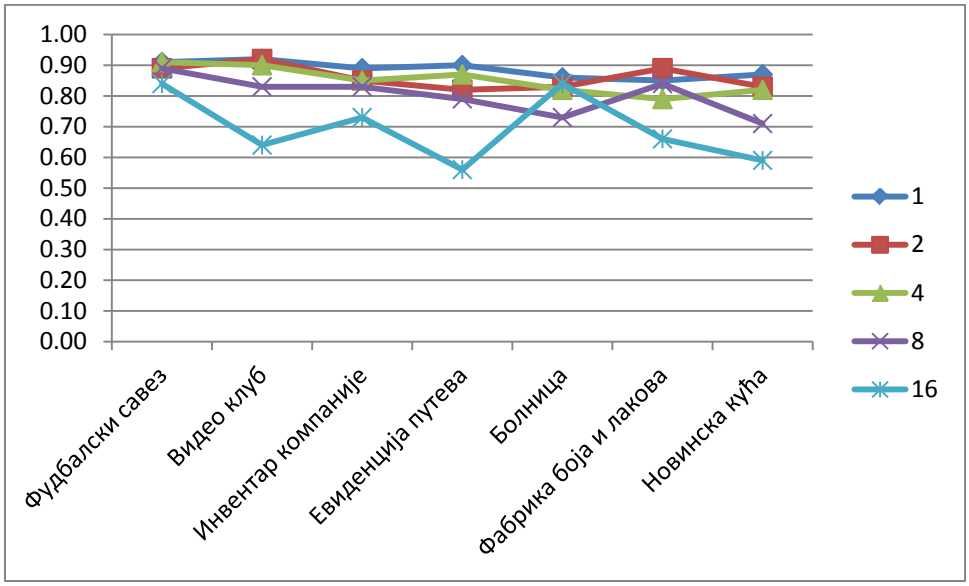
Табела 5.5 Број формалних концепата у анализираним примерима

	Фудбалски савез	Видео клуб	Инвентар компаније	Евиденција путева	Болница	Фабрика боја и лакова	Новинска кућа
Број формалних концепата	31	24	42	38	44	41	44

Вредности приказане у табели 5.4 односе се на случајеве када је присутна само једна грешка, како би се проценила способност предложеног поступака провере сличности концептуалних модела да детектује одређени тип грешке. Резултати мерења спроведених у случајевима када у примерима има већи број различитих грешака приказани су на сликама 5.22 и 5.23.



Слика 5.22 Способност детектовања грешака у присуству већег броја грешака



Слика 5.23 Вероватноћа детектовања грешака у присуству већег броја грешака

Слика 5.22 показује колико грешака је било успешно детектовано у анализираним примерима, када је у њима било присутно више различитих грешака, конкретно 1, 2, 4, 8 и 16 грешака. На слици 5.23 су приказани ти исти резултати, али у процентима. Примећује се да је вероватноћа детектовања грешака стабилна у случајевима када је број грешака мали. У случајевима када је број грешака велики, вероватноћа детекције почиње знатно да варира. Узрок томе стоји у чињеници да су посматрани модели имали у просеку по седам ентитета, па је увођење шеснаест грешака довело до сувише велике промене у моделима. Међутим, оно што треба приметити јесте да је упркос великом броју грешака, већи број њих ипак био успешно детектован.

С обзиром да се предложени поступак провере сличности ослања на ново уведену репрезентацију заснованој на формалним концептима, спроведена је анализа доприноса оваквог вида репрезентације. Анализирана је способност предложеног

поступка провере сличности да детектује одређени тип грешке, са и без присуства репрезентације у виду формалних концепата. У случају када није коришћена репрезентација у виду формалних концепата, коришћена је репрезентација предложена у систему Рондо [31]. У табели 5.6 су дати резултати спроведених експеримената у случају са и без употребе формалних концепата.

Табела 5.6 Резултати експеримената са, и без, употребе предложене репрезентације

Тип грешке	Са употребом предложене репрезентације	Без употребе предложене репрезентације
Тип 1	0,92	0,95
Тип 2	0,88	0,84
Тип 3	0,81	0,74
Тип 4	0,75	0,54
Тип 5	0,74	0,58
Тип 6	0,85	0,76

Постављена је нулта хипотеза (H_0) са тврђењем да експериментални резултати добијени при употреби предложене репрезентације нису значајно различити у односу на експерименталне резултате добијене у случајевима када предложена репрезентација није употребљавана. Овој хипотези се придружује алтернативна хипотеза (H_a) и тврди супротно, односно да између експерименталних резултата добијених при употреби, и без употребе, предложене репрезентације постоји значајна разлика.

За потребе доказа употребљен је тест значајности, конкретно, t-тест са упареним узорцима, с обзиром да су идентични примери коришћени у обе врсте

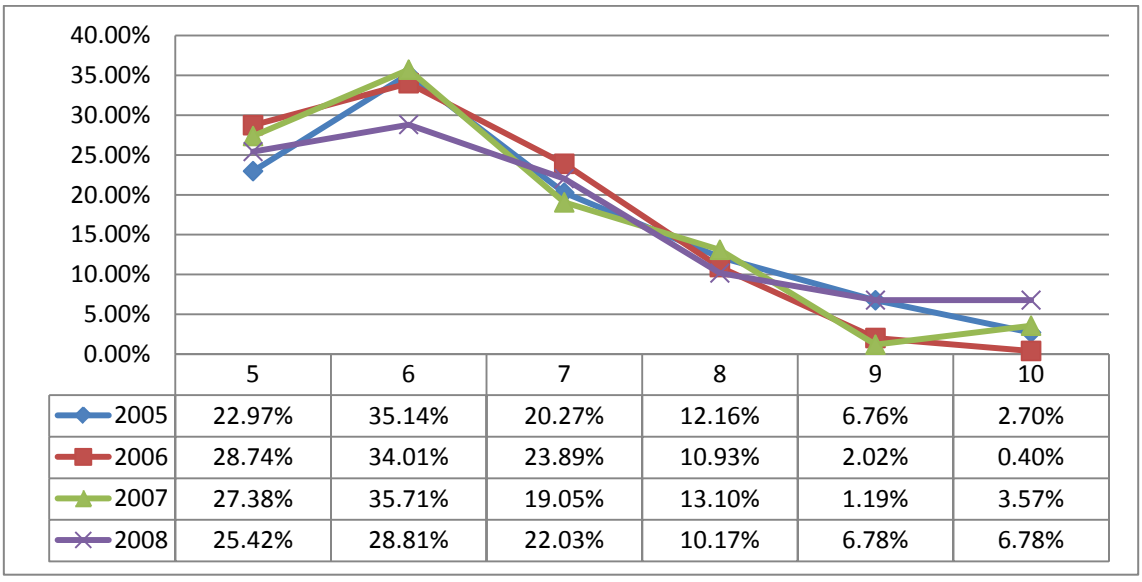
експеримената. Такође, примењен је двосмерни тест, с обзиром да није могуће унапред одредити смер разлике, односно могући исход је да предложена репрезентација даје лошије резултате, а не боље како је то планирано.

Резултати теста показују да је добијена t-вредност по апсолутној вредности већа од апсолутне вредности граничне вредности, те се стога нулта хипотеза одбацује. Такође, средња вредност резултата експеримената у којима је употребљавана предложена репрезентација (0,825) је већа од средње вредности резултата експеримента у којима предложена репрезентација није употребљавана (0,735). Стога се може закључити да увођење репрезентације засноване на формалним концептима доприноси бољем препознавању најчешћих грешака.

5.6.2. Евалуација система за интерактивну проверу сличности

У овом делу рада биће дата евалуација практичних искустава у коришћењу ADVICE система. Систем је коришћен у оквиру курса Базе података на Електротехничком факултету Универзитета у Београду. Крајња оцена на курсу Базе података зависи од испита и практичног рада. Практични рад се састоји од домаћег задатка (10%) и обавезних лабораторијских вежби (20%). Остатак оцене (70%) се добија на писменом испиту. У циљу процене ADVICE система, спроведене је квалитативна и квантитативна евалуација. Ова евалуација разматра период посматрања од 2004 до 2008. У школској години 2004/05 предложени систем није постојао; у 2005/06 прва верзија ADVICE система је развијена; у 2006/07 нова верзија система са аутоматском провером исправности је уведена у употребу; док је у 2007/08 уведена нова верзија са алатом за нормализацију.

На крају сваке школске године, Електротехнички факултет на Универзитету у Београду спроводи квалитативну евалуацију. Ова евалуација је у форми анонимне анкете и обавезна је за све студенте. Од студената се тражи да оцене квалитет курсева које су похађали у току године за коју се евалуација ради. Сваки курс се оцењује оценама одличан, врло добар, добар, задовољавајући и незадовољавајући. Просечна оцена за курс Базе података се променила са оцене добар на оцену врло добар након увођења система ADVICE у наставу. Поред оцењивања, студенти током анкете имају прилику да дају своје мишљење и коментаре на курс. Иако анкета не обухвата експлицитно питање о лабораторијским вежбама, коментари студената су били јако корисни за сваку нову верзију система ADVICE, с обзиром да су коментари описивали лабораторијске вежбе као веома корисне за разумевање материје и праћење наставе.

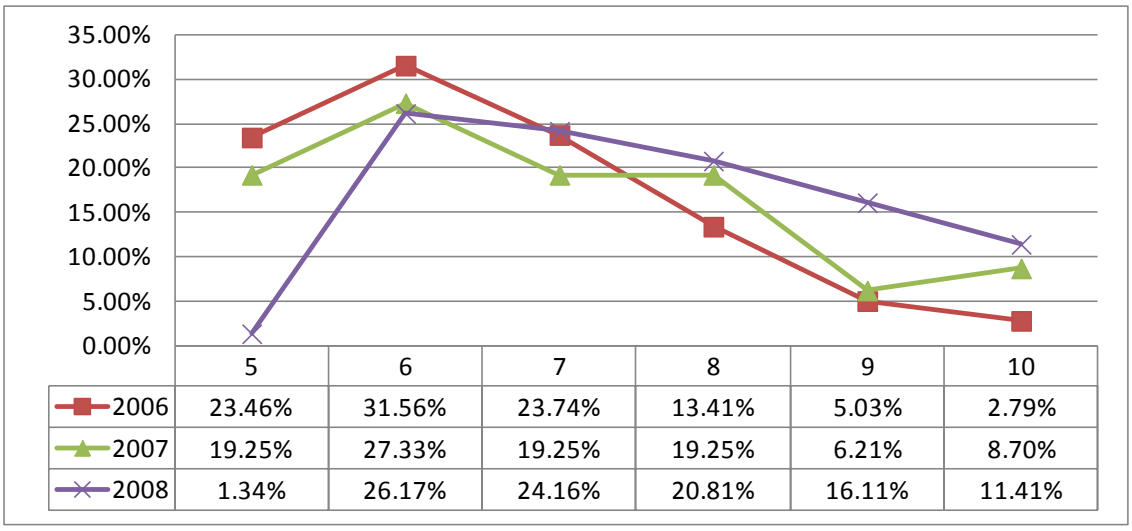


	5	6	7	8	9	10
2005	22.97%	35.14%	20.27%	12.16%	6.76%	2.70%
2006	28.74%	34.01%	23.89%	10.93%	2.02%	0.40%
2007	27.38%	35.71%	19.05%	13.10%	1.19%	3.57%
2008	25.42%	28.81%	22.03%	10.17%	6.78%	6.78%

Слика 5.24 Успех контролне група у периоду посматрања 2005-2008

За потребе квантитативне евалуације у периоду посматрања употребљене су статистичке методе. Разматране су две групе студената, експериментална и контролна. Експериментална група се састојала од студената који су користили предложени систем, док су у контролној групи била они који га нису користили. Групе нису биле формиране селекцијом, већ су се сами студенти по својој вољи опредељивали за то да ли желе да користе систем или не. Слике 5.24 и 5.25 показују успех студената на делу курса који је био покривен системом ADVICE. Успех студента се оцењује вредностима од 5 до 10, и при том је оцена 5 за оне студенте постигну успех мањи од 50% од максималног броја поена.

Слика 5.24 показује успех контролне групе. С обзиром да ADVICE систем није био у употреби у 2005., и имајући у виду да је корелација између графова већа од 0,97, може се закључити да знање студената није варирало током периода посматрања.



	5	6	7	8	9	10
2006	23.46%	31.56%	23.74%	13.41%	5.03%	2.79%
2007	19.25%	27.33%	19.25%	19.25%	6.21%	8.70%
2008	1.34%	26.17%	24.16%	20.81%	16.11%	11.41%

Слика 5.25 Успех експерименталне групе у периоду посматрања 2006-2008

Слика 5.25 показује успех који су постигли студенти који су користили ADVICE систем. Упоређујући школске године 2005/06, 2006/07 и 2007/08 може се да је сваке године растао број оних студената који су прешли границу од 50% и постизали веће оцене.

За анализу постигнућа студената примењена је двофакторска анализа варијанси, односно ANOVA тест. Први фактор у анализи је година посматрања, са три нивоа (2006, 2007, 2008). Други фактор је посматрана група, са два нивоа (експериментална, контролна). Зависна варијабла је успех студената за посматрану групу у посматраној години.

Главни ефекат фактора посматране године је значајан ($F(2, 721)=5,1286$, $p=0,00614$), што значи да постоји разлика између година. Главни ефекат фактора посматране групе је такође значајан ($F(1, 721)=90,489$, $p=0,0000$), што значи да постоји разлика између експерименталних и контролних група. Интеракција између ова два фактора, фактора година и група, није статистички значајан ($F(2, 721)=0,90318$, $p=0,40574$), што значи да разлика између експерименталних и контролних група није завистан од посматране године. Такође, може се рећи и да је профил постигнућа по години исти, како за експерименталне тако и са контролне групе.

Упоређивање графика на сликама 5.24 и 5.25 показује да је тачка пресека између резултата контролне и експерименталне групе на оцени 7. То значи да је код експерименталне групе проценат студента са већим оценама повећан, док се је проценат студената који нису прешли праг од 50% смањен. Такође, резултати на

финалном испиту су показали да је просечна оцена студената у експерименталној групи за 5.6% већа од просечне оцене студената у контролној групи.

Табела 5.7 Зависност оцене студента од просечног броја интеракција студента са системом

Оцена	**5**	**6**	**7**	**8**	**9**	**10**
Просечан број интеракција	19,83	51,91	65,79	81,60	101,33	160,82

Анализа резултата је такође показала да је постоји веза између постигнућа и интеракције са системом. Односно, успех студента је линерано пропорционалан просечном броју интеракција студента са системом. Та зависност је приказана у табели 5.7. Како би успешно завршио све лабораторијске вежбе студент мора да има најмање 72 интеракције са системом. Табела 5.7 показује да студенти који нису завршили све вежбе и имали само 70 или мање интеракција, су постигли само оцену 7 или мању.

6. Закључак

Курсеви посвећени базама података имају значајну улогу у области рачунарских наука и софтверском инжењерству. Већина курсева обухвата не само предавања и вежбе на табли, већ и практичан рад. Практичан рад је организован у виду лабораторијских вежби које се изводе уз подршку система развијених за те потребе. У овом раду је најпре дефинисан скуп критеријума заснованих на препорукама IEEE/ACM удружења, а након тога, је, на основу тих критеријума, обављена евалуација већег броја јавно доступних система који се користе као подршка настави из база података на универзитетима широм света. Упоредна анализа је показала да не постоји ни један систем који би у потпуности задовољио скуп дефинисаних критеријума. Овај резултат био је мотивација за развој новог система, описаног у овом раду, који је назван ADVICE.

Систем ADVICE , је развијен према успостављеним критеријумима, покрива све најзначајније теме и омогућава интерактивну итеративну проверу исправности рада студената, као и визуализацију упита. За потребе провере исправности, развијени су нови поступци провере сличности концептуалних и логичких модела, који су такође описани у овом раду. Како би били подржани и формални упитни језици, дефинисана су и правила превођења исказа релационе алгебре и

релационог рачуна у еквивалентне SQL упите. ADVICE је коришћен у настави из база података у оквиру лабораторијских вежби, а у овом раду су приказани најбитнији делови тих вежби. Након коришћења спроведене су квалитативна и квантитативна анализе. Резултати ових анализа показали су да је употреба система ADVICE довела до пораста задовољства студената и њихове просечне оцене на курсу. Задовољство студената је порасло са оцене „добар“ на оцену „врло добар“, док је њихова просечна оцена на испиту повећана за 5,6%.

Искустава стечена током коришћења система ADVICE показују да његова модуларна архитектура, поред свих предности које доноси, може бити посматрана и као недостатак у одређеним ситуацијама. Наиме, при развоју система ADVICE, како би се истакла његова флексибилност, сваки од његових модула, односно алата, је развијан на различитој технологији. Резултат тога јесте отежано одржавање, које је нарочито присутно приликом првог покретања система, односно инсталације. Такође, недостаци су уочљиви и по питању графичког корисничког интерфејса. При развоју корисничког интерфејса, основни критеријум је био једноставност употребе, како би период адаптације корисника био што краћи. Развијени интерфејс заиста омогућава да корисници, нарочито студенти, већ при првом контакту са системом разумеју све његове функционалности. Међутим, последица те једноставности, нарочито код алата посвећеног нормализацији, који је готово у целости заснован на тексту, довела до тога да студенти избегавају да користе тај алат. Из угла предавача, недостаци по питању корисничког интерфејса, су углавном у делу у коме се дефинишу скрипте за потребе провере исправности логичких модела.

Правци даљег развоја могу бити усмерени не само на отклањање уочених недостатака, већ и на развој нових функционалности и покретање нових истраживања. Једна од нових функционалности би могла да буде подршка система, да предавачу омогући увид у честе грешке студената. Једноставнија имплементација овога би била могућа тиме што би систем трајно чувао сваку интеракцију коју је студент имао са системом, а не само крајњи одговор студента, чиме би предавач могао да анализира како је студент дошао до решења. Међутим, велики број студената и велики број интеракција довео би до потребе система за великом количином смештајних капацитета. Практичније решење би било, логичко памћење грешака. Односно, најпре би било потребно дефинисати класе најчешћих студентских грешака, потом развити подршку у систему да сваку итерацију у којој студент има неку грешку, систем аутоматски класификује и потом упамти.

Већи број нових функционалности би могао да буде представљен развојем нових модула који би покрили и остале, тренутно не покривене, теме из области база података. На пример, модул за физичко пројектовање база података, модул за управљање трансакцијама, потом модул за дистрибуиране базе података, полу-структуриране базе података засноване на XML, објектне базе података и објектне упитне језике и други модули. Управо би ту до изражаја дошле предности које доноси модуларна архитектура система ADVICE, јер би омогућила додавање нових модула и вертикалну интеграцију система у друге курсеве који имају базе података као предуслов.

Остварени резултати, приказани у овом раду, могу бити употребљени за потребе нових истраживања. Поступци провере сличности концептуалних модела могу

бити проширени поступцима обраде природног језика (natural language processing), који би на основу описа, односно текста проблема, формирали модел ентитета и односа, који би био употребљен за проверу исправности одговора студената. Овим би била елиминисана потреба да предавач формира концептуални модел за тачно решење. Слично, поступци провере сличности логичких модела могу бити проширени поступцима провере задовољења ограничења (constraint satisfaction problem), који би на основу дефиниција табела, одредили почетни садржај базе података довољан за проверу исправности. Овим би била елиминисана потреба да предавач формира исказе за потребе скрипти за попуњавање иницијалног садржаја базе података и за потребе скрипти за додатна тестирања.

Принципи интерактивне провере сличности, иако у овом раду приказани у домену образовања, могу бити прилагођени потребама алата које би инжењери користили у свакодневном раду са базама података. Алат би могао да буде коришћен приликом дизајна комплексних база података за потребе интеграције погледа, или приликом интеграције хетерогених софтверских система ради интеграције на нивоу података. Такав алат би имао предности у односу на остале алате, због свог интегралног приступа провери сличности концептуалних и логичких модела. Такође, одређивање контра примера би помогло да корисник алата у сваком кораку разрешава по један конфликт и тако постепено конвергира ка крајњем моделу.

Многи други веома важни сценарији даљих истраживања и праваца развоја су могући. Процена употребљивости и примена резултата приказаних у овом раду у контексту тих сценарија оставља доста простора за даља истраживања. Очигледно

је да би даља истраживања могла да имају не само формално већ и јако практично усмерење.

7. Референце

[1] V. Blagojević, M. Cvetanović, and D. Belić: Модел објеката и односа као алтернативни приступ моделирању података, Etran 54., Donji Milanovac, Srbija, 2010.

[2] C. Parent and S. Spaccapietra: Issues and approaches of database integration. Communicatiosns of ACM, Vol. 41, No. 5, pages: 166–178, 1998.

[3] P.A. Bernstein and E. Rahm: Data warehouse scenarios for model management. Proc. 19th Int. Conf. Conceptual modeling, Springer-Verlag, Berlin, pages: 1–15, 2000.

[4] T. Berners-Lee, J. Hendler, and O. Lassila: The semanticweb. Sci Am, Vol. 284, No. 5, pages: 34–43, 2000.

[5] S. Castano and V. De Antonellis: Global viewing of heterogeneous data sources. IEEE Trans. Knowledge and Data Engineering, Vol.13, No.2, pages: 277–297, 2001.

[6] Q. Wang, J. X. Yu, and K. Wong: Approximate Graph Schema Extraction for Semi-Structured Data. Proc. 7th Int. Conf. Extending Database Technology: Advances in Database Technology, Springer-Verlag, London, pages: 302–316, 2000.

[7] M.Cvetanović: Metodologija integracije softverskih sistema bazirana na principima reverznog inženjerstva. Magistarska teza, Elektrotehnički fakultet, Beograd, Srbija i Crna Gora 2006.

[8] M. Cvetanović and Z. Radivojević: Implementacija rešenja integracije softverskih sistema. Telfor 13., Beograd, Srbija i Crna Gora, 2005.

[9] Z. Radivojević, M. Cvetanović, V. Milutinović, J. Sievert: Data Mining: A Brief Overview and Recent IPSI Research, Annals of Mathematics, Computing & Teleinformatics, Vol. 1, No. 1, Pages: 84-91, 2003.

[10] Rahm, E., P.A. Bernstein: A Survey of Approaches to Automatic Schema Matching. VLDB Journal, 10(4), 2001

[11] H. H. Do, S. Melnik, and E. Rahm: Comparison of schema matching evaluations. In Web, Web-Services, and Database Systems, pages 221–237, 2002.

[12] Do Hong Hai: Schema matching and mapping-based data integration, Ph.d. Thesis, Interdisciplinary Center for Bioinformatics and Department of Computer Science, University of Leipzig, Germany, August 2005

[13] Hall, P., G. Dowling: Approximate String Matching. ACM Computing Survey 12(4), 381-402, 1980

[14] Winkler, W.E.: Advanced Methods for Record Linking. Section on Survey Research Methods (American Statistical Association), 1994

[15] Embley, D.W., D. Jackmann, L. Xu: Multifaceted Exploitation of Metadata for Attribute Match Discovery in Information Integration. Proc. 1. Intl. Workshop Information Integration on the Web (WIIW), 2001

[16] Do, H.H., E. Rahm: COMA - A System for Flexible Combination of Schema Matching Approach. Proc. Intl. Conf. Very Large Databases (VLDB), 2002

[17] Madhavan, J., P.A. Bernstein, A.H. Doan, A. Halevy: Corpus-based Schema Matching. Proc. Intl. Conf. Data Engineering (ICDE), 2005

[18] Madhavan, J., P.A. Bernstein, E. Rahm: Generic Schema Matching with CUPID. Proc. Intl. Conf. Very Large Databases (VLDB), 2001

[19] Embley, D.W., D. Jackmann, L. Xu: Multifaceted Exploitation of Metadata for Attribute Match Discovery in Information Integration. Proc. 1. Intl. Workshop Information Integration on the Web (WIIW), 2001

[20] Giunchiglia, F., P. Shvaiko, M. Yatskevich: S-Match: an Algorithm and an Implementation of Semantic Matching. Proc. 1st European Semantic Web Symposium (ESWS), 2004

[21] Giunchiglia, F., M. Yatskevich, E. Giunchiglia: Efficient Semantic Matching. Proc. 2nd European Semantic Web Conf. (ESWC), 2005

[22] Do, H.H., E. Rahm: COMA - A System for Flexible Combination of Schema Matching Approach. Proc. Intl. Conf. Very Large Databases (VLDB), 2002

[23] Doan, A.H., P. Domingos, A. Halevy: Reconciling Schemas of Disparate Data Sources - A Machine-Learning Approach. Proc. ACM SIGMOD Intl. Conf. Management of Data, 2001

[24] Madhavan, J., P.A. Bernstein, E. Rahm: Generic Schema Matching with CUPID. Proc. Intl. Conf. Very Large Databases (VLDB), 2001

[25] Palopoli, L., G. Terracina, D. Ursino: The System DIKE - Towards the Semi-Automatic Synthesis of Cooperative Information Systems and Data Warehouses. Proc. ADBIS-DASFAA, 108-117, 2000

[26] Clifton, C., E. Housman, A. Rosenthal: Experience with a Combined Approach to Attribute-Matching Across Heterogeneous Databases. Proc. IFIP 2.6 Working Conf. Database Semantics, 1996

[27] Ide, N., J. Veronis: Word Sense Disambiguation - State of the Art. Computational Linguistics, 1998, 24(1)

[28] Larson, J, N. Shamkant, R. Elmasri: A Theory of Attribute Equivalence in Databases with Application to Schema Integration. IEEE Trans. on Software Engingeering 15(4), 449-463, 1989

[29] Lee, M.L, L.H. Yang, W. Hsu, X. Yang: XCLUST - Clustering XML Schemas for Effective Integration. Proc. Intl. Conf. Information and Knowledge Management (CIKM), 2002

[30] Noy, N.F., M.A. Musen: The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping. International Journal of Human-Computer Studies, 59 (6) pp. 983-1024, 2003

[31] Melnik, S., H. Garcia-Molina, E. Rahm: Similarity Flooding - A Versatile Graph Matching Algorithm. Proc. Intl. Conf. Data Engineering (ICDE), 2002

[32] Aumüller, D., H.H. Do, S. Massmann, E. Rahm: Schema and Ontology Matching with COMA++ (Software Demonstration). Proc. 24. ACM SIGMOD Intl. Conf. Management of Data, 2005

[33] Ehrig, M, S. Staab: QOM - Quick Ontology Mapping. Proc. 3. Intl. Semantic Web Conf. (ISWC), 2004

[34] C. Hernández, F. Prieto, M. A. Laguna, and Y Crespo: Formal Concept Analysis support for Conceptual Abstraction in Database Reengineering. Proc. Database Management and Reengineering Workshop (ICSM), 2002.

[35] D. Bojić: An Approach to Reverse Engineering of Use Cases. Doctoral Thesis, Faculty of Electrical Engineering, Belgrade, 2001.

[36] M. Cvetanović and D. Bojić: Architectural Investigation of XCTL by URCA. 3rd International Workshop on Software Engineering Education and Reverse Engineering, Ohrid, Macedonia, 2003.

[37] B. Ganter and R. Wille: Formal Concept Analysis: Mathematical Foundations, Springer-Verlag, Berlin, ISBN 3-63311-62767-5, 1998.

[38] C. Carpineto and G. Romano: Concept Data Analysis: Theory and Applications, Wiley, ISBN 978-0-470-85055-8, 2004.

[39] S. Melnik, H. Garcia-Molina, and E. Rahm: Similarity Flooding: A Versatile Graph Matching Algorithm, Extended Technical Report, Stanford University, 2001.

[40] T. Connolly and C. Begg: Database Systems: A Practical Approach to Design, Implementation and Management. Addison Wesley, 4 ed., ISBN 0321210255, 2004.

[41] V.Blagojević: Relacione baze podataka 1. ICNT, Beograd, ISBN 86-7652-004-6, 2006.

[42] E.F. Codd: A relational model of data for large shared data banks. Communications of the ACM, Vol. 13, No.6, pages:377–387, 1970.

[43] E.F. Codd: Further Normalization of the Data Base Relational Model. Courant Computer Science Symposia Series 6: Data Base Systems, New York City, May 24–25, 1971.

[44] C. Zaniolo: A New Normal Form for the Design of Relational Database Schemata. ACM Transactions on Database Systems vol. 7, no.3, 1982.

[45] E.F. Codd: Recent Investigations into Relational Data Base Systems. IBM Research Report RJ1385, April 23, 1974.

[46] R. Fagin: Multivalued Dependencies and a New Normal Form for Relational Databases. ACM Transactions on Database Systems, vol. 2, no. 1, 1977.

[47] R. Fagin: Normal Forms and Relational Database Operators. ACM SIGMOD International Conference on Management of Data, Boston, 1979.

[48] R. Fagin: A Normal Form for Relational Databases That Is Based on Domains and Keys, Communications of the ACM, vol. 6, pages: 387–415, 1981.

[49] C.J. Date, H. Darwen, and N. Lorentzos: Temporal Data and the Relational Model. Morgan Kaufmann, 2002.

[50] E. F. Codd: Data Base Systems – Relational Completeness Of Data Base Sublanguages, Courant Computer Science Symposia Series, vol. 6, Prentice Hall, 1972.

[51] C.J. Date: Introduction to Database Systems, 6th Edition, Computer-Science, Addison-Wesley, 1995.

[52] E. F. Codd: Relational Completeness of Data Base Sublanguages. In: R. Rustin (ed.): Database Systems: 65-98, Prentice Hall and IBM Research Report RJ 987, San Jose, California, 1972.

[53] P. Kolaitis and M. Vardi: Conjunctive-Query Containment and Constraint Satisfaction", Journal of Computer and System Sciences, vol. 61, pages 302–332, 2000.

[54] M. Vardi: The Complexity of Relational Query Languages. In. Proceedings of the fourteenth annual ACM symposium on Theory of computing, pages 137–146, 1982.

[55] S. Abiteboul, R. Hull, V. Vianu: Foundations of Databases. Addison-Wesley, ISBN 0-201-53771-0, 1995.

[56] M. Yannakakis: Algorithms for Acyclic Database Schemes. Proc. VLDB, pages: 82-94, 1981.

[57] G. Gottlob, N. Leone, and F. Scarcello: Hypertree Decompositions and Tractable Queries. Journal of Comput. Syst. Sci. vol. 64, no. 3, pages 579-627, 2002.

[58] G. Gottlob, C. Koch, and K. U. Schulz: Conjunctive queries over trees. Journal of ACM, vol. 53, no. 2, pages: 238-272, 2006.

[59] J. Albert, Y. Ioannidis, and R. Ramakrishnan : Conjunctive query equivalence of keyed relational schemas. In *Proceedings of the sixteenth* PODS. ACM, pages: 44-50, 1997.

[60] R. Chirkova and M. R. Genesereth: Equivalence of SQL queries in presence of embedded dependencies. In *Proceedings of the twenty-eighth* PODS, ACM, pages: 217-226, 2009.

[61] S. Cohen, W. Nutt, and Y. Sagiv: Containment of Aggregate Queries. Proceedings of the 9th International Conference on Database Theory, pages: 111-125, 2003.

[62] T. Teorey, S. Lightstone, T. Nadeau, and H.V. Jagadish: Database Modeling and Design: Logical Design, 4th Ed., Morgan Kaufmann, ISBN 0126853525, 2005.

[63] The Joint Task Force on Computing Curricula IEEE Computer Society Association for Computing Machinery, "Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering," Final Report, December 2004, Available: http://www.acm.org/education/education/curric_vols/CE-Final-Report.pdf, (accessed July 2010).

[64] M. Conklin and L. Heinrichs: In search of the right database text. Journal of Computing Sciences in Colleges, Vol. 21, No. 2, pages: 305-312, 2005.

[65] M. A. Robbert and C. M. Ricardo: Trends in the evolution of the database curriculum. Proceedings of the 8th annual conference on Innovation and technology in computer science education, Thessaloniki, Greece, pages: 139-143, 2003.

[66] M. Cvetanović, Z. Radivojević, V. Blagojević, M. Bojović: ADVICE—Educational System for Teaching Database Courses. IEEE Transactions on Education, Vol. 54, No. 3, pages: 398-409, 2011.

[67] A. Mitrovic and B. Martin: Evaluating the Effects of Open Student Models on Learning. Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Sonthofen, Germany, pages: 296-305, 2002.

[68] A. Mitrovic: NORMIT: a Web-enabled tutor for database normalization. Proceedings of the International Conference on Computers in Education, Vol. 2, pages: 1276-1280, 2002.

[69] P. Suraweera and A. Mitrovic: KERMIT: a Constraint-based Tutor for Database Modeling. Proceedings of the 6th International Conference on Intelligent Tutoring Systems, Biarritz, France, pages: 377-387, 2002.

[70] H. Laine: SQL-Trainer. Proceedings of the First Annual Finnish/Baltic Sea Conference on Computer Science Education, Koli Calling, Finland, pages: 13-17, 2002.

[71] J. D. Ullman: Gradiance On-Line Accelerated Learning. Proceedings of the Twenty-eighth Australasian conference on Computer Science, Newcastle, Australia, Vol. 38, pages: 3-6, 2005.

[72] SQL Course, Available: http://www.sqlcourse.com/, (accessed August 2007).

[73] S. W. Dietrich, E. Eckert, and K. Piscator: WinRDBI: a windows-based relational database educational tool. Proceedings of the 28th ACM SIGCSE Technical Symposium on Computer Science Education, San Jose, California, Vol. 29, No. 1, pages: 126-130, 1997.

[74] H. J. Kung and H. L. Tung: A Web-Based Tool To Enhance Teaching/Learning Database Normalization. Proceedings of the 2006 Southern Association for Information Systems Conference, Jacksonville, Florida, pages: 251-258, 2006.

[75] N. Georgiev: A Web-Based Environment for Learning Normalization of Relational Database Schemata. M.S. thesis, Department of Computer Science, Umea University, Umea, Sweden, 2008.

[76] M. Murray and M. Guimaraes: Animated database courseware: using animations to extend conceptual understanding of database concepts. Journal of Computing Sciences in Colleges, Vol. 24, No. 2, pages: 144-150, 2008.

[77] M. Guimaraes: The Kennesaw Database Courseware (KDC): strong points, weak points, and experience using it in a classroom environment. Journal of Computing Sciences in Colleges, Vol. 21, No. 3, pages: 91-96, 2006.

[78] A. Mitrovic, P. Suraweera, B. Martin, and A. Weerasinghe: DB-suite: Experiences with Three Intelligent, Web-based Database Tutors. Journal of Interactive Learning Research, Vol. 15, No. 4, pages: 409-432, 2004.

[79] Z. Radivojević, M. Cvetanović, Z. Jovanović: Reengineering the SLEEP simulator in a concurrent and distributed programming course. Computter Applications in Enginerring Education, doi: 10.1002/cae.20527, 2011.

[80] E. Rahm and P. A. Bernstein,: A survey of approaches to automatic schema matching. The International Journal on Very Large Data Bases, Vol. 10, No. 4, pages: 334-350, 2001.

[81] S. Melnik, H. Garcia-Molina, and E. Rahm: Similarity flooding: a versatile graph matching algorithm and its application to schema matching. Proceedings of 18th International Conference on Data Engineering, San Jose, California, pages: 117-128, 2002.

[82] M. Harao, K. Yamada, and K. Hirata: Efficient Second-Order Predicate Schema Matching Algorithm. Proceedings of the fourth Korea-Japan Joint Workshop on Algorithms and Computation, Seoul, Korea, pages: 31-38, 1999.

[83] S. Cohen: Equivalence of queries combining set and bag-set semantics. Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, Chicago, Illinois, pages: 70–79, 2006.

[84] J. Huo: KMVQL: a Visual Query Interface Based on Karnaugh Map. Proceedings of the working conference on Advanced visual interfaces, Napoli, Italy, pages: 243-250, 2008.

[85] L. Hu, K. A. Ross, Y. C. Chang, C. A. Lang, and D. Zhang: QueryScope: visualizing queries for repeatable database tuning. Proceedings of the VLDB Endowment, Auckland, New Zealand, Vol. 1, No. 2, pages: 1488-1491, 2008.

[86] H. Tardieu, A. Rochfeld, and R. Colletti: La Méthode Merise: Principes et outils, Paris, France: Editions d'Organisation, 2000.

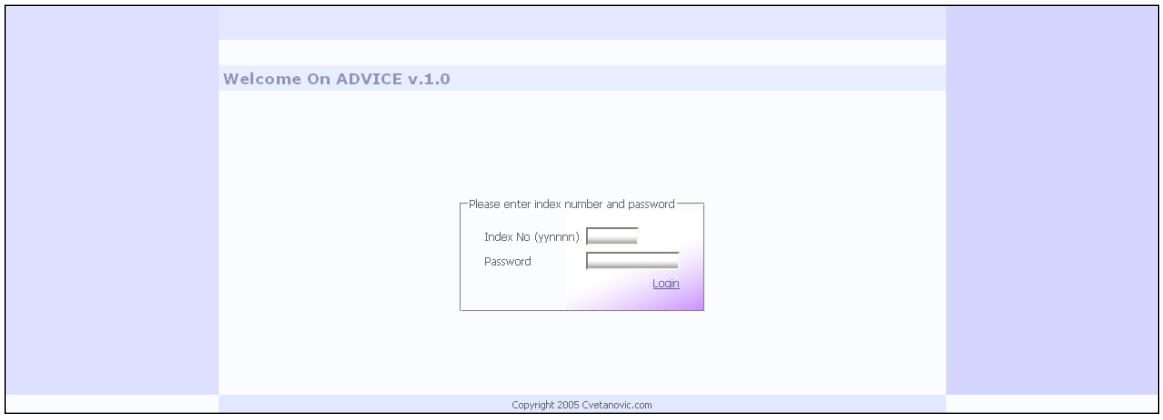
[87] A. Silberschatz, H. F. Korth, and S. Sudarshan: Database System Concepts, 5th ed., New York: McGraw-Hill, 2005.

[88] Academic Ranking of Worldwide Universities 2010, Shanghai Ranking Consultancy, http://www.arwu.org/ARWU2010.jsp.

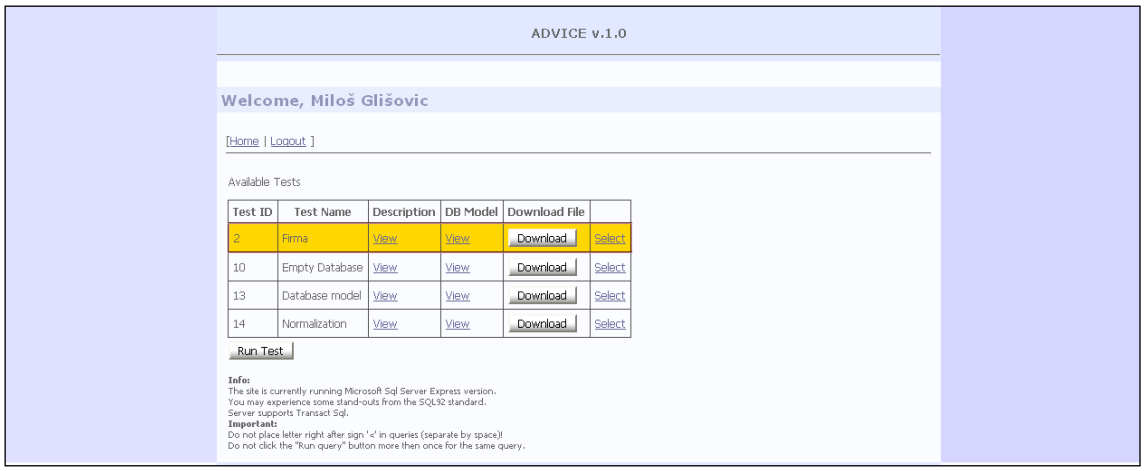
Прилог А – Приказ појединих екрана система ADVICE

У овом прилогу су дате слике појединих екрана система ADVICE. Слике су подељене у две групе, прву коју чине екрани из перспективе студената и другу коју чине екрани из перспективе предавача.

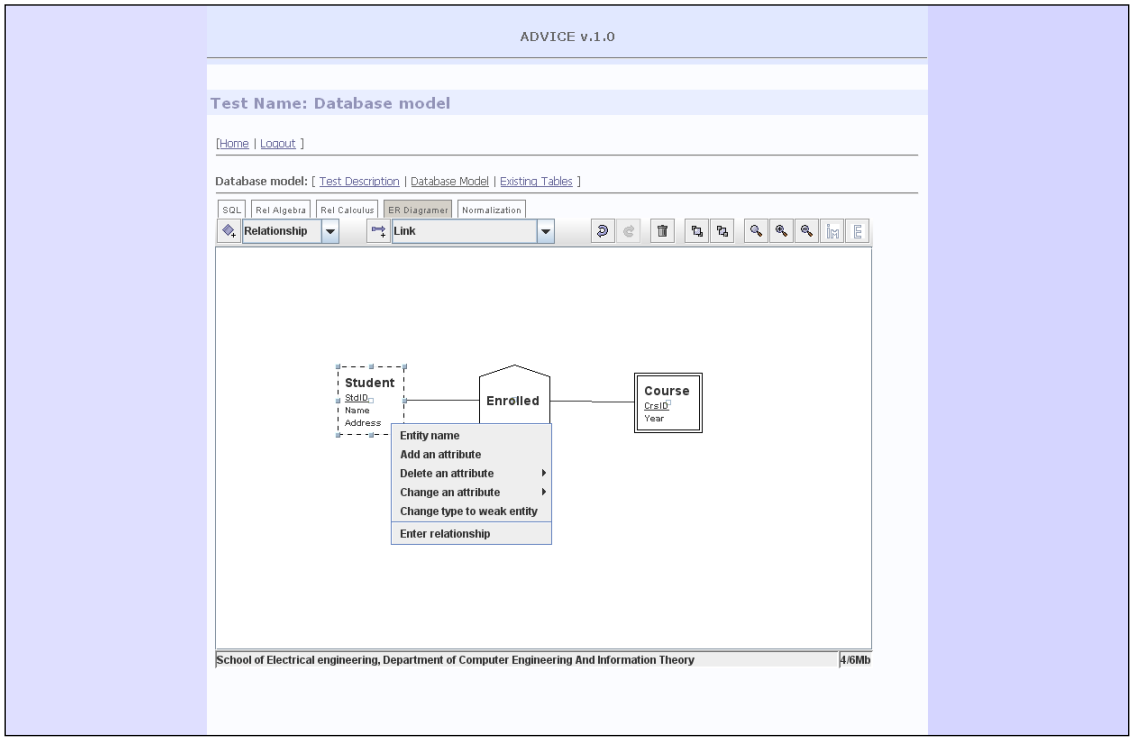
Перспектива студента



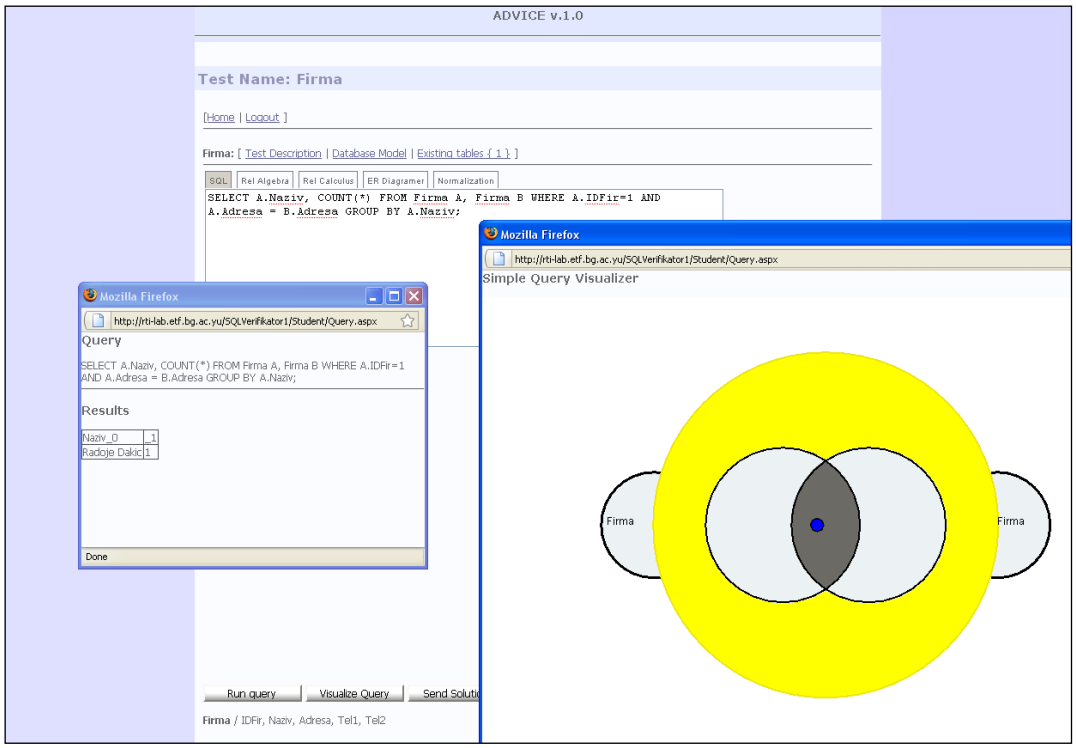
Прилог А. 1 Пријављивање студента на систем



Прилог А. 2 Одабир вежбе



Прилог А. 3 Употреба алата за концептуалне моделе

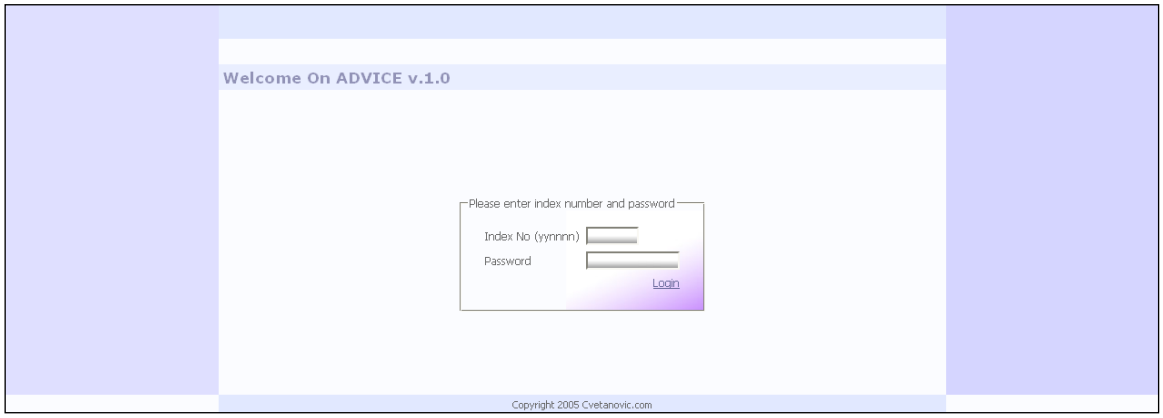


Прилог А. 4 Употреба алата за SQL

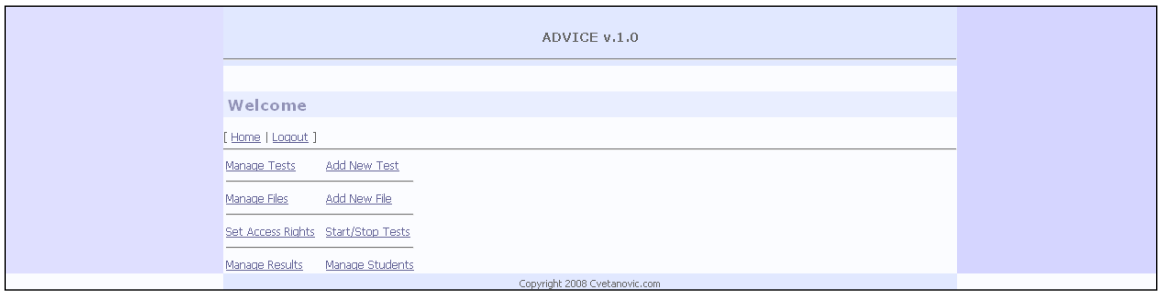
Прилог А. 5 Употреба алата за релациону алгребру

Прилог А. 6 Употреба алата за релациони рачун

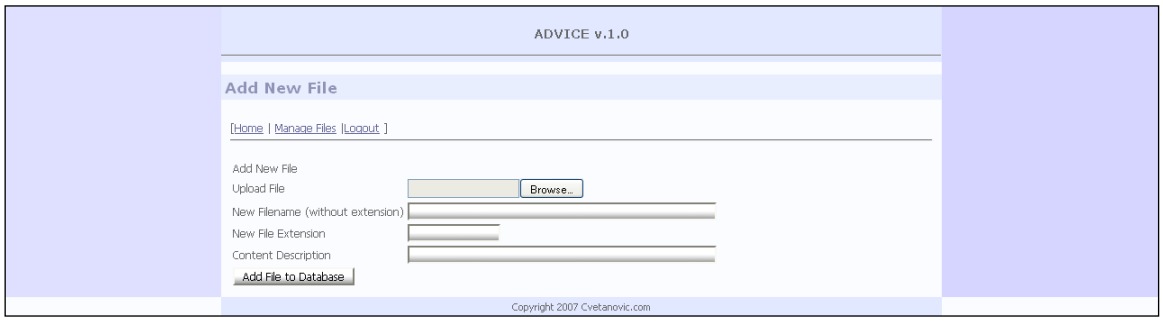
Перспектива предавача



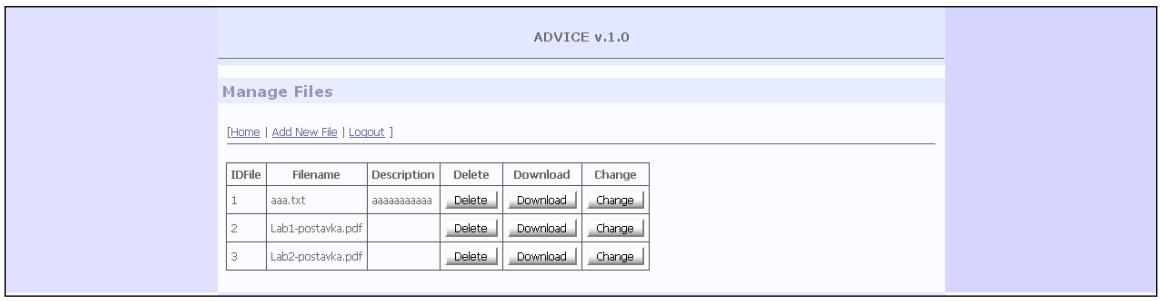
Прилог А. 7 Пријављивање предавача на систем



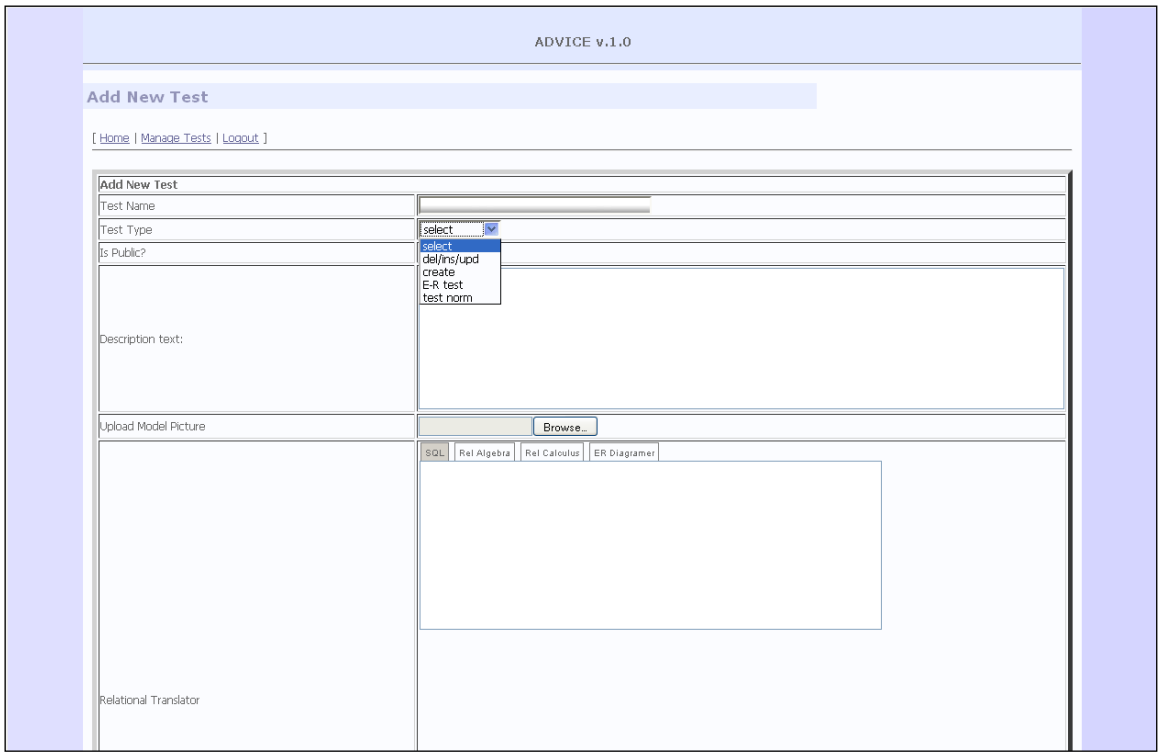
Прилог А. 8 Опције на располагању предавачу



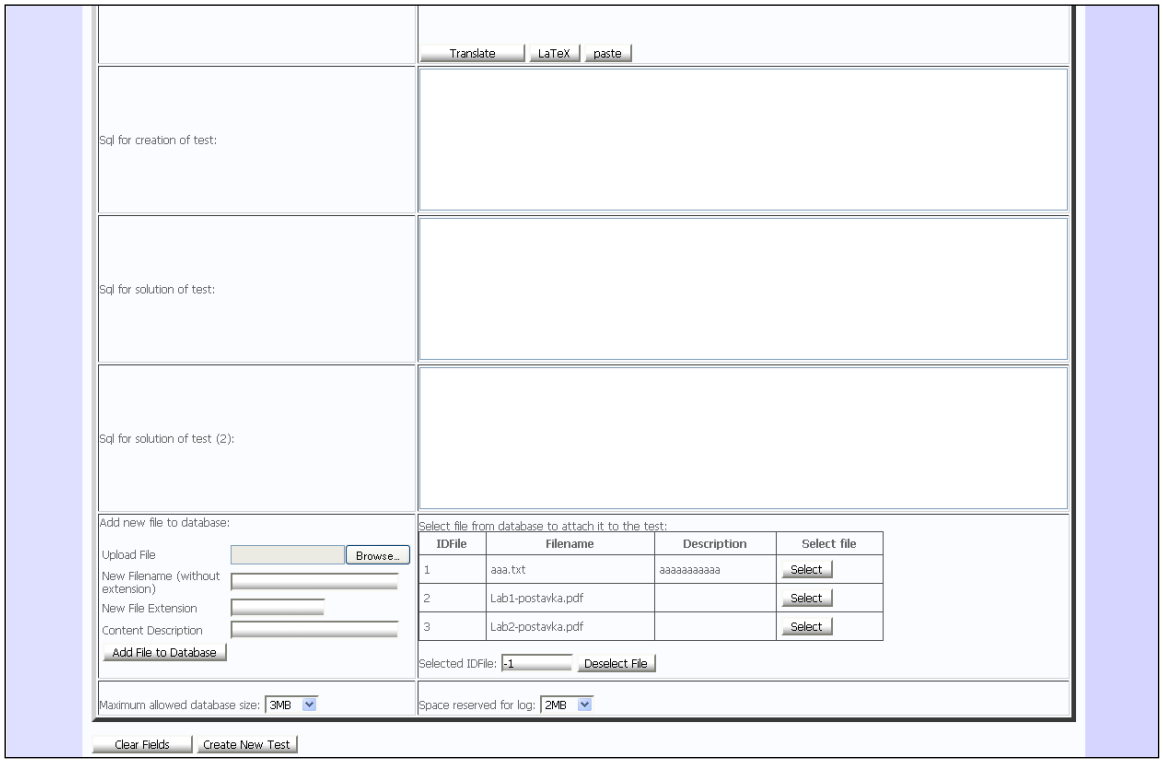
Прилог А. 9 Додавање нових датотека са ресурсима



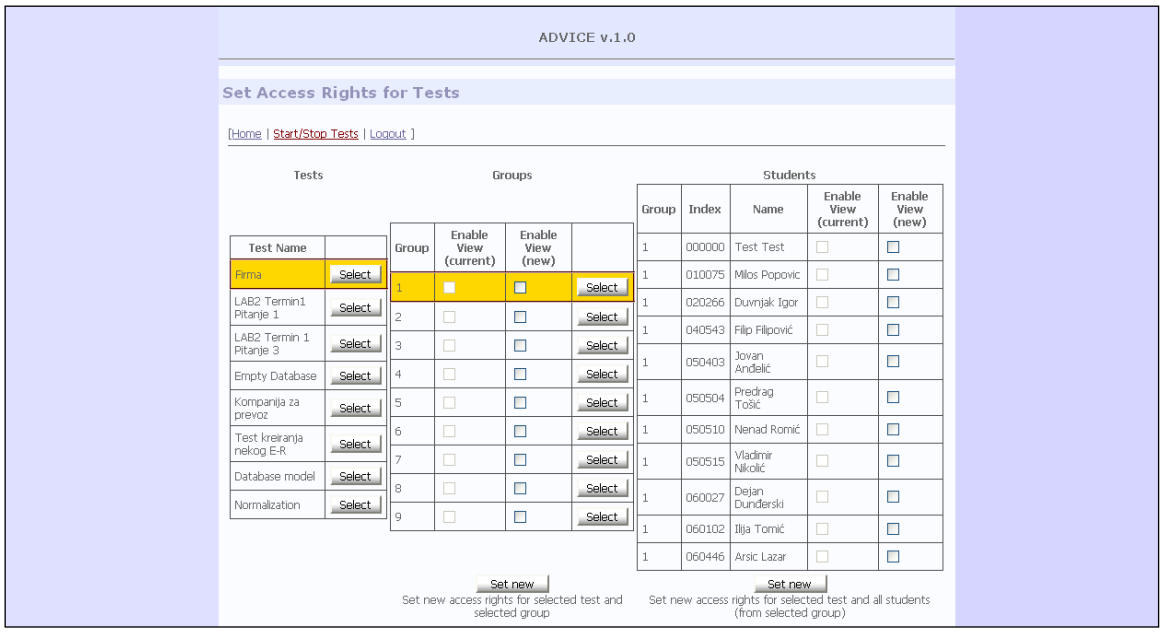
Прилог А. 10 Управљање датотекама са ресурсима



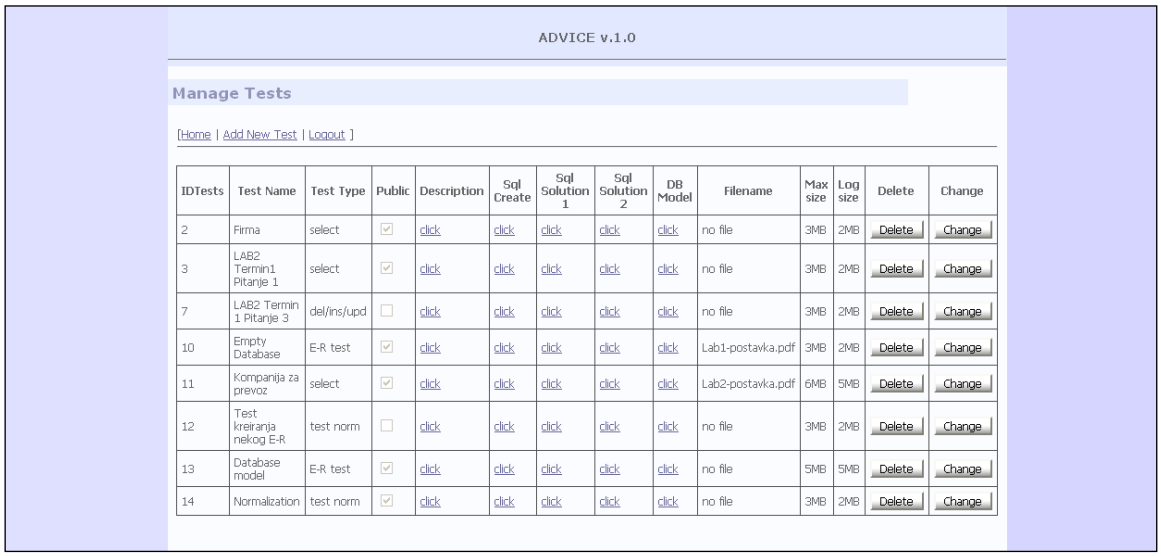
Прилог А. 11 Додавање нових вежби (први део)



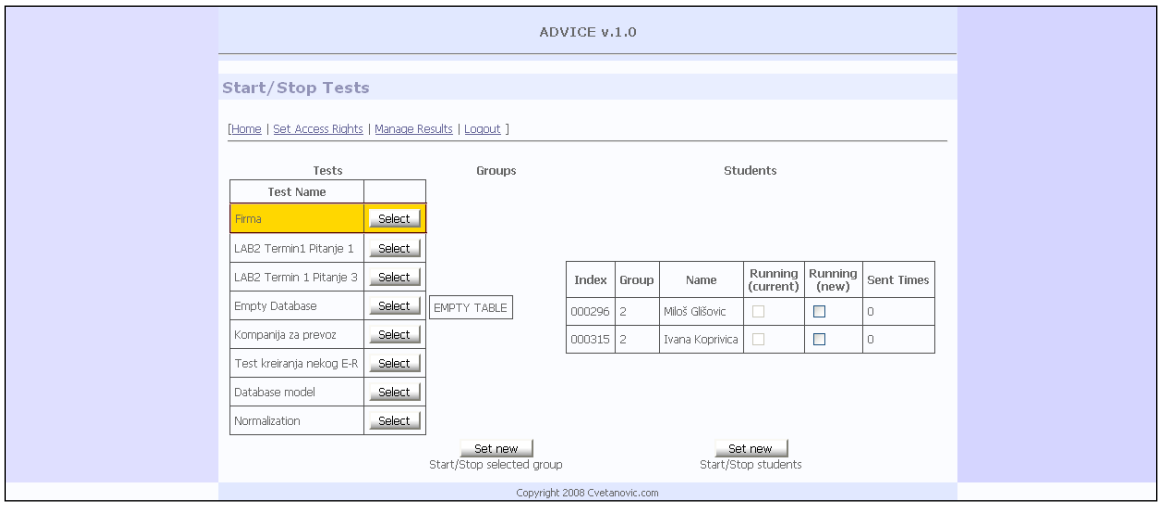
Прилог А. 12 Додавање нових вежби (други део)



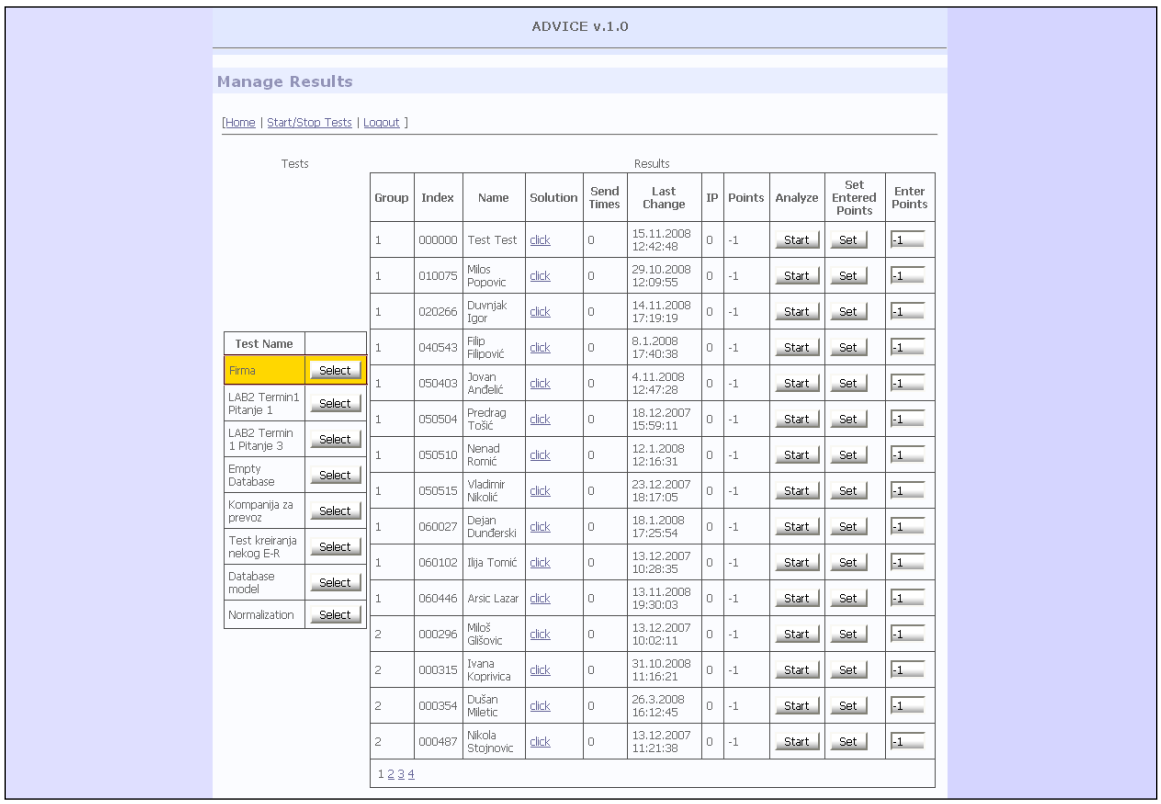
Прилог А. 13 Дефинисање права приступа вежбама



Прилог А. 14 Управљање вежбама



Прилог А. 15 Покретање и заустављање тестова



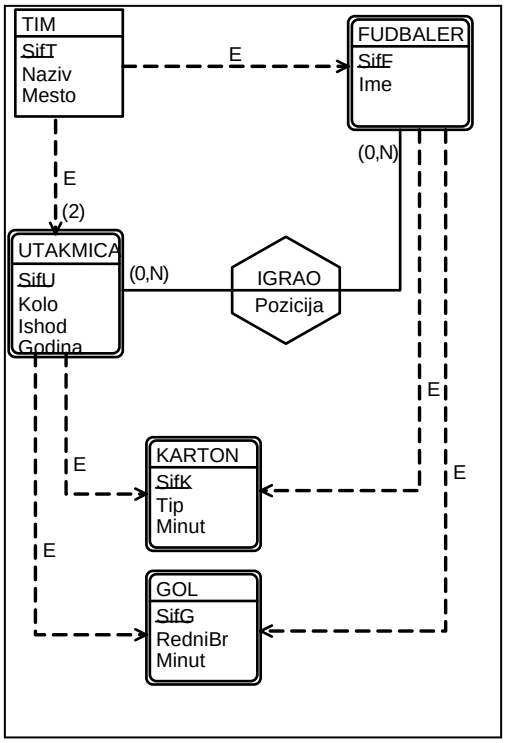
Прилог А. 16 Управљање резултатима тестова

Прилог Б – Тест примери концептуалних модела

У наставку су дати примери концептуалних модела коришћених при евалуацији система ADVICE.

Фудбалски савез

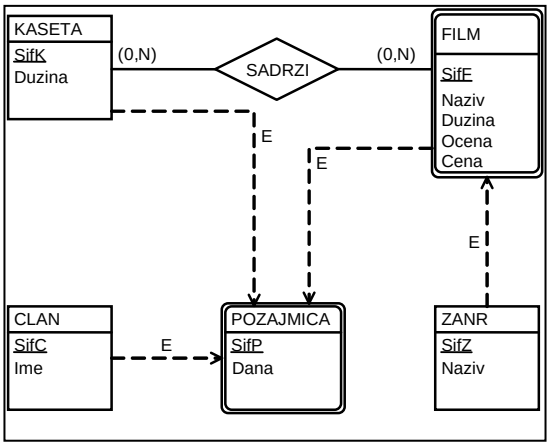
Посматрани систем је фудбалски савез који евидентира утакмице одигране у току једне сезоне. За сваку утакмицу се трајно чувају подаци о томе који тимови су играли, у ком колу је та утакмица одиграна (и које године), какав је исход утакмице био, који су све играчи играли и на којим позицијама. За сваког фудбалера се памти име и тим за који игра, док се за тимове памти назив и место из кога долазе. Претпоставка је да фудбалери не могу да мењају тим у коме играју, у току сезоне. Такође је потребно трајно евидентирати сваки постигнути гол као и то који фудбалер га је постигао, у ком минуту и који је то гол био по реду на посматраној утакмици. Поред голова трајно се евидентирају и сви картони, жути и црвени, додељени на утакмици и то ком фудбалеру и у ком минуту.



Прилог Б. 1 Концептуални модел фудбалског савеза

Видео клуб

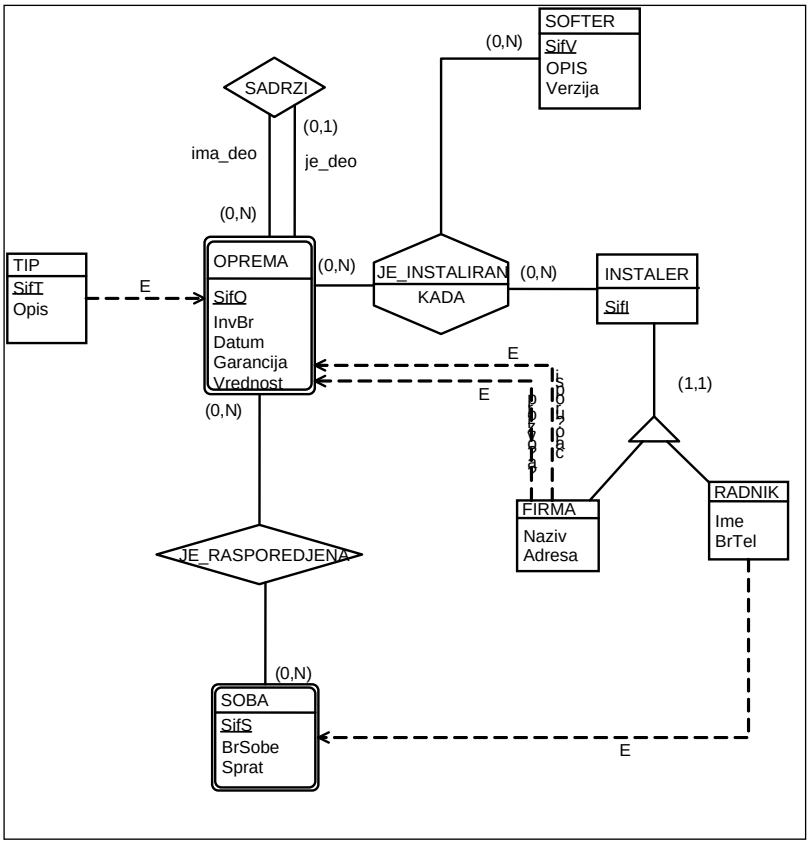
Посматрани систем је видео клуб који члановима издаје касете са снимљеним филмовима. Касете могу да буду различите дужине и на себи могу да садрже ниједан, један или више филмова. За сваки филм се евидентира његов назив, дужина, оцена која може и не мора да постоји, цена и жанр коме припада. За сваки жанр се прати његов назив. У систему се такође евидентира свака позајмица и то тако што се памти који члан је позајмио коју касету, због ког филма и на колико дана.



Прилог Б. 2 Концептуални модел видео клуба

Инвентар компаније

Пројектовати базу података за потребе вођења ИТ инвентара (опреме) у некој компанији. У систему треба водити евиденцију о инвентарском броју, датуму почетка употребе, трајању гаранције, вредности, типу и опису опреме, а такође и о називу и адреси фирме која је испоручила опрему а која уједно може, а не мора, бити произвођач те опреме. Опрема може бити распоређена у једној или више соба које припадају компанији, при том свака соба поседује јединствени идентификациони број и информацију на ком се спрату налази. Свака соба мора имати дефинисану одговорну особу (име, број телефона) која је радник компаније. Опрема може да се састоји из више делова који су такође опрема. У систему је потребно водити евиденцију и о разноврсном софтверу који може бити инсталиран на већем броју опреме, при том је потребно водити евиденцију о томе ко је и када обавио инсталацију. Инсталацију може обавити особа која је радник компаније или нека од фирми са којом компанија сарађује.

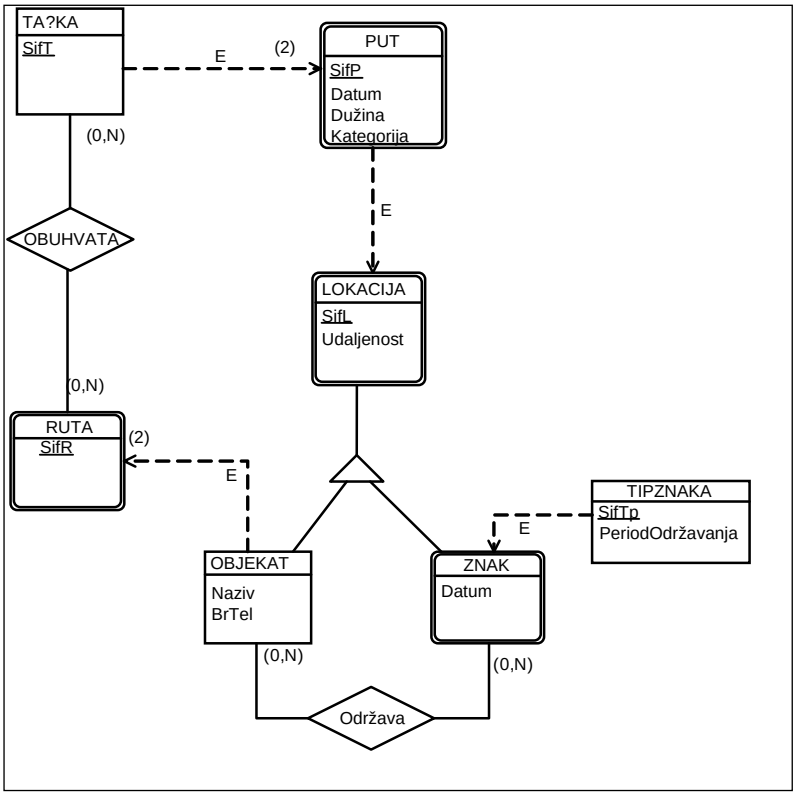


Прилог Б. 3 Концептуални модел инвентара компаније

Евиденција путева

Посматра се база података намењена евиденцији путева. Сваки пут је дефинисан двема тачкама између којих се простире, при чему је смер дефинисан почетном и крајњом тачком. За сваки пут се поред његове дужине, датума изградње води и евиденција о категорији (три предефинисана типа: аутопут, споредни пут, улица). На путу се може налазити више путних локација (удаљеност од почетне тачке пута), које могу бити или путни знак или објекат на путу (назив, број телефона). У случају путног знака води се евиденција о датуму постављања и типу знака (шифра, период одржавања). Објекат на путу може бити задужен да одржава неке од знака на путу. У бази треба предвидети могућност чувања руте између два

објекта на путу, која се може састојати из произвољног броја тачака (тачака нема у случају када се објекти налазе на истом путу).

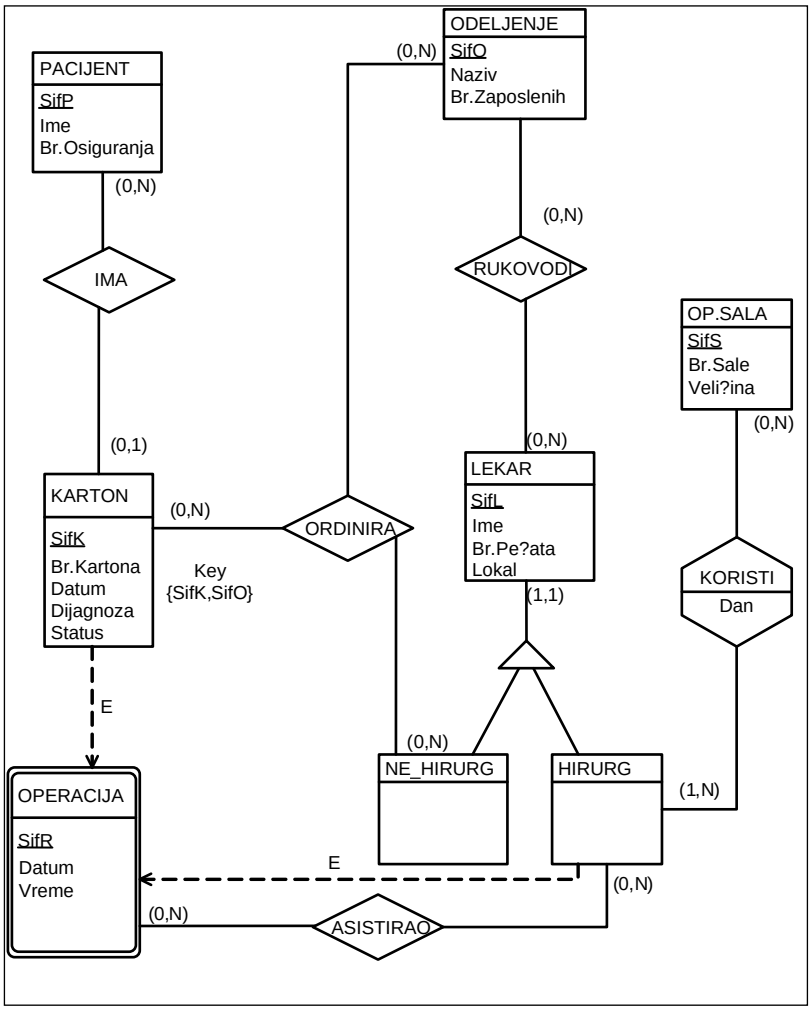


Прилог Б. 4 Концептуални модел евиденције путева

Болница

Посматра се део информационог система болнице. Болница има више одељења (назив, број запослених). Одељењем може да руководи највише један лекар (име, бр.печата, локал). Пацијенти (име, бр.осигурања) могу имати више отворених картона, при том се за сваки прати бр.картона, датум отварања, дијагноза и статус. Постоји захтев да пацијент коме је отворен картон може бити анониман. Болница располаже са више операционих сала за које се прати бр.сале и величина. Лекари који су хирурзи морају имати бар једну додељену операциону салу коју могу да користе одређеним даном у недељи. У систему се води евиденција о обављеним операцијама тако што се за сваку прати датум, време, информација о одговорном

хирургу, картон на који се операција односи као и о томе који су хирурзи асистирали. Сваки картон може имати одређеног највише једног ординирајући лекар на сваком од одељења, и при том лекар не сме бити хирург.

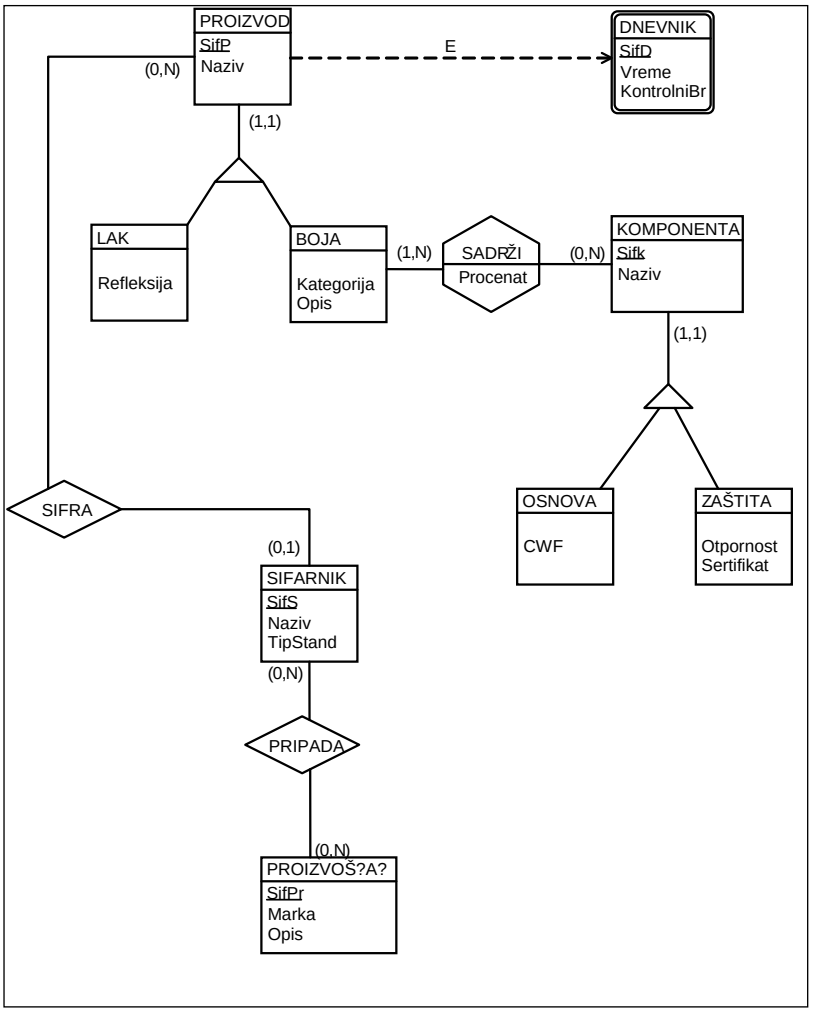


Прилог Б. 5 Концептуални модел болнице

Фабрика боја и лакова

Посматра се фабрика за производњу боја и лакова. Свака боја (води се евиденција о називу, категорији и опису боје) се производи на основу формуле о садржају одређених компоненти у одређеном процентуалном износу. При том, за сваку

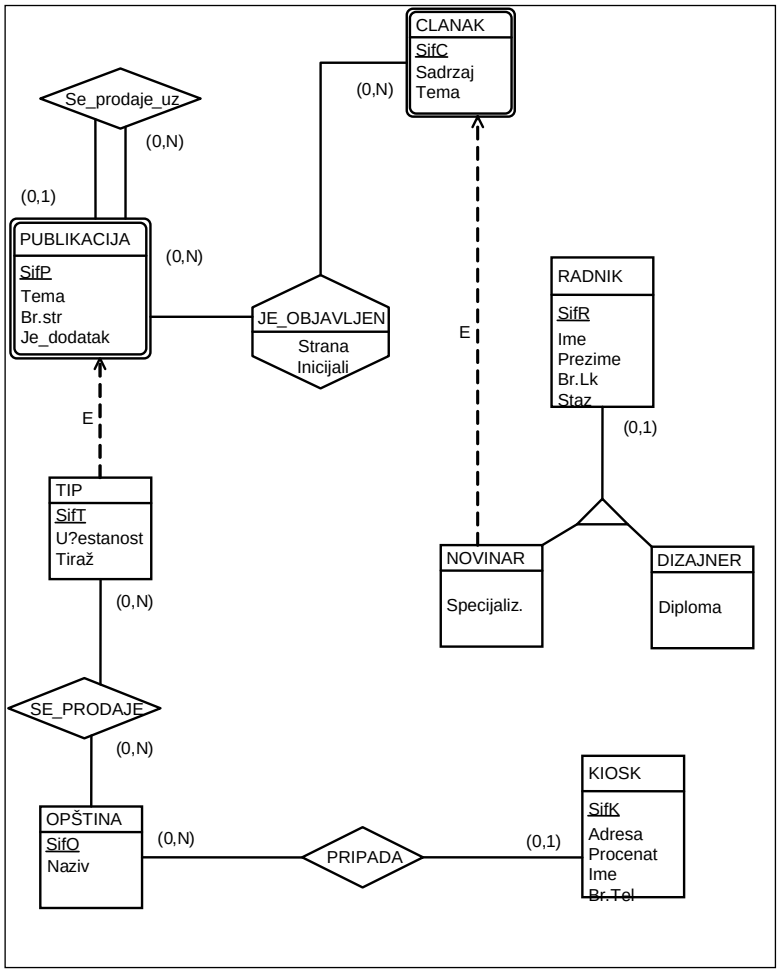
компоненту постоји информација о називу и CWF фактору уколико се ради о основној компоненти, односно називу, отпорности и сертификату уколико се ради о заштитној компоненти. За сваки лак се води евиденција о називу и степену рефлексије. У систему је дефинисан јединствен шифарник у коме за сваки лак и сваку боју може постојати више шифара, при чему се за сваку прати назив и тип стандарда. Такође је потребно водити евиденцију о томе који све произвођачи користе ту шифру. За сваког произвођача се у систему чува његова марка и опис. У фабрици постоји дневник, у коме се за сваки производ, који се направи, чува информација о датуму производње и контролном броју.



Прилог Б. 6 Концептуални модел фабрике боје и лакова

Новинска кућа

Пројектовати систем за потребе новинске куће. У систему треба водити евиденцију о запосленима (име, презиме, бр.л.к, стаж), од којих неки могу бити новинари (специјализација) или дизајнери (диплома). Посматрана новинска кућа издаје периодике тј. публикације које морају бити неког дефинисаног типа. Тип треба да дефинише учестаност излажења, тираж, као и скуп општина у којима излази (тј. у којима се продаје). Публикација може имати додатак, за који се води евиденција о свим стварима као и за публикацију уз коју се продаје. Новинари пишу чланке, и за сваки чланак се прати у којим је публикацијама и додацима објављен, на којој страни, и под којим иницијалима. У бази се води евиденција о киосцима са којима новинска кућа има сарадњу. При том мора бити дефинисана општина којој припада киоск, а поред тога треба да постоје информације о контакт особи (име, бр.тел), адреси киоска, и договореном проценту од продаје.



Прилог Б. 7 Концептуални модел новинске куће

Прилог Ц – Тест примери логичких модела

У наставку су дати примери логичких модела коришћених при евалуацији система ADVICE.

Фудбалски савез

Дата је шема релационе базе података фудбалског савеза за потребе евиденције утакмица једне сезоне:

FUDBALER (SifF, Ime, SifT); TIM (SifT, Naziv, Mesto);

UTAKMICA (SifU, SifTDomaci, SifTGost, Kolo, Ishod, Godina);

IGRAO (SifF, SifU, PozicijaIgraca); GOL (SifG, SifU, SifF, RedniBrGola, Minut);

KARTON (SifK, SifU, SifF, Tip, Minut);

1. Саставити SQL скрипт којим се формира табела УТАКМИЦА, уколико је познато да утакмицу играју два различита тима чије се шифре налазе у табели ТИМ, да исход утакмице може бити из скупа вредности {X-нерешено, 1-победа домаћих, 2-победа гостију} и да постоји свега 30 кола у којима се утакмице играју.

У току сезоне сваки пар тимова одигра две утакмице, при чему је једна на домаћем, а друга на гостујућем терену.

```
CREATE  TABLE Utakmica(
SifU INT PRIMARY KEY,
SifTDomaci INT NOT NULL REFERENCES Tim(SifT)
                            ON DELETE NO ACTION
                            ON UPDATE CASCADE,
SifTGost INT NOT NULL REFERENCES Tim(SifT)
                           ON DELETE NO ACTION
                           ON UPDATE CASCADE,
Kolo INT NOT NULL  CHECK(Kolo BETWEEN 1 AND 30),
Ishod CHAR NOT NULL CHECK (Ishod IN ('X','1','2')),
Godina INT NOT NULL CHECK (Godina > 1990),
CHECK(SifTDomaci <> SifTGost),
UNIQUE(SifTDOmaci, SifTGost)
);
```

2. Саставити SQL скрипт који као резултат даје шифре и имена фудбалера као и просечан број голова по утакмици који су ти фудбалери дали, али само за оне фудбалере који су дали бар један гол након што су добили жути картон, и то у утакмици где њихов тим био гостујући.

```
CREATE VIEW BrUtakmica(SifF, BrojUtakmica)
AS
SELECT SifF, COUNT(*)
FROM Igrao
GROUP BY SifF;

CREATE VIEW BrGolova(SifF, BrGolova)
AS
SELECT SifF, COUNT(*)
FROM Gol
GROUP BY SifF;

CREATE VIEW DaliGolNakonKartona(SifF)
AS
SELECT SifF
FROM Fudbaler F, Utakmica U, Gol G, Karton K
WHERE F.SifF=G.SifF
AND F.SiF = K.SifF
AND G.Minut > K.Minut
AND G.SifU = K.SifU
AND G.SifU = U.SifU
AND F.SIfT = U.SifTGost;

SELECT F.SifF, F.Ime, BG.Brgolova / BU.BrUtakmica
FROM Fudbaler F, BrGolaova BG, BrUtakmica BU
WHERE F.SifF = BG.SifF
AND F.SifF = BU.SifF
AND F.SifF IN (SELCT SifF FROM DaliGolNakonKartona);
```

3. Саставити SQL скрипт који као резултат даје шифре и имена фудбалера, али само за оне фудбалере који су дали последња два гола на некој од утакмица на којој је њихов тим победио.

```
CREATE VIEW Uslov(SifF,SifU)
AS
SELECT U.SifU
FROM Fudbaler F, Utakmica U, Gol G
WHERE F.SifF = G.SifF
AND G.SifU = U.SifU
AND(   (F.SifT=U.SifTDomaci AND U.Ishod='1')
       OR
       (F.SifT=U.SifTGost AND U.Ishod='2')
    )
AND NOT EXSIST (    SELECT SifF
                    FROM Gol L
                    WHERE L.SifF <> F.SifF
                    AND L.SifU = U.SifU
                    AND L.Minut > G.Minut
                  )
GROUP BY SifF, SifU
HAVING COUNT(*) = 2;

SELECT DISTINCT F.SifF, F.Ime
FROM Fudbaler F, Uslov U
WHERE F.SifF = U.SifF;
```

4. Саставити упит релационе алгебре који даје шифре и имена оних фудбалера који су играли на свим утакмицама а нису дали ниједан гол.

$$\pi_{SifF,SifU}(Igrao) \rightarrow t_1(SifF, SifU)$$
$$\pi_{SifU}(Utakmica) \rightarrow t_2(SifU)$$
$$t_1 / t_2 \rightarrow t_3(SifF)$$
$$\pi_{SifF}(Gol) \rightarrow t_4(SifF)$$
$$t_3 - t_4 \rightarrow t_5(SifF)$$
$$\pi_{SifF,ime}\left(t_5 \underset{*}{\infty} Fudbaler\right) \rightarrow resenje(SifF, ime)$$

5. Саставити исказ релационог рачуна домена који даје шифре и називе оних тимова који су са неким другим тимом и на гостујућем и на домаћем терену постигли нерешен резултат.

```
{<SifT, Naziv>|∃Mesto(<SifT, Naziv, Mesto>∈TIM ∧ ∃SifU1, SifT1, Kolo1,
Ishod, Godina1(<SifU1, SifT1, Kolo1, Ishod, Godina1>∈UTAKMICA ∧ Ishod= ‘X’
∧
∧∃SifU2, Kolo2, Godina2 (<SifU2, SifT1, SifT, Kolo2, Ishod, Godina2>∈
UTAKMICA)))}
```

Видео клуб

Дата је шема релационе базе података (уз напомену да оцена филма не мора да буде унешена, у том случају је NULL):

FILM (SifF, Naziv, Duzina, Ocena, Cena, SifZ); KASETA (SifK, Duzina);

ZANR(SifZ, Naziv); CLAN (SifC, Ime); POZAJMICA (SifP, SifK, SifF, SifC, Dana);

SADRZI (SifK, SifF);

6. Саставити SQL скрипт који брише оне филмове који имају најмању оцену међу свим оним филмова који у свом називу садрже реч “seven”.

```
CREATE VIEW Minimum(Ocena)
AS
SELECT MIN (Ocena)
FROM Film
WHERE Naziv LIKE ’%SEVEN%’;

CREATE VIEW Sifre(SifF)
AS
SELECT SifF
FROM Film
WHERE Naziv LIKE ’%SEVEN%’
AND Ocena = (SELECT Ocena FROM Minimum);

DELETE FROM Film
WHERE SifF IN (SELECT SifF FROM Sifre);
```

7. Саставити SQL скрипт који даје шифру, име и укупну дужину одгледаних филмова оних чланова који су бар једном позајмили најгледанији филм (најчешће позајмљиван филм).

```
CREATE VIEW Gledanost(SifF, BrPozajmica)
AS SELECT SifF, COUNT(SifP)
FROM Pozajmica
GROUIP BY SifF;

CREATE VIEW ClanPoz(SifC, Suma)
AS
SELECT P.SifC, SUM(F.Duzina)
FROM FILM F, Pozajmica P
WHERE P.SifF = F.SifF
GROUP BY P.SIfC;

SELECT C.SifC, C.Ime, CP.Suma
FROM Caln C, ClanPioz CP
WHERE C.SifC = CP.SIfC
AND C.SifC IN ( SELECT P.SifC
                FROM Gledanost G, Pozajmica P
                WHERE P.SIfF = G.SifF
                AND G.BrPozajmica =( SELECT MAX(BrPozajmica)
                                     FROM Gledanost)
                                   );
```

8. Саставити исказ релационог рачуна домена који даје називе филмове који су садржани на једној или више касета а нису никад позајмљивани.

```
{<Naziv>|∃SifF, Duzina, Ocena(<SifF, Naziv, Duzina, Ocena>∈Film∧
∧∃SifK(<SifK, SifF>∈Sadrzi) ∧
∧¬∃SifP, SifK1, SifC, Datum(<SifP, SifK1, SifF, SifC, Datum>∈Pozajmica))}
```

9. Саставити SQL скрипт који даје шифре и имена чланова који су позајмљивали све филмове које је позајмљивао члан са шифром ’MB01’.

```
SELECT SifC, Ime
FROM Clan C
WHERE NOT EXISTS ((SELECT SifF
                   FROM Pozajmica
                   WHERE SifC=’MB01’)
                   EXCEPT
                  (SELECT SifF
                   FROM Pozajmica
                   WHERE SifC=C.SifC))
AND C.SifC <> ’MB01’;
```

10. Саставити исказе релационе алгебре који дају шифре и називе филмова који су садржани на једној или више касета а нису позајмљивани.

$$\pi_{SifF}(Sadrzi) \rightarrow t_1(SifF)$$
$$\pi_{SifF}(Pozajmica) \rightarrow t_2(SifF)$$
$$t_1 - t_2 \rightarrow t_3(SifF)$$
$$\pi_{SifF,Naziv}\left(t_3 \underset{(SifF)}{\infty} Film\right) \rightarrow resenje(SifF, Naziv)$$

11. Саставити SQL скрипт који брише из евиденције информације о оним члановима који имају минимално просечно трајање позајмица.

```
CREATE VIEW Prosek(SifC, Prosek)
AS
SELECT SifC, AVG (Dana)
FROM Pozajmica
GROUP BY SIfC;

CREATE VIEW Minimalni(SifC)
AS
SELECT SifC
FROM Prosek
WHERE Prosek = (SELECT MIN (Prosek)
                FROM Prosek);

DELETE FROM Clan
WHERE SifC IN (SELECT SifC FROM Minimalni);
```

12. Саставити исказ релационог рачуна домена који даје шифре касета које су позајмљиване са бар два различита филма на себи.

```
{<SifK>|∃Duzina, SF(<SifK,Duzina,SF>∈Kaseta ∧
∧∃SP1, SC1, SF1, Dana1(<SP1, SC1, SF1, SifK, Dana1>∈Pozajmica
∧(∃SP2, SC2, SF2, Dana2(<SP2, SC2, SF2, SifK, Dana2>∈Pozajmica
∧SF1≠SF2))))}
```

13. Саставити SQL скрипт који даје парове шифара чланова који су гледали тачно исте филмове.

```
SELECT C1.SifC, C2.SifC
FROM Clan C1, Clan C2
WHERE (NOT  EXSISTS ( SELECT P1.SifF
                      FROM Pozajmica P1
                      WHERE P1.SifC = C1.SifC
                      AND P1.SifF NOT IN (SELECT P2.SiF
                                          FROM Pozajmica P2
                                    WHERE P2.SifC= C2.SIfC
                                          )
                     )
       )
AND (NOT EXSIST ( SELECT P1.SifF
                  FROM Pozajmica P1
                  WHERE P1.SifC = C2.SifC
                  AND P1.SifF NOT IN ( SELECT P2.SifF
                                       FROM Pozajmica P2
                                       WHERE P2.Sif=C1.SifC
                                      )
                )
    )
AND C1.SifC < C2.SifC;
```

14. Саставити SQL скрипт којим се за 10% повећавају цене оних филмова чија је оцена бар за 20% већа од просечне оцене филмова максималне дужине.

```
UPDATE Film
SET Cena = 1.1*Cena
WHERE Ocena >= 1.2*(SELECT AVG(Ocena)
                    FROM Film
                    WHERE Duzina= (SELECT MAX(Duzina)FROM Film)
);
```

15. Саставити SQL скрипт који за сваки жанр даје његову шифру, назив и број филмова тог жанра који су били позајмљивани.

```
CREATE VIEW ZanrFilm(SifZ, Naziv, BrFilmova)
AS
SELECT  Z.SifZ,Z.Naziv, COUNT(DISTINCT P.SifF)
FROM POZAJMICA P, FILM F, ZANR Z
WHERE P.SifF= F.SifF
AND F.SifZ= Z.SifZ
GROUP BY Z.SifZ,Z.Naziv

SELECT SifZ, Naziv, BrFilmova
FROM ZanrFilm
UNION
SELECT SifZ, Naziv, 0
FROM ZANR
WHERE SifZ NOT IN (SELECT SifZ  FROM ZanrFilm);
```

16. Саставити исказе релационе алгебре који дају шифре и имена чланова који су позајмили бар један од филмова који имају најмању оцену међу филмовима свог жанра.

$$\pi_{F1.SifF}\left(Film\ F1 \underset{\substack{F1.SifZ=F2.SifZ \\ F1.Ocena>F2.Ocena}}{\infty} Film\ F2\right) \rightarrow t_1(SifF)$$

$$\pi_{SifF}(Film) - t_1 \rightarrow t_2(SifF)$$

$$\pi_{SifC,\mathrm{Im}e}\left(Clan \underset{*}{\infty} \pi_{sifC}\left(Pozajmica \underset{*}{\infty} t_2\right)\right) \rightarrow reenje(SifC, IME)$$

17. Саставити SQL скрипт којим се вредност колоне Попуст поставља на вредност 20 оним члановима који су имали највећи број позајмица филмова одређеног (произвољног) жанра.

```
CREATE ZanrClan(SifZ,SifC,BrPozajmica)
AS
SELECT  F.SifZ, P.SifC, COUNT(P.SifP)
FROM POZAJMICA P, FILM F
WHERE P.SifF=F.SifF
GROUP BY F.SifZ, P.SifC;

CREATE VIEW MaxClan(SifC)
AS
SELECT  Z.SifC
FROM ZanrClan Z
WHERE Z.BrPozajmica=(      SELECT MAX(BrPozajmica)
                      FROM ZanrClan Z2
                      WHERE Z2.SifZ = Z.SifZ
                     );
UPDATE CLAN
SET Popust = 20
WHERE SifC IN (SELECT SifC FROM MaxClan);
```

18. Саставити SQL скрипт који враћа десет најчешће позајмљиваних филмова и за сваки од њих назив, оцену, број позајмица тог филма као и укупан број позајмица свих филмова који припадају истом жанру као и тај филм.

```
CREATE VIEW BrojPozajmicaF(SifF, SifZ, Naziv, Ocena, BrPozajmicaF)
AS
SELECT F.SifF, F.SifZ, F.Naziv, F.Ocena, COUNT(SifP)
FROM Pozajmica P, Film F
WHERE P.SifF = F.SifF
GROUP BY F.SifF, F.SifZ, F.Naziv, F.Ocena;

CREATE VIEW BrojPozajmicaZ(SifZ, BrPozajmicaZ)
AS
SELECT  SifZ, SUM(BrPozajmicaF)
FROM BrojPozajmicaF
GROUP BY SifZ;

SELECT PF.Naziv, PF.Ocena, PF.BrojPozajmicaF, PZ.BrojPozajmicaZ
FROM BrojPozajmicaF PF, BrojPozajmicaZ PZ
WHERE (PF.SifZ = PZ.SifZ)
AND (10 > ( SELECT COUNT(*)
            FROM BrojPozajmicaF PF2
            WHERE PF.BrPozajmicaF < PF2.BrojPozajmicaF
           )
     );
```

Биографија аутора

Милош Цветановић, магистар електротехничких наука, рођен 10.08.1978. године у Врању република Србија од оца Миодрага и мајке Душанке Цветановић. Основну и средњу школу завршио је у Врању као један од најбољих ученика, и при том учествовао на многобројним такмичења из природних наука.

Електротехнички факултет у Београду уписао 1997. године, и дипломирао у предвиђеном року као један од најбољих студената у класи са просечном оценом 9.10 током студија и оценом 10 на дипломском. Јула 2006. године магистрирао је на Електротехничком факултету смер Архитектура и организација рачунарских система и мрежа са тезом "Методологија интеграције софтверских система базирана на принципима реверзног инжењерства" код ментора проф. др Мирослава Бојовића и проф. др Вељка Милутиновића.

Од јуна 2003. ради на Електротехничком факултету у Београду на месту асистента из предмета Програмирање 1, Програмирање 2, Базе података 1, Базе података 2, Информациони системи 1, Информациони системи 2, Софтверски алати база података. Од октобра 2008. године је ангажован и као саветник управника Рачунског центра Електротехничког факултета.

Област његовог научног истраживања обухватају базе података, информационе системе, архитектуру и организацију рачунара, реверзно инжењерство.

Аутор је или коаутор два рада у међународним часописима са impact фактором који су на SCI листи од којих је један у директној вези са дисертацијом, једног рада у часопису, осам радова са међународних конференција и девет радова на националним конференцијама. Учествовао је на више међународних ФП7 пројекта, као и на више пројекта финансираних од Министарства за науку Републике Србије. Аутор је једног признатог проналаска под називом „Систем и поступак за обраду порука и генерисање мултимедијалног садржаја управљан даљински“.

Прилог 1.

Изјава о ауторству

Потписани-а Милош Цветановић

број индекса _____

Изјављујем

да је докторска дисертација под насловом

СИСТЕМ ЗА ИНТЕРАКТИВНУ ПРОВЕРУ СЛИЧНОСТИ КОНЦЕПТУАЛНИХ И ЛОГИЧКИХ МОДЕЛА РЕЛАЦИОНИХ БАЗА ПОДАТАКА

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

Потпис докторанда

У Београду, 27.04.2012.

Miloš Cvetanović

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Милош Цветановић

Број индекса _____

Студијски програм _____

Наслов рада Систем за интерактивну проверу сличности концептуалних и логичких модела релационих база података

Ментор др Мирослав Бојовић, ванредни професор, ЕТФ, УБ

Потписани/а Милош Цветановић

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања 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 докторанда

У Београду, 27.04.2017.

Milos Cvetanovic

Прилог 3.

Изјава о коришћењу

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

СИСТЕМ ЗА ИНТЕРАКТИВНУ ПРОВЕРУ СЛИЧНОСТИ КОНЦЕПТУАЛНИХ И ЛОГИЧКИХ МОДЕЛА РЕЛАЦИОНИХ БАЗА ПОДАТАКА

која је моје ауторско дело.

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

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

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, 27.04.2012.

Милетановић

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 прераде. 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 прераде. 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.