УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

Спецификација и валидација ограничења у XML моделу података

ДОКТОРСКА ДИСЕРТАЦИЈА

Кандидат

мр Јована Видаковић

Ментор

др Иван Луковић, ред. проф.

Нови Сад, 2014.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:		
Идентификациони број, **ИБР**:		
Тип документације, **ТД**:		Монографска документација
Тип записа, **ТЗ**:		Текстуални штампани материјал
Врста рада, **ВР**:		Докторска дисертација
Аутор, **АУ**:		мр Јована Видаковић
Ментор, **МН**:		др Иван Луковић, редовни професор
Наслов рада, **НР**:		Спецификација и валидација ограничења у XML моделу података
Језик публикације, **ЈП**:		српски
Језик извода, **ЈИ**:		српски
Земља публиковања, **ЗП**:		Србија
Уже географско подручје, **УГП**:		АП Војводина
Година, **ГО**:		2015.
Издавач, **ИЗ**:		Факултет техничких наука
Место и адреса, **МА**:		Трг Доситеја Обрадовића 6, 21000 Нови Сад
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/159/93/0/10/0/1
Научна област, **НО**:		Електротехничко и рачунарско инжењерство
Научна дисциплина, **НД**:		Примењене рачунарске науке и информатика
Предметна одредница/Кључне речи, **ПО**:		XML модел података, типови ограничења у XML моделу података, XML базе података
УДК		
Чува се, **ЧУ**:		Библиотека Факултета техничких наука, Трг Доситеја Обрадовића 6, 21000 Нови Сад
Важна напомена, **ВН**:		
Извод, **ИЗ**:		Циљ истраживања реализованих у овом раду, био је да се формално опишу типови ограничења у XML моделу података, по угледу на типове ограничења у релационом моделу података. У складу са постављеним циљем, урађена је класификација типова ограничења у XML моделу података, њихова формална спецификација и имплементација у репрезентативним XML СУБП.
Датум прихватања теме, **ДП**:		
Датум одбране, **ДО**:		
Чланови комисије, **КО**:	Председник:	др Милош Рацковић, ред. проф.
	Члан:	др Зоран Марјановић, ред. проф.
	Члан:	др Соња Ристић, ван. проф.
	Члан:	др Бранко Милосављевић, ред. проф. — Потпис ментора
	Члан, ментор:	др Иван Луковић, ред. проф.

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monograph documentation	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Ph.D. thesis	
Author, **AU**:		Jovana Vidaković, M.Sc.	
Mentor, **MN**:		Ivan Luković, Ph.D. Full Professor	
Title, **TI**:		Specification and Validation of Constraints in XML Data Model	
Language of text, **LT**:		Serbian	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Serbia	
Locality of publication, **LP**:		AP Vojvodina	
Publication year, **PY**:		2015	
Publisher, **PB**:		Faculty of Technical Sciences	
Publication place, **PP**:		Trg Dositeja Obradovića 6, 21000 Novi Sad	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/159/93/0/10/0/1	
Scientific field, **SF**:		Software Engineering	
Scientific discipline, **SD**:		Applied computer science	
Subject/Key words, **S/KW**:		XML data model, constraint types in XML data model, XML database management system	
UC			
Holding data, **HD**:		Library of Faculty of Technical Sciences, Tgr Dositeja Oradovića 4, 21000 Novi Sad	
Note, **N**:			
Abstract, **AB**:		The goal of the research conducted in this thesis was to formally describe the types of the constraints in the XML data model, according to the types of the constraints in the relational data model. In accordance with the set goal, the types of the constraints in the XML data model were classified, formally specified, and implemented in the representative XML DBMS.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Miloš Racković, Ph.D. Full Professor	
	Member:	Zoran Marjanović, Ph.D. Full Professor	
	Member:	Sonja Ristić, Ph.D. Associate Professor	
	Member:	Branko Milosavljević, Ph.D. Full Professor	Menthor's sign
	Member, Mentor:	Ivan Luković, Ph.D. Full Professor	

Садржај

1 Увод

Модерни информациони системи моделирају велик број правила пословања, која постоје у реалним системима. Постоје различити, мање или више формализовани приступи обезбеђивању поштовања правила пословања. Нека од правила пословања изражавају се путем различитих врста ограничења над подацима смештеним у базу података и могу бити имплементирана у оквиру шеме базе података. Постоје и она правила пословања која не резултују у ограничењима шеме базе података, већ у обавезама или условима спровођења операција над подацима. Оваква правила такође могу бити пројектована и имплементирана на нивоу шеме базе података. Предмет истраживања у овој дисертацији су правила пословања која могу бити исказана путем ограничења над подацима, као и њихова презентација и имплементација у оквиру шеме базе података. Ограничења над подацима у бази података информационог система омогућују поштовање правила пословања из реалног система. Осим тога, контролом ограничења обезбеђује се конзистентност података у бази података информационог система. Аутоматизовано поштовање оваквих правила убрзава рад са системом, смањује број грешака, олакшава унос података и смањује цену одржавања информационог система. Због тога, веома је битно дефинисати оквир за презентовање и имплементацију ограничења у систему базе података.

Ограничења се изражавају одређеном нотацијом приликом пројектовања базе података, а уграђују се у систем базе података приликом њене реализације. Ограничења представљају неопходан део сваке дефиниције шеме базе података. Она се користе приликом формирања и ажурирања базе података да би се база података одржавала у конзистентном стању, односно да би садржај базе података, у сваком тренутку, био у сагласности са свим дефинисаном ограничењима, тј. условима интегритета.

Релациони системи за управљање базом података су и даље највише заступљена технологија за реализацију базе података. Дефиниције и имплементације ограничења су стандардни део сваке релационе базе података. Последњих година, појављују се XML базе података, где се подаци чувају у XML датотекама, и омогућују смештај, ажурирање и претрагу структурираних докумената. XML базе података имају предност над релационим, када се смештају и претражују структурирани документи. Ограничења над подацима у XML базама података нису у довољној мери истражена, као што је то случај са релационим базама података. Постојећи стандарди омогућују опис и имплементацију елементарних ограничења, док је рад на многим ограничењима комплексније природе и даље предмет истраживања. Ова дисертација бави се описом и имплементацијом ограничења у XML базама података.

За опис структурираних докумената користи се XML нотација. Extensible Markup Language (XML) представља фамилију језика за спецификацију докумената, који је убрзо по свом увођењу 1998. године стекао велику популарност. XML је лако разумљив и лак за учење, стандардизован и отворен за даља проширивања. XML служи за меморисање, публиковање и размену информација, и омогућава странама у комуникацији да саставе свој језик са тачно дефинисаном семантиком, синтаксом и правилима структурирања. Коришћење XML језика за формирање докумената помаже избегавању вођења докумената у приватном формату. За XML документе се каже да су самообјашњавајући, јер садрже и податке и мета податке. Сваки део документа је ограничен почетним и завршним тагом, који носе информацију о семантици тог садржаја. То чини XML документе лако читљивим. Осим размене XML докумената, потребно је дефинисати начин смештаја, претраживања и ажурирања XML докумената. Да би све стране у комуникацији користиле исту врсту XML докумената, потребно је дефинисати семантику, синтаксу и правила структурирања документа.

Интензивна размена података путем XML докумената створила је потребу да се они складиште, претражују и ажурирају. То је довело до развоја система за управљање базама података (СУБП) заснованим на XML технологији. Они се користе у изградњи XML система база података. XML поседује језике и концепте који се могу користити у улози модела података са својом структуралном, интегритетном и операцијском компонентом, чиме су обезбеђени сви потребни услови за изражавање шеме XML базе података на формалан начин. Модел података који ће бити коришћен у овој докторској дисертацији, назваћемо XML модел података.

XML модел података је у интензивном развоју. Подржава поступке за дефинисање ограничења са придруженом семантиком. Већ структурална компонента XML модела података садржи неке уграђене концепте за дефинисање ограничења над XML структурама.

Мета језик DTD подржавао је дефинисање ограничења само путем тзв. ID/IDREF механизама. Овај језик има ограничене могућности за дефинисање типова података који се користе у XML документима. У основи, он омогућује само основну дефиницију структуре XML документа (који елементи, којим редоследом, и која бројност елемента), уз елементарно покривање референце једног елемента ка другом. Тако, атрибут типа IDREF представља везу ка другом елементу, који поседује атрибут типа ID.

Развој система за управљање базама података заснованих на XML језицима захтева постојање свеобухватно дефинисаног модела података, што доводи до потребе дефинисања концепата и поступака за потпунију спецификацију и контролу ограничења, а који су усвојени у оквиру мета језика XML Schema. Овај мета језик прецизно дефинише структуру и типове података у XML документима. Подржан је велик број типова података, који постоје у већини програмских језика који се данас користе, као и регуларни изрази, када је потребно дефинисати строжија правила за валидацију садржаја елемената или њихових атрибута. Подржани су и комплексни, често коришћени типови података, попут датумских типова, типова URI-ова, QName (Namespace Qualified Name) и сл. XML Schema подржава концепт наслеђивања, којим се омогућује профињење дефиниције неког постојећег типа. XML Schema подржава везе између елемената путем типова атрибута ID/IDREF. Као и у релационом моделу података, потреба пројектовања шема у XML Schema језику, чије појаве ће бити погодне за ажурирање, поново је иницирала истраживања у области функционалних зависности и дефинисања нормалних форми, али сада на нивоу XML модела података.

Може се оценити да је проблем представљања и имплементације ограничења до данас најбоље разрешен у релационом моделу података. Осим што је у питању модел података који је данас већ најдуже у употреби, разлог томе је и да је он развијен на формалним основама теорије скупова и релација, као и предикатског рачуна првог реда, с јаком оријентацијом ка обезбеђењу логичке и физичке независности програма од података. Објектно-оријентисани модел података је настао после релационог модела и поседује формални механизам за представљање ограничења. Када су у питању системи база података, објектно-оријентисани модел података није заживео на имплементационом нивоу, већ се преводи у релациони модел података. Не постоји стандардизован начин имплементације ограничења у објектно-оријентисаном моделу, већ постоје истраживања код којих се та ограничења преводе или у програмски код или у ограничења у релационом моделу. Поставља се питање да ли је могуће искористити знање из релационог или објектно-оријентисаног модела података за моделирање и имплементацију ограничења у XML базама података. Такође, отворено је питање употребљивости постојећег знања из других модела података, као и како то знање употребити приликом спецификације ограничења у XML моделу података. Ова дисертација ће покушати да да допринос у тражењу одговора на таква отворена питања.

Из претходно описаног, произилази прва хипотеза која је постављена у овој дисертацији:

***Хипотеза 1**: Типизација ограничења у XML моделу података може се дефинисати по угледу на типизацију ограничења у релационом моделу података.*

Чињеница је да у пракси постоји значајан број ограничења која се често јављају и могу се типизовати, али у доступној литератури није пронађено пуно информација о томе да је тако нешто заиста и урађено. Постоји више приступа у дефинисању и имплементацији ограничења у XML базама података, са различитим формализмима, који обухватају само мали број ограничења. Дакле, проблем је како на формалан начин описати што више типова ограничења и сама ограничења, као и да ли је могуће искористити постојеће знање из других модела података. Од посебног интереса су ограничења у релационом моделу података, пошто су у тој области урађена најобимнија истраживања. У XML моделу података постоји затечено стање које се може унапредити. Тако на пример, неке врсте ограничења уопште нису ни типизиране, или постоје типови ограничења за које не постоје одговарајући концепти. Истраживања која су спроведена у оквиру ове дисертације, односе се на један приступ спецификацији и имплементацији ограничења у XML моделу података, који се ослања и на постојеће дефиниције ограничења из других модела података. Структурална компонента је обогаћена новим концептима, уведени су нови типови ограничења, као и правила за њихово записивање и интерпретацију. Предложено решење је допринело остварењу циља да се пројектантске спецификације, превасходно ограничења шеме базе података, а посредно и апликативне софтверске подршке, могу аутоматски трансформисати у извршни облик.

Основни циљеви ове дисертације су да се у XML моделу података развију формализми, уведу концепти и предложи нотација путем које ће бити могуће дефинисати ограничења различитих типова. Направљена је типизација великог броја ограничења која се појављују у пракси. Такође, искоришћено је знање о свим постојећим формализмима из релационог модела података, али и из других модела података, које је такође искоришћено за спецификацију система за проверу ограничења у XML моделу података. Развијени су нотација и алгоритми за проверу важења ограничења, који су омогућили дефинисање основе за имплементацију система за проверу ограничења.

Очекивани резултати истраживања су типизација великог броја ограничења која се појављују у пракси и дефинисање основа за имплементацију система за проверу ограничења и имплементација изабраних метода у оквиру две репрезентативне XML базе података. Један од резултата рада на дисертацији је предлог дефиниције доменски оријентисаног језика за моделирање ограничења. Овај језик омогућава дефинисање свих ограничења наведених у овој дисертацији, што олакшава рад пројектанту који не мора да води рачуна о детаљима имплементације самих ограничења. На основу овог језика имплементиран је генератор кода који реализује проверу ограничења. Захваљујући генератору, олакшан је посао и програмера, који не мора да реализује проверу ограничења дефинисаних доменским језиком, већ користи код генерисан генератором.

Из овога произилази друга хипотеза ове дисертације:

***Хипотеза 2**: Генератор кода може да формира ограничења у имплементираној шеми базе података на основу формалног описа ограничења.*

Типови ограничења који су дефинисани у дисертацији су и валидирани у две репрезентативне XML базе података, што је и основа за формирање треће хипотеза ове дисертације:

***Хипотеза 3**: Валидација ограничења дефинисаних у XML моделу података може се извршити у XML бази података.*

Текст ове дисертације организован је у осам поглавља: Увод, пет тематских поглавља, Закључак и Литература.

Након овог, уводног поглавља, у којем су изложене хипотезе истраживања, у другом поглављу презентовани су истраживање и анализа тренутног стања у области. Поглавље почиње прегледом модела података који су имали утицаја на ову дисертацију. Дата је анализа постојећих приступа дефинисању ограничења. Описују се језици DTD и XML Schema, стандарди и приступи у дефинисању ограничења у XML документима, а затим даје се преглед истраживања којима се они проширују. Истраживања иду у више праваца. Једна група радова бави се дефинисањем XML нормалних форми и функционалних зависности за XML документе. Други правац заснива се на проширењу дефиниције кључа, увођењем појма релативног и апсолутног кључа. Једна група истраживања уводи систем заснован на правилима (rule-based system) за додатну валидацију XML докумената, као што је, на пример, Shematron језик. У овом поглављу приказане су и анализиране позитивне и негативне карактеристике постојећих предлога и решења. Дат је и преглед постојећих типова ограничења у релационом моделу података, који су, уз поменуте позитивне и негативне карактеристике постојећих решења, искоришћени као полазна основа за моделовање додатних типова ограничења у XML моделу података.

Треће поглавље бави се израдом таксономије типова ограничења у XML моделу података, њиховом интерпретацијом и дефинисањем формализама за њихово записивање. Развијена је посебна нотација за приказ ових ограничења, и представљена у форми једног доменски оријентисаног језика.

У четвртом поглављу дата је спецификација система за валидацију ограничења. Дефиниција типова ограничења у овом систему користи нотацију дефинисану у претходном поглављу. Спецификација је реализована путем UML дијаграма, а имплементација самог система извршена је у програмском језику Java.

Пето поглавље приказује имплементацију система за валидацију ограничења која је демонстрирана кроз репрезентативне студије случаја. Ограничења су дефинисана на нивоу XML Schema докумената, а за имплементацију коришћене су XQuery функције, као и тригери.

У закључку је презентована дискусија постављених хипотеза, изведена из добијених резултата. Истакнута су и значајна ограничења приступа, предложеног у овој докторској дисертацији. Дискутовани су могући правци даљег истраживања.

На крају текста рада, дат је списак коришћене литературе.

Као резултат рада на овој дисертацији, добијен је систем за валидацију XML докумената, проширен новим дефиницијама ограничења. Добијена софтверска подршка могла би бити уграђена у изабрани CASE алат, тако да буде обезбеђено аутоматско генерисање ограничења и трансформација пројектантских спецификација у извршни облик, спреман за оперативно коришћење. Ово решење може се користити и у фази пројектовања и у фази имплементације информационог система. Пројектанти добијају могућност да на формалан начин изразе и она ограничења, која не постоје у конвенционалним XML СУБП. Програмери не морају да имплементирају провере ограничења у коду, већ могу да препусте проверу генерисаном коду, који је настао на основу спецификације ограничења.

Научни допринос ове докторске дисертације односи се на моделирање додатних ограничења у XML бази података, која нису подржана постојећим стандардима. За спецификацију ограничења, развијен је један доменски оријентисан језик. На основу овог језика, реализован је генератор кода, који генерише код који обезбеђује проверу ограничења у XML бази података. У циљу квалитетније подршке процеса пројектовања XML шеме базе података,

развијен је механизам за опис додатних ограничења, а на нивоу имплементације шеме базе података, обезбеђено је софтверско решење које на основу формираних спецификација обавља проверу валидности података, исказаних у форми XML докумената.

2 Преглед тренутног стања у области истраживања

У овом поглављу дати су преглед и анализа актуелног стања у области истраживања. Поглавље почиње прегледом модела података који су имали утицаја на ову дисертацију. Дата је анализа постојећих приступа дефинисању ограничења у релационом и XML моделу података. Осим описа језика DTD и XML Schema дат је опис упитних језика XPath и XQuery.

У овом поглављу приказане су и анализиране позитивне и негативне карактеристике постојећих предлога и решења из доступне литературе. Дат је и преглед постојећих типова ограничења у релационом моделу података, који су, уз поменуте позитивне и негативне карактеристике постојећих решења, искоришћени као полазна основа за моделовање додатних типова ограничења у XML моделу података. Дат је преглед истраживања ограничења, дефиниције кључева и валидације.

На крају поглавља, дат је предлог структуре XML документа који ће бити коришћен у овој дисертацији.

2.1 Модели података

Модел података је математичка апстракција која се користи за пројектовање модела реалног система [Mog96]. Модел података представља формалну основу, на основу које је могуће пројектовати шему базе података и реализовати базу података неког информационог система. Он треба да садржи концепте и језике, на основу којих се исказују информације о:

- дефиницији података, односно о њиховој статичкој структури и особинама,
- уграђеним ограничењима,
- динамичким особинама, односно скупом принципа којима је описана манипулација подацима.

Статичке особине садрже информације о структури будуће базе података и релативно су независне од времена. Оне се исказују структуралном компонентом модела података.

Ограничења дефинишу правила пословања и понашања у реалном систему, као и дозвољене вредности и односе између података у реалном систему. Ова ограничења постају услови интегритета у бази података који обезбеђују да подаци о реалном систему у бази података буду усаглашени са ограничењима која постоје у реалном систему. Ова правила исказују се интегритетном компонентом модела података.

Динамичке особине одсликавају промене у реалном систему. Оне су описане у моделу помоћу операција које доводе базу података у сагласност са актуелним подацима у реалном систему. Ове операције исказане су путем операцијске компоненте модела података. Структуре над скупом операција представљају програме. Ти програми мењају базу података сагласно променама у реалном систему.

Током последњих педесетак година развијен је већи број различитих модела података. Неки од њих представљали су само покушај или „успутну станицу“ у развоју других модела података, а други су заживели и оставили траг како у теорији, тако и у пракси база података. Неки од модела који су се користили, или се и данас користе у пракси, су:

- мрежни модел података,
- хијерархијски модел података,
- релациони модел података,
- модел типова ентитета и повезника,
- објектно-оријентисани модел података и
- XML модел података.

Модели података који су од значаја за истраживање у овој тези су хијерархијски и релациони модел података. Неке њихове основне карактеристике биће приказане у наставку овог поглавља. XML модел података је у центру истраживања у овој докторској дисертацији и њему је у овом поглављу посвећена највећа пажња.

2.1.1 Хијерархијски модел података

Хијерархијски и мрежни модели података појавили су се у другој половини шездесетих година двадесетог века [Sil05]. Убрзо су у свакодневну употребу уведени системи за управљање базама података засновани на хијерархијском и мрежном моделу података. Међутим, ти системи за управљање базама података нису довели до очекиваног пораста продуктивности ни програмера ни корисника, јер логички и физички аспекти базе података нису били довољно раздвојени, структуре података су биле комплексне, а коришћени су процедурални језици у улози операционе компоненте. Хијерархијски модел података представљао је основу за IBM систем за управљање базама података IMS (Information Management System).

Хијерархијска база података састоји се од слогова који су повезани једни са другима путем веза. Сваки слог је колекција поља (атрибута), од којих свако садржи само једну вредност. Веза повезује тачно два слога. Хијерархијска база је колекција стабала са кореном и формира шуму.

Садржај једног слога може бити поновљен више пута на различитим позицијама у бази података. Понављање се може десити у истом стаблу или у различитим стаблима једне базе података. Због тога се може јавити неконзистентност података приликом операција ажурирања, а долази и до непотребног трошења меморијског простора додељеног бази.

У стаблу са кореном не постоје цикличне структуре. Везе између слогова могу бити само с кардиналитетима „један-према-један“ (1:1) или „један-према-више“ (1:M). Тип слога на највишем нивоу хијерархије назива се коренски слог. Сваки слог дете (подређени слог) има само једног родитеља (надређени слог) са којим је повезан. Слог родитељ може имати више деце са којима је повезан. Свако претраживање овакве базе података мора кренути од корена.

Сви атрибути неког типа слога представљени су као листа унутар самог типа слога. Тип слога у хијерархијској бази података еквивалентан је табели у релационом моделу података. Сваки слог у оквиру неког типа слога представљен је као врста, а атрибут као колона.

Сваки тип слога има атрибут који је кључ. Он се користи за приступ подацима сваког слога. Слог је директно повезан са својим родитељем или дететом, док је таква веза у релационом моделу података представљена страним кључевима.

Дефинисање веза са кардиналитетом N:M доводи до стварања редундантних података. Уводе се показивачи на слог родитељ који је у логичкој вези са конкретним слогом. Овакви показивачи подсећају на IDREF механизам у XML моделу података.

Хијерархијски системи за управљање базама података подржавали су изградњу великих база података. Добра особина хијерархијских база података је у томе што се сваком слогу може приступити брзо, и такође извршити промену над њим брзо, због структуре типа стабла која то

омогућава, и због тога што су везе између слогова дефинисане унапред. Мана овакве базе података је у томе што сваки слог дете у структури може имати само један слог родитељ, а везе између слогова деце нису дозвољене, чак и ако та веза семантички постоји.

XML модел података у извесној мери подсећа на хијерархијски модела података. Елементи у XML документима организовани су као стабла, баш као и слогови у хијерархијским базама података. Атрибути елемената у XML документима описују елементе, као што и атрибути описују слогове у хијерархијском моделу.

Хијерархијски модел података није се задржао у употреби, јер га је заменио релациони модел података, који је исправио недостатке хијерархијског модела. Они су се огледали у: недовољном раздвајању логичких од физичких аспеката базе података, комплексности структура података и коришћењу процедуралног и навигационог језика [Mog96].

2.1.2 Релациони модел података

Релациони модел података [Dat03] је настао седамдесетих година прошлог века и основни разлози увођења били су увођење јасне разлике између логичких и физичких аспеката базе података, структурална једноставност и постојање декларативног језика за дефинисање и коришћење базе података.

У дотадашњим моделима података информације о физичкој структури су биле уграђене у саме програме. Уколико би се физичка структура базе података променила, морали би се мењати и програми на које та измена утиче. У релационом моделу података највећа организациона јединица података је н-арна релација, која садржи скуп н-торки. Опис релације на нивоу апстракције обележја назива се шема релације и представља именовани пар N(R, C), где је N назив шеме релације, R скуп обележја, а C је скуп услова интегритета релације који указују на особине торки релације. Релациону шему базе података чини скуп шема релација с придруженом скупом ограничења. То је именовани пар N(S, I), где је S скуп шема релација, а I скуп међурелационих ограничења.

На нивоу екстензије, концепте релационог модела података представљају: домен обележја, торка, релација и појава базе података, док на нивоу интензије, основне концепте представљају: обележје, шема релације и шема базе података.

Као и у другим моделима података, ограничења служе за одржавање базе података у сагласности са реалним системом. У релационом моделу података ограничења се могу поделити на ограничења торки, релациона и међурелациона ограничења. Провера важења ограничења торке се спроводи за сваку торку релације посебно. За проверу важења релационог ограничења морају се посматрати међусобни односи више торки једне релације. Међурелациона ограничења проверавају усаглашеност базе података као појаве, са ограничењима која су дефинисана у шеми базе података.

Операцијска компонента релационог модела података служи за опис динамичких карактеристика реалног система и њом се дефинише језик за изражавање упита над базом података и језик помоћу ког се врши ажурирање базе података како би она била усаглашена са стањем реалног система. Операцијска компонента садржи упитни језик, језик за ажурирање података и језик за дефиницију података. Стандардни језик релационих система за управљање базама података је SQL (Structured Query Language).

Формализација типова ограничења у релационом моделу података је изузетно добро развијена. То знање може се искористити у овој докторској дисертацији за формализацију типова ограничења у XML моделу података.

2.1.3 XML модел података

XML модел података није нов модел података, у употреби је више од једне деценије. Иако није нов, он није довољно истражен, а посебно у поређењу са релационим моделом података. Ограничења у XML моделу података такође спадају у мање истражене научне области. Због свега наведено, за област истраживања у овој дисертацији одабрана су ограничења у XML моделу података.

XML модел података, као и други модели података, има структуралну, интегритетну и операцијску компоненту. Структурална компонента XML модела података заснива се на концепту XML документа, на нивоу екстензије. На нивоу интензије користе се XML Schema документ или DTD документ. У овој дисертацији детаљно је проучена интегритетна компонента, и у наставку су описана ограничења која важе у XML моделу података. Шема базе података представља модел типа стабла. Могу се пронаћи сличности са хијерархијским моделом података, што је описано у одељку 2.1.1. Листови тог стабла су прости елементи, а нетерминални чворови су сложени елементи.

XML модел података је специфичан у односу на друге моделе података по томе што XML екстензија може да постоји и без XML интензије. То је последица самообјашњавајуће природе XML екстензије. Међутим, ако не постоји шема по којој је формиран XML документ, не може се говорити о XML бази података, у правом смислу појма база података.

2.1.4 XML документ

XML документ је основни концепт XML језика [XML]. XML се користи за меморисање, објављивање и размену информација. Он омогућава странама у комуникацији да дефинишу синтаксу, семантику и правила структуирања [Vid06], на такав начин да су информације разумљиве и за људе и за рачунарске програме.

XML се формира помоћу два основна концепта: елемента и атрибута. Остале компоненте које може да садржи XML документ су:

- XML декларације и инструкције обраде,
- коментари,
- спољни ентитети,
- CDATA секције и
- простори имена.

XML документи се препознају на основу XML декларације. Декларација је инструкција обраде и налази се на почетку XML документа. У документу могу постојати и друге инструкције обраде.

Елемент је главни градивни елемент XML документа. Сваки елемент мора имати почетни и завршни таг, а садржај елемента може бити текст или неки други елементи. Сваки XML документ има тачно један елемент који садржи све остале елементе и он се назива коренским елементом. Елементи морају бити правилно угњеждени.

У XML документу постоје два основна типа елемената:

- прости елементи, који садрже само текст, тј. атомичне вредности,
- сложени елементи, који садрже бар два концепта од следећа три: елемент, атрибут, текст.

Сложени елемент може садржати поделементе, чиме се прави хијерархијска структура. Атрибутима се детаљније описују елементи и приказује се њихова семантика. За опис података који се чувају у елементима користе се атрибути, а њихова примена не мора увек одговарати употреби атрибута у релационом моделу података.

Атрибут је дефинисан својим називом, а има и своју вредност. Атрибут се може појавити у декларацији XML документа, инструкцији обраде или почетном тагу елемента. Назив и вредности представљају низове карактера који су одвојени знаком једнакости. Вредност атрибута је наведена између знакова навода. Атрибути се наводе у оквиру почетног тага елемента, после назива елемента. Сваки елемент одређеног типа може имати само један атрибут са датим називом, али се атрибут са истим називом може јавити у оквиру почетног тага елемената више различитих типова.

Коришћење атрибута уместо елемената даје компактнију структуру XML документа. Међутим, атрибут може садржати само текст, док елемент може садржати и друге елементе, подређени елементи се могу јавити више пута у оквиру надређеног. Атрибути су погодни за просте податке, који немају подструктуре и појављују се највише једном.

XML документ мора да буде добро формиран. То значи да мора да задовољи одређена синтаксна правила:

- мора да почне XML декларацијом,
- мора да има један јединствени коренски елемент,
- сваки почетни таг '<' мора да има одговарајући затварајући таг '>',
- називи елемената су *case-sensitive,*
- сви елементи морају бити затворени,
- сви елементи морају бити правилно угњеждени,
- вредности атрибута се стављају под знаке навода и
- за специјалне карактере се користе ентитети.

XML документ не захтева постојање шеме, не мора да постоји предефинисани скуп елемената који се могу појавити у документу. У том случају аутору документа оставља се слобода приликом дефинисања елемената, али се и онемогућава аутоматска интерпретација значења података у документу.

Уколико се XML документ формира према DTD-у или XML шеми, и задовољава сва правила која су у тим документима дата, поред тога што је добро формиран, он је и валидан. Структура елемената у XML документу мора да прати спецификацију структуре која је дефинисана у посебном документу. Језици за спецификацију структуре XML документа су XML Document Type Definition (DTD) и XML Schema.

Пример једног XML документа дат је у Листинг 1. Овај документ описује процес извођења наставе на факултету. Сваки наставник може да предаје више предмета, а предмет може да буде предаван од стране више наставника. Сваки наставник има тачно једно звање. У оквиру коренског елемента fakultet може се наћи више елемената nastavnik, predmet, predaje, zvanje. У оквиру елемента predaje вредност пара атрибута (id_nastavnika, id_predmeta) може се појавити само једном. XML Schema документ који описује структуру овог XML документа описан је у наредном делу.

```
<?xml version="1.0" encoding="UTF-8"?>
<fakultet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="fakultet.xsd">
 <predmet id_predmeta="BP1">Baze podataka 1</predmet>
 <nastavnik id_nastavnika="N1">
  <ime>Ana</ime>
```

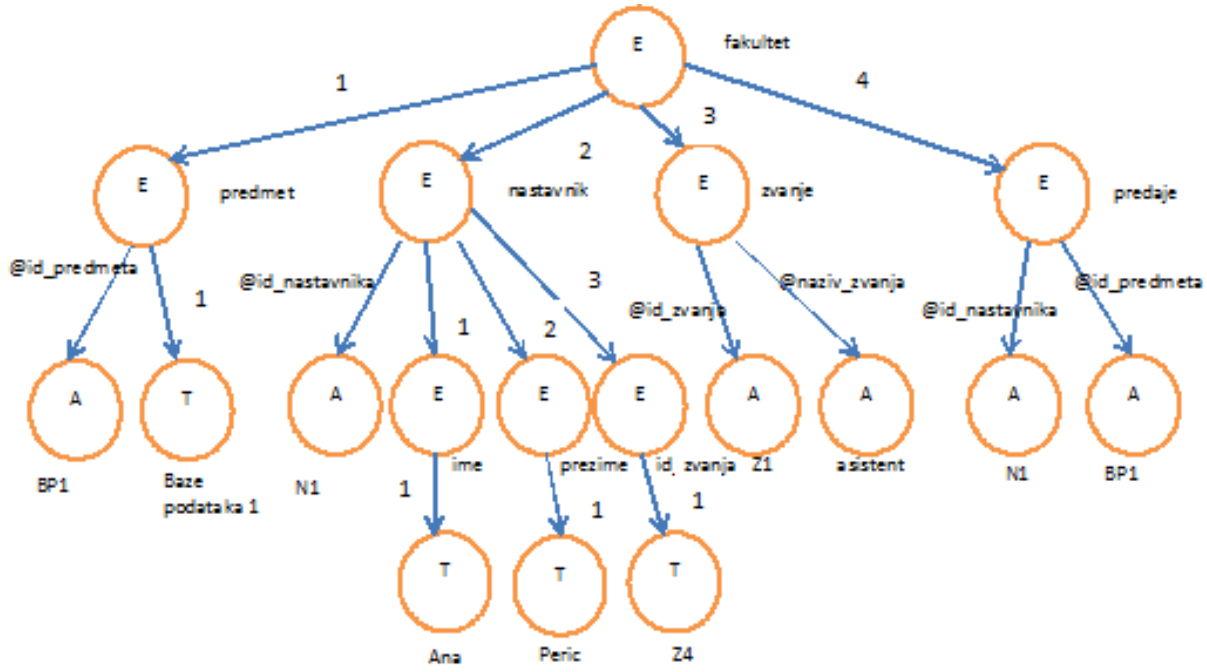
```
  <prezime>Peric</prezime>
  <id_zvanja>Z4</id_zvanja>
 </nastavnik>
 <nastavnik id_nastavnika="N2">
    <ime>Maja</ime>
  <prezime>Peric</prezime>
  <id_zvanja>Z1</id_zvanja>
 </nastavnik>
    <nastavnik id_nastavnika="N3">
    <ime>Milan</ime>
  <prezime>Savic</prezime>
  <id_zvanja>Z4</id_zvanja>
 </nastavnik>
 <zvanje id_zvanja="Z1" naziv_zvanja="asistent"/>
 <zvanje id_zvanja="Z2" naziv_zvanja="docent"/>
 <zvanje id_zvanja="Z3" naziv_zvanja="vanredni profesor"/>
 <zvanje id_zvanja="Z4" naziv_zvanja="redovni profesor"/>
 <predaje id_nastavnika="N1" id_predmeta="BP1"/>
</fakultet>
```

Листинг 1 Пример једног XML документа

Добро формиран XML документ представља структуру стабла. Елемент који садржи све остале елементе, је корен стабла. Елементи, атрибути и текст у садржају неког елемента *e* представљају чворове, подређене том елементу *e*. Текстуалне вредности простих елемената и атрибута налазе се у листовима стабла. Чворови у XML документу који је представљен као стабло морају бити једнозначно идентификовани. То се, пре свега, односи на елементе, јер се текстуална вредност неког елемента дефинише као низ повезаних текстуалних вредности свих његових подређених простих елемената. Чињеница да један XML документ може да садржи више концепата са истим називом и типом, који имају и исту текстуалну вредност, доводи до потребе увођења поступка за једнозначну идентификацију концепата у оквиру XML документа. Неки од таквих поступака су: поступак заснован на Document Object Model (DOM) [DOM] и поступак заснован на додели једнозначних идентификатора чворовима - елементима.

DOM омогућава лако додавање, брисање и измену XML садржаја. Према DOM моделу, XML документ је структура стабла над скупом чворова различитог типа. Неки од типова, који су овде од значаја, су: чворови елементи, чворови атрибути и текстуални чворови. Текстуални чворови (T) немају име, али садрже текст. Чворови атрибути (A) имају име и садрже текст. Чворови елементи (E) имају име и подређене чворове. Чворови атрибути и текстуални чворови су терминални. DOM дефинише поступак приступања директно подређеним чворовима неког чвора елемента. Директно подређени чворови елементи и текстуални чворови налазе се у једнодимензионалном низу. Индекс сваког таквог подређеног чвора је одређен његовим релативним положајем у оквиру надређеног чвора. Назив атрибута мора бити јединствен у оквиру неког елемента, па се он користи за јединствену идентификацију атрибута у оквиру елемента. Редослед навођења атрибута у оквиру надређеног елемента је неважан. Испред назива атрибута наводи се префикс '@'. Индекси текстуалних чворова и чворова елемената, као и називи атрибута, представљају ознаке ивице графа, који репрезентује XML документ. Ознаке ивица графа једнозначно идентификују директно подређене чворове у оквиру једног надређеног чвора. Коришћењем ознака ивица графа, за сваки чвор *n* може се направити пут од корена, који јединствено идентификује чвор *n*. Тај пут се назива адресом чвора. Нека је (i_1#...#i_n) адреса чвора *n*. Тада је ознака i_1 ознака ивице која полази од тог корена, '#' симбол за повезивање путева, а i_n ознака ивице, која повезује чвор *n* са његовим директно надређеним чвором.

На Слици 1 приказано је DOM стабло XML документа из Листинг 1. Адреса (2#1#1) једнозначно идентификује текстуални чвор са вредношћу Ана.



Слика 1 Стабло XML документа

Други поступак идентификације делова XML документа је заснован на додели једнозначних идентификатора елементима од врха стабла ка дну, с лева на десно, по нивоима. Испред атрибута се наводи префикс '@', а идентификација се врши посредно, помоћу елемента који га садржи. Сваком чвору атрибуту и чвору простом елементу додељује се њихова текстуална вредност. Текстуални чворови се експлицитно не представљају. Идентификатори чворова носе информацију о уређењу подређених чворова у оквиру једног надређеног чвора, па није потребно наводити индексе ивица графа које повезују чворове елементе.

2.1.5 DTD

Document Type Definition (DTD) [DTD] је део основног XML стандарда. Описује формат XML документа и дефинише називе елемената и атрибута, као и тип садржаја елемената и атрибута. DTD има мало типова података, а синтакса му је другачија од XML-а, па захтева постојање посебних процесора докумената. XML документи морају да поседују елементе баш у оном редоследу у ком су дефинисани у DTD документу. Од ограничења могуће је дефинисати само кључ и страни кључ помоћу ID и IDREF атрибута. Међутим, они су више усмерени на дефинисање јединствених вредности атрибута који је типа ID, и референцирање на неке од постојећих вредности у XML документу, помоћу IDREF атрибута. Ограничења која су везана за складиштење података у базу података није могуће дефинисати.

Пример једног DTD документа је дат у Листинг 2. XML документ из Листинг 1 је валидан у односу на овај DTD.

Коренски елемент fakultet има поделементе: predmet, nastavnik, zvanje и predaje. Сваки од ових поделемената се мора појавити бар једном у оквиру свог надређеног елемента fakultet, што је обележено знаком +. Елемент predmet садржи текст, а садржи и атрибут id_predmeta, који је типа ID. То значи да ће атрибут id_predmeta бити кључ елемента predmet, а његова употреба је обавезна, што је означено кључном речи REQUIRED. Елемент nastavnik има поделементе ime, prezime, и id_zvanja. Елементи ime и prezime садрже текст. Елемент id_zvanja је типа IDREF, што значи да има референцу на елемент или атрибут са истим називом, а који је

типа ID. У овом примеру, он се односу на атрибут id_zvanja елемента zvanje. Атрибут елемента nastavnik је id_nastavnika, који је и кључ овог елемента. Елемент zvanje има два атрибута: id_zvanja је кључ овог елемента, а naziv_zvanja задржи тест и његова употреба је обавезна. Елемент predaje такође има два атрибута, id_nastavnika и id_predmeta. Оба ова атрибута су референце на одговарајуће атрибуте из других елемената.

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT fakultet (predmet+, nastavnik+, zvanje+, predaje+)>
<!ELEMENT predmet (#PCDATA)>
<!ATTLIST predmet id_predmeta ID #REQUIRED>
<!ELEMENT nastavnik (ime, prezime, id_zvanja)>
<!ATTLIST nastavnik id_nastavnika ID #REQUIRED>
<!ELEMENT ime (#PCDATA)>
<!ELEMENT prezime (#PCDATA)>
<!ELEMENT id_zvanja (IDREF)>
<!ELEMENT zvanje (EMPTY)>
<!ATTLIST zvanje id_zvanja ID #REQUIRED naziv_zvanja CDATA #REQUIRED>
<!ELEMENT predaje (EMPTY)>
<!ATTLIST predaje id_nastavnika IDREF #REQUIRED id_predmeta IDREF #REQUIRED>
```

Листинг 2 DTD документ

Провера валидности XML документа врши се у односу на DTD документ тако што вредност атрибута standalone треба да буде „no“, а назив DTD документа наводи се у заглављу.

DTD има веома скромне могућности за дефинисање типова података. Постоји само 10 типова података. Синтакса је другачија од XML-а, па захтева постојање посебних процесора докумената. ID/IDREF механизам, којим се у DTD-у дефинишу јединствене вредности и референце, има ограничене могућности. Атрибут који је типа ID има јединствене вредности у оквиру целог XML документа, међу свим атрибутима који су типа ID. То значи да два различита елемента, на пример, predmet и nastavnik, не могу имати исте вредности атрибута id_predmeta и id_nastavnika, редом, иако су то различити елементи. Такође, атрибут који је типа IDREF, односи се на неки атрибут типа ID у документу, али није јасно на који. У елементу predaje атрибут id_nastavnika је типа IDREF, али вредност тог атрибута мора само да постоји у документу и може да буде вредност било код атрибута типа ID.

2.1.6 XML Schema

XML Schema [XMLSch] је скуп предефинисаних декларација XML елемената, чија семантика и синтакса су прописани W3C стандардом. Тај скуп декларација је, поред DTD-а, други мета језик за дефинисање XML шема. Уведен је како би се отклонили недостаци уочени код DTD-а. Све ознаке у XML шеми имају префикс xs: што указује да припадају одређеном простору имена, где је дефинисана њихова синтакса и семантика. XML шема поседује механизме за декларацију компонената, структуре XML документа и ограничења. Коришћењем тих механизама формирају се конкретне XML шеме. XML документи који су формирани у складу са неком конкретном XML шемом називају се појавама те шеме. XML шема се декларише коришћењем исте синтаксе која се користи у XML документима и нуди механизме за извођење нових типова података ограничавањем или проширивањем постојећих и дефинисање много ефикаснијих ограничења.

Помоћу XML Schema језика дефинишу се елементи који се појављују у XML документу, атрибути, постојање, редослед и број подређених елемената, да ли је елемент празан или може да садржи текст, типови података за елементе и атрибуте, као и предефинисане и фиксне вредности за елементе и атрибуте. Могуће је дефинисати патерне (формат података), фасете (рестрикције над подацима), као и типове података изведене из других стандардних

типова података. Дефинисање разних типова података омогућава лакши опис садржаја документа, лакшу валидацију података, конверзију података различитих типова.

Уколико XML документ постоји независно од интензије, довољно је да буде добро формиран. Међутим, уколико је XML документ формиран према неком XML Schema документу, он мора бити и валидан, то јест, мора да задовољи сва правила која су у шеми наведена. У XML документу је потребно референцирати шему чија је инстанца тај документ.

XML schema (шема) је заснована на моделу типа стабла. Основни концепти у шеми су елементи и атрибути, а из релационог модела података преузети су кључеви, референце и идентификатори.

XML schema дефинише елементе које треба да садржи XML документ: опис шеме и XML просторе имена, анотације, елементе шеме, типове елемената и интегритет шеме.

Коренски елемент сваке XML шеме је елемент <schema>. Он може садржати атрибуте, а неки од њих су:

- targetNamespace, који указује на то из ког простора имена потичу елементи у конкретној шеми, како би се спречила колизија елемената са истим именом; један тип документа може садржати елементе из више простора имена, а један елемент може бити коришћен у више шема и
- elementFormDefault, који указује на то да елементи декларисани у шеми који се користе у XML документу, који је инстанца конкретне шеме, морају бити квалификовани простором имена.

Анотација је спецификација коментара и описа XML Schema документа. На пример, xs:documentation је спецификација коментара намењеног људима, а атрибут xml:lang дефинише језик који се користи у XML документу. xs:appinfo специфицира коментаре намењене апликацијама.

Елементи шеме служе за дефинисање елемената који се користе у XML документу. Могу бити прости (simple) и сложени (complex). Прост елемент може садржати само текст и не може да садржи друге елементе или атрибуте. Међутим, текст може бити различитих типова, као нпр. boolean, string, или date, или неког кориснички дефинисаног типа. Такође, могуће је додати фасет (рестрикцију) типу података да би се ограничио садржај, или захтевати да податак одговара неком дефинисаном патерну (формату података). XML Schema има много уграђених типова података, на пример: xs:string, xs:decimal, xs:integer, xs:boolean, xs:date и xs:time. Прости елементи могу имати предефинисану вредност, која се аутоматски додељује елементу када му није придружена ниједна конкретна вредност, или фиксну вредност, која се, такође, аутоматски додаје елементу и тада није могуће додати неку другу вредност.

Прости елементи не могу имати атрибуте. Ако елемент садржи атрибут, мора бити сложеног типа (complex type). Атрибути такође могу имати предефинисане или фиксне вредности и могу бити неког од уграђених типова података, или кориснички дефинисаних. Атрибути су по дефиницији опционални, а по потреби се могу декларисати као обавезни (required), што се наводи као вредност атрибута use у елементу xs:attribute.

Рестрикције (facet) се користе за дефинисање дозвољених вредности за XML елементе или атрибуте. Рестрикције се могу дефинисати над вредностима помоћу елемената xs:minInclusive и xs:maxInclusive, над скуповима вредности помоћу елемента xs:enumeration. Рестрикција низа вредности се постиже помоћу дефинисања патерна xs:pattern, а рестрикција дужине вредности у елементу помоћу xs:length, xs:minLength, xs:maxLength.

Сложени елементи садрже друге елементе и/или атрибуте. Могу бити празни елементи, елементи који садрже само друге елементе, елементи који садрже само текст и елементи који садрже и текст и друге елементе. Сваки од ових елемената такође може садржати и атрибуте. Да би био дефинисан начин навођења елемената у XML документу, користе се индикатори:

- индикатори редоследа:
  - all – елементи деца могу се појавити у било ком редоследу тачно једном,
  - choice – само један елемент дете се може појавити,
  - sequence – елементи деца морају се појавити у тачно утврђеном редоследу,
- индикатори појављивања:
  - maxOccurs – максималан број појављивања елемента,
  - minOccurs – минималан број појављивања елемента,
- индикатори групе:
  - group – користи се за дефинисање групе елемената који се касније могу референцирати,
  - attributeGroup – дефинисање групе атрибута који се могу референцирати.

Пример шеме, на основу које је формиран XML документ из Листинг 1, дат је у наставку. У оквиру коренског елемента може се појавити више елемената *predmet*, *nastavnik*, *zvanje*, *predaje*. Елементи *predmet*, *nastavnik* и *zvanje* имају атрибуте који су типа ID и који представљају кључеве. Елемент *predaje* има два атрибута *id_nastavnika* и *id_predmeta*, који представљају стране кључеве преузете из елемената *nastavnik* и *predmet*.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="fakultet">
  <xs:annotation>
   <xs:documentation>Comment describing your root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="predmet">
     <xs:complexType>
      <xs:simpleContent>
       <xs:extension base="xs:string">
        <xs:attribute name="id_predmeta" type="xs:ID" use="required"/>
       </xs:extension>
      </xs:simpleContent>
     </xs:complexType>
    </xs:element>
    <xs:element name="nastavnik">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="ime"/>
       <xs:element name="prezime"/>
       <xs:element name="id_zvanja" type="xs:IDREF"/>
      </xs:sequence>
      <xs:attribute name="id_nastavnika" type="xs:ID" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="zvanje" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="id_zvanja" type="xs:ID" use="required"/>
      <xs:attribute name="naziv_zvanja" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:enumeration value="asistent"/>
         <xs:enumeration value="docent"/>
         <xs:enumeration value="vanredni profesor"/>
         <xs:enumeration value="redovni profesor"/>
```

```
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
     </xs:complexType>
    </xs:element>
    <xs:element name="predaje">
     <xs:complexType>
      <xs:attribute name="id_nastavnika"/>
      <xs:attribute name="id_predmeta"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:key name="zvanje_PK">
   <xs:selector xpath="zvanje"/>
   <xs:field xpath="@id_zvanja"/>
  </xs:key>
  <xs:key name="nastavnik_PK">
   <xs:selector xpath="nastavnik"/>
   <xs:field xpath="@id_nastavnika"/>
  </xs:key>
  <xs:key name="predmet_PK">
   <xs:selector xpath="predmet"/>
   <xs:field xpath="@id_predmeta"/>
  </xs:key>
  <xs:keyref name="nastavnik_zvanje_FK" refer="zvanje_PK">
   <xs:selector xpath="zvanje"/>
   <xs:field xpath="@id_zvanja"/>
  </xs:keyref>
  <xs:keyref name="predaje_nastavnik_FK" refer="nastavnik_PK">
   <xs:selector xpath="nastavnik"/>
   <xs:field xpath="@id_nastavnika"/>
  </xs:keyref>
  <xs:keyref name="predaje_predmet_FK" refer="predmet_PK">
   <xs:selector xpath="predaje"/>
   <xs:field xpath="@id_predmeta"/>
  </xs:keyref>
 </xs:element>
</xs:schema>
```

Листинг 3 XML Schema

2.1.7 Schematron

Осим DTD-а и XML Schema-е, за валидацију XML докумената користи се и Schematron [Schem, Rob03]. Schematron је језик заснован на правилима (*rule-based*) уз помоћ којих се дефинишу обрасци у XML документу. Он се у основи разликује од осталих шема језика по томе што није заснован на граматици, него на претраживању образаца у парсираном документу. Овакав приступ дозвољава употребу различитих врста структура, поготово оних које су неуобичајене и тешке за представљање у језицима заснованим на граматикама. Schematron омогућава развој две врсте шема:

- елементе *assert* који омогућавају потврду да ли документ задовољава одређени образац и
- елементе *report* који представљају негацију логике елемента *assert*.

Schematron је заснован на две једноставне акције: прво пронађи контексни чвор у документу, обично елемент, на основу услова у XPath путањи, а затим провери да ли су неки други XPath изрази тачни за сваки од пронађених чворова.

Може се користити заједно са многим језицима који су засновани на граматикама, а који служе за проверу валидације структуре: DTD, XML Schema, и других. Schematron је део ISO

стандарда, који је дизајниран да омогући да више језика који служе за валидацију XML докумената раде заједно.

2.2 Ограничења у моделима података

Ограничења чине веома важан део сваке базе података. Сваки модел података треба да има интегритетну компоненту у којој ће бити описани типови ограничења који важе у том моделу података, а системи за управљање базама података треба да подржавају механизме за контролу тих типова ограничења. Релациони модел података има добру математичко – формалну основу. У овом одељку дат је приказ типова ограничења у релационом моделу података, јер су у њему детаљно развијени многи типови ограничења, а дозвољено је и креирање корисничких типова ограничења. Предлог типова ограничења у XML моделу података биће дат на основу типова ограничења у релационом моделу података.

2.2.1 Органичења у релационом моделу података

У релационом моделу података развијена је формална нотација за исказивање различитих типова ограничења [Mog96, Luk03, Luk07]. Сва ограничења требало би да буду реализована на нивоу сервера базе података путем механизама које садржи систем за управљање базом података. Тада је контрола ограничења централна и не може је заобићи ниједан корисник или програм. Корисници нису ни свесни постојања неког ограничења све док се оно не наруши. У случају покушаја нарушавања ограничења применом неке операције ажурирања, систем за управљање базом података доводи базу података у конзистентно стање, или изазива грешку, прекида операцију и прослеђује корисничком програму поруку о грешци, да би програм обрадио ту поруку у проследио је кориснику.

Да би ограничење шеме базе података било имплементирано, потребно је дефинисати параметре ограничења, а то су: дефинисање ограничења датог типа, дефинисање операција које могу довести до нарушавања ограничења, за сваку операцију дефинисање акција које се спроводе у случају нарушавања ограничења, а да би се база података одржала у конзистентном стању. Систем за управљање базом података аутоматски проверава важење ограничења, сагласно дефинисаним правилима.

Свако ограничење везано је за неко обележје и неке шеме релација шеме базе података. Операције које могу довести до нарушавања дефинисаног ограничења су упис нове торке у релацију, брисање постојеће торке из релације и модификација вредности обележја у торци. Уколико дође до покушаја нарушавања неког ограничења, покрећу се акције очувања конзистентности базе података. Оне се везују за сваку операцију која може нарушити ограничење и могу бити активне, пасивне или комбиноване, то јест, оне које у неким ситуацијама могу бити активне, а у неким пасивне. Активне акције обезбеђују спровођење даљих операција ажурирања над базом података, које ће је одржати у конзистентном стању, док пасивне спречавају операцију која може да наруши ограничење.

Ограничења шеме базе података се имплементирају помоћу механизама система за управљање базом података. Механизму се придружују сви параметри ограничења: дефиниција, критичне операције које могу нарушити ограничење, а за сваку критичну операцију, акција очувања конзистентности базе података.

Систем за управљање базом података покреће механизам контроле важења ограничења аутоматски или након извођења критичне операције за ограничење. Предности овакве реализације интегритетне компоненте шеме базе података је у томе што се врши аутоматска контрола имплементираних ограничења на нивоу система за управљање базом података,

Такође, обезбеђена је конзистентност базе података у сваком тренутку. Постоји стандардизација начина за имплементацију ограничења. Како се ограничења имплементирају и контролишу једном, на нивоу система за управљање базом података, нема функционалне потребе да се ограничења реализују у програму, иако постоји потреба да се контрола неких ограничења понови и у програмима, у циљу обезбеђења боље удобности рада корисника.

Недостатак оваквог приступа реализацији интегритетне компоненте шеме базе података је у томе што имплементација шеме базе података у великој мери зависи од произвођача, типа и верзије система за управљање базом података, јер не подржавају сви системи за управљање базом података у истој мери постојеће стандарде, који, са друге стране, не покривају увек све неопходне детаље. Ипак остаје потреба да се нека ограничења реализују унутар програма да би сам програм био погоднији за употребу, иако би таква контрола ограничења била двострука.

2.2.1.1 Спецификација типа ограничења

Карактеристике типа ограничења у моделу података су следеће [Luk07a]:

- област дефинисаности, која представља тип логичке структуре обележја над којом се дефинише ограничење,
- област интерпретације, која представља тип логичке структуре података над којом се ограничење имплементира,
- формализам за записивање,
- правило за интерпретацију,
- скуп критичних операција над базом података, које могу довести до нарушавања ограничења датог типа и
- скуп могућих акција којима се обезбеђује очување валидности базе података при покушају нарушавања ограничења датог типа, а који се дефинишу за сваку критичну операцију.

Тип ограничења је задат на следећи начин:

TipO(T(t), TOd, TOi, TFz, TPi)

где су:

- TipO – ознака типа ограничења,
- T(t) – дефиниција типа логичке структуре обележја с укљученим критичним операцијама и могућим акцијама,
- TOd – спецификација области дефинисаности,
- TOi – спецификација области интерпретације,
- TFz – дефиниција формуле за записивање,
- TPi – дефиниција правила за интерпретацију.

2.2.1.2 Типизација ограничења

У релационом моделу података, уочени су следећи типови ограничења које пројектант користи при специфицирању конкретних ограничења шеме базе података:

- ограничење домена,
- ограничење вредности обележја,
- ограничење торке,

- проширено ограничење торке,
- ограничење ентитета,
- ограничење јединствене вредности обележја,
- међурелациона ограничења:
    - зависност садржавања,
    - проширена зависност садржавања,
    - селективна проширена зависност садржавања,
    - ограничење референцијалног интегритета:
        - проширено,
        - селективно,
        - селективно и проширено,
    - ограничење инверзног референцијалног интегритета:
        - проширено,
        - селективно,
        - селективно и проширено.

Сваки тип ограничења карактеришу особине:

- параметризована формула (синтакса израза) за спецификацију ограничења датог типа,
- правило за интерпретацију ограничења датог типа,
- скуп операција над базом података које могу довести до нарушавања конзистентности базе података с обзиром на ограничење датог типа и
- скуп активности за обезбеђење конзистентности базе података, које се спроводе само у случају да операција над базом података нарушава ограничење; за сваку такву операцију, дефинише се по једна активност.

У наставку су приказани конкретни типови ограничења у релационом моделу података, према управо наведеним особинама.

2.2.1.3 Ограничење домена

Ограничење домена је ванрелационо ограничење којим се ограничавају могуће вредности обележја. За сваки домен мора бити дефинисан тип податка, који имају вредности обележја из тог домена. Такође, мора бити задата максимална дужина и додатни услов, који вредности из домена морају да задовоље. Синтакса формуле за записивање ограничења датог типа дата је изразом:

$$TFz = id(D) = (Tip, Dužina, Uslov),$$

где је id(D) ограничење домена са називом D. Оваква синтакса користи се за ограничења домена, креирана путем правила наслеђивања.

Постоје примитивни и кориснички дефинисани домени, који представљају проширење примитивног домена. Ограничење домена се интерпретира за сваку могућу вредност d, на следећи начин:

$$id(D)(d) = (Tip, Dužina, Uslov)(d) = Tip(d) \wedge Dužina(d) \wedge Uslov(d) .$$

За сваку вредност из домена потребно је проверити да ли је дефинисаног типа податка, да ли задовољава ограничење максималне дужине и додатни услов.

2.2.1.4 Ограничење вредности обележја

Ограничење вредности обележја служи да обележју придружи ограничење домена и да специфицира да ли је за обележје дозвољено задавање недостајућих вредности. Ово ограничење се користи за једну шему релације и једно обележје. Синтакса формуле за записивање ограничења вредности обележја дата је изразом:

$$TFz = \tau(N, A) = (id(D), NullSpec), NullSpec \in \{Null, NotNull\},$$

где је $\tau(N, A)$ ограничење вредности обележја, id(D) је ограничење домена дефинисано у одељку 2.2.1.3, а NullSpec означава једну од две вредности Null или NotNull.

Ограничење вредности обележја служи за спецификацију дозволе или забране нула вредности за свако обележје у шеми релације. Интерпретира се за сваку могућу вредност d, на следећи начин:

$$\tau(N, A)(d) = (id(D), NullSpec)(d) = id(D)(d) \wedge NullSpec(d).$$

За сваку могућу вредност обележја потребно је проверити да ли припада дефинисаном домену у да ли задовољава спецификацију недостајуће вредности. Акције које могу нарушити ово ограничење су унос и модификација. Поред ограничења домена, обележје мора задовољити и услов дозволе или забране нула вредности.

2.2.1.5 Ограничење торке

Ограничење торке је ограничење вредности обележја на нивоу једне торке, у појави над шемом релације. Ограничењем торке задаје се логички услов ограничења који торке у релацији треба да задовоље. Ограничење торке се дефинише за тачно једну шему релације над скупом више обележја. Синтакса формуле за записивање ограничења вредности обележја дата је изразом:

$$TFz = \tau(N) = (\{\tau(N, A) | A \in R\}, Con(N)),$$

где је $\tau(N)$ ограничење торке, N је назив шеме релације, R је ознака скупа свих обележја шеме релације N, а Con(N) је логички услов ограничења торке.

Интерпретира се за сваку појединачну торку t из релације r(N), на следећи начин:

$$t \in r(N),\ \tau(N)(t) = (\{\tau(N, A) | A \in R\}, Con(N))(t) = (\forall A \in R)(\tau(N, A)(t[A])) \wedge Con(N)(t).$$

Приликом уноса или модификације торке у релацији, свако обележје из те торке мора задовољити ограничење торке.

2.2.1.6 Проширено ограничење торке

Проширено ограничење торке даје логички услов који повезује скупове обележја више шема релација. Ово ограничење се користи за више шема релација над скупом више обележја сваке од њих. Логички услов проширеног ограничења торке дефинише се над скупом обележја која припадају унији свих скупова обележја посматраних шема релација. Синтакса формуле за записивање проширеног ограничења торке дата је изразом:

$$TFz = \tau_{ex}(N_1 \rhd\lhd ... \rhd\lhd N_m) = Con(N_1 \rhd\lhd ... \rhd\lhd N_m),$$

где је $\tau_{ex}(N_1 \rhd\lhd ... \rhd\lhd N_m)$ проширено ограничење торке, N_1,..., N_m су називи шема релација које су повезане овим ограничењем, а $Con(N_1 \rhd\lhd ... \rhd\lhd N_m)$ логички услов проширеног ограничења торке.

Интерпретира се за сваку појединачну торку t која припада природном споју свих релација, на следећи начин:

$$t \in r(N_1) \rhd\lhd ... \rhd\lhd r(N_m),\ \tau_{ex}(N_1 \rhd\lhd ... \rhd\lhd N_m)(t) = Con(N_1 \rhd\lhd ... \rhd\lhd N_m)(t).$$

Слично као и код обичног ограничења торке, и проширено ограничење торке мора бити задовољено приликом уноса или модификације торки у релацијама.

2.2.1.7 Ограничење кључа

Свака шема релације поседује бар један кључ. То је скуп обележја чија вредност јединствено идентификује сваку торку у релацији. Ограничење кључа се користи за тачно једну шему релације над скупом више обележја. Синтакса формуле за записивање ограничења кључа дата је изразом:

$$TFz = Key(N, X),\ X \subseteq R,$$

где је X кључ шеме релације N, са скупом обележја R. Ниједно обележје у кључу не сме имати нула вредност, што је записано на следећи начин:

$$(\forall A \in X)(Null(A) = \perp).$$

Ограничење кључа интерпретира се на следећи начин:

$$1: (\forall u, v \in r(N))(u[X]=v[X] \Rightarrow u=v)$$
$$2: (\forall X' \subset X)(\neg\ 1^0(X'))$$

Ниједно обележје у кључу не сме имати нула вредност, а вредност кључа је за сваку торку јединствена. Први услов указује на јединственост вредности кључа, у смислу да се свакој торци релације додељује друга вредност кључа и да се путем вредности кључа може извршити једнозначна идентификација торке. Други услов указује на минималност скупа обележја кључа. Кључ је минималан скуп обележја који јединствено одређује сваку торку у релацији r(N) и не постоји мањи скуп обележја за који важи исто.

2.2.1.8 Ограничење јединствене вредности обележја

Ограничење јединствене вредности обележја обезбеђује да задати скуп обележја мора да има јединствене вредности у релацији. Ограничење јединствености се дефинише за тачно једну шему релације над скупом више обележја, а синтакса формуле за записивање ограничења јединствене вредности обележја дата је изразом:

$$TFz = Unique(N, X),\ X \subseteq R,$$

где је X скуп обележја који треба да има јединствене вредности у релацији r(N).

Уколико у две торке неки скуп обележја нема нула вредност, онда вредности тог скупа обележја морају бити међусобно различите. Интерпретира се за целу релацију r(N), на следећи начин:

$$1: (\forall u, v \in r(N))((u[X] \neq \omega \wedge v[X] \neq \omega) \Rightarrow (u[X] = v[X] \Rightarrow u = v))$$

2: $(\forall X' \subset X)(\neg\ 1^0(X'))$

Први услов указује на јединственост вредности скупа обележја, ако вредност није нула вредност. Други услов указује на минималност скупа обележа над којим се дефинише ограничење јединствене вредности.

2.2.1.9 Ограничење референцијалног интегритета

Ограничење референцијалног интегритета представља основно и најчешће коришћено међурелационо ограничење. Путем овог ограничења повезују се подаци у две релације, како би они били у усаглашеном стању. Ограничење референцијалног интегритета је међурелационо ограничење који се дефинише за две шеме релација, на следећи начин:

$$TFz = N_i[X] \subseteq N_j[Y],$$

где је Y кључ шеме релације N_j, а X је страни кључ у шеми релације N_i.

Ово ограничење се интерпретира над целим релацијама $r(N_i)$ и $r(N_j)$, на следећи начин:

$$\pi_X(r(N_i)) \subseteq \pi_Y(r(N_j)).$$

Пројекција релације $r(N_i)$ на скуп обележја X, мора бити подскуп пројекције релације $r(N_j)$ на скуп обележја Y, који је примарни кључ шеме релације N_j. То значи да се у релацији $r(N_i)$, као вредности страног кључа могу појавити само постојеће вредности примарног кључа из релације $r(N_j)$.

2.2.1.10 Проширено ограничење референцијалног интегритета

Проширено ограничење референцијалног интегритета повезује више шема релација, код којих се не може директно дефинисати референцијални интегритет. Ово је међурелационо ограничење дефинисано између више шема релација, а синтакса формуле за записивање проширеног ограничења референцијалног интегритета дата је изразом:

$$TFz = (\triangleright\triangleleft N_{k=i1}{}^{im})[X] \subseteq (\triangleright\triangleleft N_{l=j1}{}^{jm})[Y],$$

где X представља низ обележја референцирајућих шема релација, а Y представља доменски компатибилни низ обележја референцираних шема релација.

Проширена зависност садржавања се интерпретира над спојевима релација, на следећи начин:

$$\pi_X(\triangleright\triangleleft_{k=i1}{}^{im}(r(N_k))) \subseteq \pi_Y(\triangleright\triangleleft_{l=j1}{}^{jm}(r(N_l))).$$

Пројекција природног споја релација $r(N_k)$ на низ обележја X, мора бити подскуп пројекције природног споја релација $r(N_l)$ на низ обележја Y, који је доменски компатибилни низ обележја низу обележја X.

2.2.1.11 Ограничење инверзног референцијалног интегритета

Ограничење инверзног референцијалног интегритета може да важи само ако важи и одговарајуће ограничење референцијалног интегритета. Ово ограничење повезује две шеме релација N_i и N_j. Уколико постоји ограничење референцијалног интегритета $N_i[X] \subseteq N_j[Y]$, а торке релације $r(N_j)$ егзистенцијално зависе од торки релације $r(N_i)$, може се дефинисати и ограничење инверзног референцијалног интегритета између шема релација N_i и N_j.

Ограничење инверзног референцијалног интегритета је међурелационо ограничење који се дефинише за две шеме релација, а синтакса формуле за записивање ограничења инверзног референцијалног интегритета дата је изразом:

$$TFz = N_j[Y] \subseteq N_i[X],$$

где је Y кључ шеме релације N_j.

Ово ограничење се интерпретира над целим релацијама $r(N_i)$ и $r(N_j)$, на следећи начин:

$$\pi_Y(r(N_j)) \subseteq \pi_X(r(N_i)).$$

Пројекција релације $r(N_j)$ на скуп обележја Y, мора бити подскуп пројекције релације $r(N_i)$ на скуп обележја X. То значи да за сваку Y вредност у релацији $r(N_j)$ мора постојати бар једна торка са истом X вредношћу у релацији $r(N_i)$.

2.2.2 Ограничења података у XML моделу података

Постојећи системи који подржавају XML модел података, подржавају имплементацију следећих типова ограничења: ограничење јединствености, ограничење примарног кључа и ограничење референцијалног интегритета. Они се могу специфицирати у XML Schema документу. Иако постоје радови који се баве ограничењима у XML моделу података, ниједан од предлога до сада није стандардизован.

2.2.2.1 Ограничење јединствености

Овим ограничењем се дефинише јединствена вредност елемента на нивоу XML документа. У XML Schema-и користи се елемент xs:unique. Његов поделемент xs:selector служи за дефинисање опсега важења ограничења, јер се наводи тип елемента на који се примењује ограничење. Поделемент xs:field одређује поље унутар изабраног типа елемента на које се примењује ограничење. Вредност тог поља ће бити јединствена.

2.2.2.2 Ограничење примарног кључа

У XML Schema-и користи се елемент xs:key који садржи поделементе xs:selector и xs:field. Они редом означавају који тип елемента, односно, које поље из тог елемента чини примарни кључ. Примарни кључ не може имати null вредност, то јест, вредност атрибута nullable је увек false.

2.2.2.3 Ограничење референцијалног интегритета

Користи се за референцирање на неки претходно дефинисани примарни кључ. Број и тип поља у референцираном у кључу мора одговарати броју и типу поља у кључу. За специфицирање ограничења референцијалног интегритета у XML Schema-и користи се елемент xs:keyref, чији атрибут refer има вредност примарног кључа који се референцира. Слично као и код претходна два ограничења, поделементи xs:selector и xs:field одређују редом, који тип елемента, односно које његово поље представљају страни кључ.

2.3 Операцијска компонента XML модела података

XML језици и технологије широко се користе за обезбеђење размене података. Због тога се јавила потреба да се уведе упитни језик, који ће омогућити постављање упита над подацима у XML формату. W3C је предложио језике XPath и XQuery, који су данас у широкој употреби.

Постоје и други упитни језици, али наведена два језика користи већина XML СУБП. Због тога су ова два језика одабрана за реализацију програмског кода који ће проверавати ограничења описана у овој дисертацији.

2.3.1 XPath

XPath [XPath] је упитни језик за адресирање подструктура у XML структурама, неопходно у циљу реализације операција приступања подацима или претраживања у XML документима. Користи се за селектовање чворова у XML документу, као и за израчунавање вредности на основу садржаја XML документа. Као и сам XML, дефинисан је од стране World Wide Web Consortium (W3C). Последња верзија XPath 3.1 издата је у децембру 2014. XPath користи изразе путање (path expressions) за приступање чворовима XML документа. XPath изрази могу да се користе и за специфицирање области дефинисаности ограничења идентитета. XPath има велики број уграђених функција које, између осталог, омогућавају рад са бројевима, стринговима и boolean вредностима.

Најважнија врста израза у XPath-у је путања локације (location path), која се састоји од низа корака. За декларисање XPath израза користе се две синтаксе: потпуна и скраћена. Путем обе синтаксе декларишу се апсолутне и релативне путање кроз XML документе. Апсолутна путања почиње знаком /, а релативна не. Путања се састоји од једног или више корака који су раздвојени знаком /. Сваки корак садржи осу, тест чвора и нула или више предиката. Израз се израчунава у односу на текући чвор. Оса одређује смер у ком се креће у односу на текући чвор. Тест и предикат се користе за избор чворова по оси која је одређена у изразу. Корак се записује на следећи начин: osa::test_čvora[predikat].

Према XPath синтакси, релативна путања кроз XML документ дефинише се путем израза облика: $korak_1/.../korak_n$. Израчунава се корак по корак, с лева на десно. Сваки корак се примењује на један чвор – контекстни чвор. Примена корака на неки чвор генерише низ резултантних чворова. Након тога, сваки чвор резултантног низа користи се као контекстни чвор за наредни корак.

Када се XPath изрази користе за одређивање области дефинисаности ограничења идентитета, XPath израз је део XML шеме и угњежден је у декларацију неког елемента, чије појаве представљају низ контекстних чворова за ту путању.

Оса одређује смер кретања у стаблу којим је представљен XML документ. Списак могућих оса, у пуном и скраћеном запису, дат је у следећој табели.

Потпуна синтакса	Скраћена синтакса	Значење
`ancestor`		Сви преци текућег чвора
`ancestor-or-self`		Сви преци или текући чвор
`attribute`	`@`	Сви атрибути текућег чвора
`child`		Сва деца текућег чвора
`descendant`		Сви потомци текућег чвора
`descendant-or-self`	`//`	Сви потомци или текући чвор
`following`		Све што се налази иза

		затварајућег тага текућег чвора
following-sibling		Сви рођаци после текућег чвора
namespace		
parent	..	Родитељ текућег чвора
preceding		Све што се налази пре отварајућег тага текућег чвора
preceding-sibling		Сви рођаци пре текућег чвора
self	.	Текући чвор

Табела 1 Осе у XPath изразима

Тест чвора је услов који треба да буде задовољен за сваки чвор у резултантном низу неког корака.

Предикати се користе за проналажење чвора који садржи задату вредност. Наводе се у угластим заградама. На пример, у изразу датом у Листинг 4, треба пронаћи елемент zvanje, који се налази испод елемента fakultet, а чија вредност атрибута naziv_zvanja је редовни професор.

```
//fakultet/zvanje[@naziv_zvanja = "redovni profesor"]
```

Листинг 4 XPath израз

2.3.2 XQuery

XQuery [XQuery] је упитни програмски језик који служи за постављање упита над структуираним и неструктуираним подацима, најчешће над XML документима. Може се рећи да је XQuery у XML базама података еквивалент SQL-у у релационим базама података. Развијен је од стране XQuery радне групе, у оквиру W3C. Прва верзија 1.0 настала је у јануару 2007. а последња верзија 3.0 постала је препорука од стране W3C у априлу 2014. Основна улога XQuery језика је манипулација подацима у XML документима, као и у било којој бази података која податке чува у облику XML-а. XQuery користи XPath изразе, за приступ одређеном делу XML документа над којим се реализује операција. XQuery и XPath користе исти модел података и подржавају исте операције и функције.

XQuery користи FLOWR изразе, где је FLWOR скраћеница настала од почетних слова кључних речи For, Let, Where, Order by, Return, као што је приказано у Листинг 5.

```
FOR <formiranje sekvencu čvorova>
LET <povezivanje promenljivih sa sekvencom čvorova>
WHERE <uslov po kom se biraju čvorovi>
ORDER BY <sortiranje čvorova>
RETURN <specifikacija rezultata upita>
```

Листинг 5 FLWOR изрази

Клаузула FOR може да се појави нула или више пута и променљивој додељује један по један елемент из повратне вредности израза путање. Клаузула LET може да се појави нула или више пута и додељује променљивој једну вредност до краја извршења упита. Клаузула WHERE може да се појави нула или један пут и дефинише додатни услов над селекцијом елемената. Клаузула ORDER BY може да се појави нула или један пут и дефинише начин исписа резултата израза. Клаузула RETURN мора да се појави тачно једном и дефинише изглед враћеног XML

документа. Променљиве дефинисане у клаузулама FOR и LET могу да се користе у остатку израза.

Листинг 6 приказује XQuery упит који из документа датог у Листинг 6 издваја све наставнике са звањем редовни професор.

```
for $y in doc("fakultet.xml")//fakultet/zvanje[@naziv_zvanja = "redovni profesor"]
for $x in doc("fakultet.xml")//fakultet/nastavnik[id_zvanja = $y/@id_zvanja]
return <res>{$x/ime, $x/prezime}</res>
```

Листинг 6 XQuery упит

Резултат извршавања овог упита приказан је Листинг 7.

```
<res>
  <ime>Ana</ime>
  <prezime>Peric</prezime>
</res>
<res>
  <ime>Milan</ime>
  <prezime>Savic</prezime>
</res>
```

Листинг 7 Резултат извршавања XQuery упита

2.4 Преглед стања у области који се односи на до сада предложена ограничења у XML моделу података

Када је XML постао широко употребљив у разним областима информационих технологија, дошло се до закључка да он није само опште прихваћено решење за размену података преко Интернета, већ да се може користити и у базама података. Разни правци истраживања у области XML-а развијани су у последњих петнаестак година.

У великом броју доступних радова [Bun01, Bun02, Bun03, Fan00, Fan01, Fan05, Sha08, Sha09] аутори се баве дефинисањем кључева, апсолутних и релативних, примарних и страних, као и ограничењем јединствене вредности. Такође, баве се нормалним формама и функционалним зависностима [Sch05, Liu03, Vin04, Vin07, Dav07]. У литератури се, међутим, не могу пронаћи радови о осталим типовима ограничења који се јављају у релационом моделу података, а могли би бити дефинисани и коришћени и у XML моделу података. У прегледаним радовима, ограничења се дефинишу над DTD документима, јер су они једноставнији од XML Schema докумената, иако XML Schema документи имају много богатију семантку. Радови [Hu05, Hu06] укључују и XML Schema-у у спецификацију ограничења.

Затим, неки аутори [Bon10, Jum08, Jum10, Kri03, Ata07] се баве проблемом складиштења XML докумената у релационе базе података. Долази до развоја XML Native база података [Sedna, eXist], које могу да складиште XML документе и да управљају њима. Развијени су упитни језици за XML базе података [Deu99].

Како је и у релационом моделу података, нека ограничења потребно имплементирати уз помоћ процедуралних механизама и тригера, прегледана је доступна литература и на тему тригера [Tat01, Bon01, Gri05]. Постоје групе аутора које су дале предлоге за дефинисање XML тригера, постоје системи за управљање XML базом података који подржавају тригере [Sedna], али већина представљених тригера односи се на правила пословања.

2.5 Кључеви. Апсолутни и релативни кључ

Развој ограничења за XML почео је дефинисањем неких типова ограничења за полуструктуиране документе [Bun01]. Полуструктуирани документи најчешће се описују као самоописујући и немају шему. XML може бити само добро формиран, и да не постоји шема по којој је направљен. Уведен је појам ограничења путање (path constraint) који је битан за дефинисање ограничења, како за полуструктуиране, тако и за структуиране документе. Оваквом врстом ограничења могуће је изразити унарне зависности садржавања, али није могуће дефинисати кључ.

Као што је већ описано, DTD даје једноставан ID/IDREF механизам за дефинисање кључева. Међутим, он је више базиран на референцама, а њиме се не могу прецизно изразити ограничења. У раду [Fan00] аутори се баве истраживањем ограничења за XML, тако што су проширили DTD са неколико врста ограничења: кључеви, страни кључеви и ограничење референцијалног интегритета. Аутори су увели ограничења у XML модел података, тако што су предложили три језика ограничења, L, L_u и L_{id}. Први језик, L, служи за опис ограничења која важе у релационим базама података, тако што су уведени појмови кључа и страног кључа. Други језик, L_u, је минимално проширење оригиналног ID/IDREF механизма, које омогућује да атрибут типа ID има јединствену вредност само међу елементима истог типа, а не на нивоу целог документа . Трећи језик, L_{id}, задржава оригинално значење атрибута типа ID, а то је да вредност таквих атрибута мора бити јединствена на нивоу целог документа. Допринос овог рада је у проширењу DTD-а дефиницијама кључева, страних кључева и ограничења референцијалног интегритета. Мана оваквог приступа је што се базира на DTD-у.

Због хијерархијске структуре XML докумената, постоји потреба да се одреди опсег важења јединствености неког елемента. Поставља се питање да ли је тај елемент јединствен у оквиру целог документа, или је јединствен само у делу тог документа. Због тога су уведени појмови апсолутног и релативног кључа.

У раду [Bun02] постављени су темељи разматрања о кључевима у XML документима. Уведен је појам релативног кључа који се често користи у хијерархијски структуираним документима. У [Bun03] даље се разматрају класе XML кључева уведених у раду [Bun01]. Различите спецификације кључа за XML су до тада дефинисане у XML стандарду [XML], XML Data [XMLD] и XML Schema-и [XMLSch]. У DTD документима, сваки елемент се може јединствено идентификовати коришћењем атрибута типа ID. Међутим, атрибутима типа ID у DTD-у опсег важења је цео документ. Вредност једног атрибута типа ID је јединствена у целом документу, а не само међу елементима једног типа. На пример, у једном документу не могу два различита елемента, као што су студент и предмет, имати исту вредност атрибута ID. Коришћење атрибута ID даје и ограничење да кључеви могу имати само један атрибут, а и елемент може имати само један кључ.

За дефинисање кључа у раду [Bun02] користе се: скуп над којим се дефинише кључ – у релационој бази података то је релација (скуп торки који се идентификује преко назива и шеме релације) и атрибути који заједно јединствено идентификују елементе у скупу. Кључ је пар (Q, {P1, ..., Pn}), где је Q израз путање, а {P1, ..., Pn} је скуп простих израза путања. Q идентификује скуп чворова над којима важи ограничење кључа. Q је циљна путања (*target path*), а скуп {P1, ..., Pn} је путања кључа (*key path*). Ово одговара апсолутној и релативној путањи у XPath терминологији.

Дата је дефиниција релативног кључа, за којим се појавила потреба у хијерархијским структурама. И у релационим базама података јавља се хијерархијска структура кључа: кључ слабог типа ентитета се састоји од кључа надређеног регуларног типа ентитета и од

идентификатора слабог типа ентитета. Да би била објашњена хијерархијска структура кључа, уведен је појам релативног кључа, који се састоји од пара (Q, K), где је Q израз путање, или само путања, а K је кључ. Нека је n[[Q]] скуп чворова до којих се стиже из чвора n пратећи путању Q. Са [[Q]] је скраћено обележен израз root[[Q]], односно скуп свих чворова до којих се стиже из корена пратећи путању Q. Документ задовољава релативни кључ (Q, (Q', S)), ако и само ако за све чворове n из [[Q]], n задовољава кључ (Q', S). То значи да је (Q, K) релативни кључ, ако је K кључ сваког поддокумента са кореном у чвору који припада [[Q]].

На пример, у кључу (bible.book.chapter, (verse, {number})) број стиха *verse number* јединствено одређује стих *verse* у оквиру поглавља *chapter*.

У кључу (bible.book, (chapter, {number})) број поглавља *chapter number* јединствено идентификује поглавље *chapter* у оквиру књиге *book*.

У кључу (bible, (book, {name})) постоји само један чвор *bible* одмах испод коренског елемента.

У релативном кључу (Q, (Q', S)), Q полази од корена, док, Q' полази од чвора [[Q]].

Један од главних доприноса овог рада је увођење новог предлога дефиниције кључа, полазећи од анализе и објашњавања нотације релативног кључа и разумевања структуре хијерархијског кључа. Последица овакве дефиниције кључа је да се у подређеним елементима не понављају атрибути из надређених елемената, који представљају стране кључеве. Међутим, за дефинисање ограничења, као што је, на пример, проширено ограничење референцијалног интегритета, неопходно је поновити неке од атрибута и у подређеним елементима, што није могуће помоћу механизма приказаног у овом раду.

У неколико радова разматран је проблем рачунске сложености алгоритама помоћу којих се проверавају ограничења. У раду [Fan01] аутори испитују сложеност проблема да ли постоји XML документ који задовољава и DTD и ограничења. Проблем је у општем случају неодлучив. Када се ради о кључевима који садрже само једно обележје и о страним кључевима, проблем је NP комплетан. XML документ се валидира у односу на DTD који га описује, а обично постоје и дефинисана ограничења, као што су кључеви и страни кључеви. Аутори постављају питање да ли је таква спецификација конзистентна, тачније, да ли постоји коначни XML документ који задовољава и ограничења и валидан је у односу на дати DTD. У овом раду су разматрани само примарни и страни кључеви, али је остало отворено питање осталих ограничења која се јављају у базама података, пре свега за инверзна ограничења.

Рад [Wan03] описује везу између XML кључева и релационих кључева. Предложен је језик за трансформацију XML података у релациону базу података, када је потребно XML складиштити у релациону базу података. Такође, дат је алгоритам за извођење кључева у релационом моделу из XML кључева. Проблем са XML-ом који је описан у раду је тај што XML представља само синтаксу, а не носи информацију о семантици података. У релационим базама података, ограничења су веома корисна у креирању добре базе података без аномалија, са оптимизацијом упита и ефикасним складиштењем и приступањем. Због тога се доста радова бави ограничењима за XML. Ограничење кључа је посебно важан део базе података, посебно релационе базе података. Дефиниција кључа у овом раду одређује да су хијерархијски подређени чворови одређени својим претходницима, а да чворови рођаци не могу да одређују једни друге. Аутори су показали да постоје случајеви у којима предложени алгоритам не омогућава аутоматску конверзију XML кључева у релационе кључеве, него је потребна ручна интервенција.

2.6 Нормалне форме и функционалне зависности у XML моделу података

Да би били избегнути редунданса података и аномалије ажурирања, као у релационим базама података, потребно је дефинисати и нормалне форме за XML документе. У радовима [Are04, Are06, Are06] аутори уводе појам нормализације за XML документе. Редундансе се јављају због постојања одређених функционалних зависности међу путањама у документу. Циљ рада је да се нађе начин да се произвољни DTD преведе у такав DTD у којем не постоје проблеми са редундансом података. Уводи се појам функционалне зависности, затим се дефинише XML нормална форма (XNF), којом се избегавају аномалије ажурирања и редундансе. Показано је да је XNF генерелизација BCNF. На крају је дат алгоритам који конвертује било који DTD документ у такав DTD који је у XNF.

Зависности и нормалне форме, које су детаљно обрађене у релационом моделу података, захтевају генерализацију [Sch05], како би могле бити примењене и на XML модел података, тачније на структуре типа стабла којима се представљју XML документи.

Поред обичних функционалних зависности, дефинисане су и локалне функционалне зависности [Liu03]. То су функционалне зависности које важе само у једном делу XML документа, а не у целом документу. У раду су дефинисане и доказана је коректност аксиома за импликациони проблем локалних фунцкионалних зависности у XML-у. Испитана је и веза између локалних функционалних зависности и кључева и показано је да је релативни кључ специјалан случај локалне функционалне зависности.

Једна група радова [Vin04, Vin07, Sha08a] односи се на дефинисање XML функционалних зависности и њихову примену у нормалним формама за XML, али та тема превазилази оквире ове дисертације, па није посебно разматрана.

У [Dav07] аутори предлажу алгоритме који проверавају да ли је функционална зависност изведена из XML кључа. Такође, описан је алгоритам који налази минимални покривач скупа функционалних зависности изведених из XML кључева.

2.7 Ограничење референцијалног интегритета у XML моделу података

Ограничење референцијалног интегритета је једно од ограничења које се јавља у свим моделима података, у различитим формама. Како је XML све више у употреби, ограничење референцијалног интегритета је дефинисано и у XML моделу података. Неки од радова који се баве овом врстом ограничења за XML су [Fan05, Fan00].

У раду [Sha08], дефинисани су XML зависности садржавања (*XML Inclusion Dependency* – XID) и XML страни кључ (XFK) над DTD-ом, а дати су и XML документи који задовољавају ова ограничења. Уводи се појам торке (*tuple*) која обезбеђује коректне вредности у XML документима након провере валидности. Такође се уводи појам двосмислености торке, код којег је торка двосмислена ако се путања свих елемената торке разликује за више од једног чвора, почев од коренског елемента. Показује се и да је дефиниција XFK комбинација XID и XML кључа. Да би XFK био задовољен, што значи да морају бити задовољени и XID и XML кључ, потребно је генерисати торке које нису двосмислене.

У примеру је дат DTD документ који описује похађање наставе од стране студената на неком департману. Елемент studs садржи атрибуте sid и sname. Елемент courses садржи атрибуте cid и cname, а елемент enroll повезује студенте у курсеве и садржи атрибуте cid и sid.

```
<!ELEMENT depts(did, studs, courses, enroll)+>
<!ELEMENT studs(sid, sname)+>
<!ELEMENT courses(cid, cname)+>
<!ELEMENT enroll(cid, sid)+>
```

Сваки департман се идентификује преко did и може да садржи више елемената studs, courses и enroll. Свака вредност атрибута sid из елемента enroll мора да постоји у елементу studs. Такође, свака вредност атрибута cid из елемента enroll мора да постоји у елементу courses. У раду је предложена нотација за записивање ових ограничења:

- ϒ(depts, ({enroll/sid} ⊆ {studs/sid})) и
- ϒ(depts, ({enroll/cid} ⊆ {courses/cid})).

Знаком ϒ означена је XML зависност садржавања (XML Inclusion Dependency – XID). Путања path је селектор (*selector*). Путање {enroll/sid} и {enroll/cid} називају се *dependents*, а путање {studs/sid} и {courses/cid} *referenced*. Селектор почиње од коренског елемента и спаја се са зависним и референцираним путањама.

XML страни кључ (XML foreign key - XFK) је дефинисан помоћу XID и XML кључа, који је овде апсолутни кључ. XFK је записан као *F*(depts, ({enroll/sid} ⊆ {studs/sid})), где је ϒ(depts, ({enroll/sid} ⊆ {studs/sid})) XID, а (depts, {studs/sid}) је XML кључ. Слично, *F*(depts, ({enroll/cid} ⊆ {courses/cid})) је XFK, gde je ϒ(depts, ({enroll/cid} ⊆ {courses/cid})) XID, а (depts, {courses/cid}) је XML кључ.

Доприноси рада [Sha08] су:

- дефинисање XML страних кључева над DTD-ом, при чему се дефинишу и XML зависности садржавања (XID) и XML кључеви над DTD-ом. DTD је узет у обзир, јер има једноставнију структуру у односу на XML Schema-у,
- формирају се XML документи који задовољавају XID и XML кључеве. Уводи се појам торке. Задовољење ограничења подразумева да се спречава додавање торки које су двосмислене и
- даје се дискусија која показује на који начин су дефиниције XID и XFK корисне у моделирању XML докумената.

Дефиниције XID и XFK дате у овом раду могу се користити приликом модификације и брисања у XML документу, приликом нормализације и XML упита.

Ограничења у [Sha08] дефинисана су над DTD-ом, али XML Schema има много веће могућности за дефинисање типова података и других врста ограничења од DTD-а.

У раду [Sha09] наставља се дискусија о XML зависностима садржавања (XID) и XML страним кључевима (XFK). Посебно се обраћа пажња на примену XML референцијалниг интегритета. Користи се као метрика за проверу квалитета података у смислу редундантности и козистентности. Користи се концепт торке. Торка садржи семантички коректне вредности које задовољавају и XID и XFK. У раду се наводи да ова особина није доступна ни у DTD-у, који поседује ID и IDREF механизме, ни у XML Schema-и, која поседује Key и KeyRef механизме. Дефиниција ограничења дефинисаних у овом раду може да се користи [Sch05, Are06, Bun01] за нормализацију, брисање и модификацију у XML документима, као и за управљање XML упитима и погледима.

У циљу дефинисања ограничења која важе и у релационом моделу, требало би изабрати такав начин формирања XML Schema документа, који ће, између осталог, омогућити проверу ограничења референцијалног интегритета и јединствености вредности кључа. Како хијерархијска структура то не омогућава, што је истакнуто у [Nei06], требало би се определити за релациону структуру, односно такав начин формирања XML документа, код кога су сви елементи на истом нивоу испод коренског елемента [Nei06].

У хијерархијској структури, за елемент који је подређен неком елементу подразумева се да кључ надређеног елемента важи као страни кључ и у подређеном елементу. Између подређеног и надређеног елемента важи веза „родитељ-дете“ и исказана је идентификациона зависност подређеног од надређеног елемента. Ако је потребно да подређени елемент садржи и референцу ка неком другом елементу, то се не може исказати хијерархијском структуром. На пример, ако треба исказати везу са кардиналитетима „M:N“, елемент који представља повезник могао би бити наведен као подређени елемент једног од повезаних елемената, али веза ка другом повезаном елементу морала би бити исказана као додатни атрибут у оквиру елемента који представља повезник. У таквим ситуацијама боље би било користити релациону структуру XML документа.

2.8 Генерисање XML Schema-е из релационе шеме базе података

Када је XML постао опште прихваћени формат за размену података, јавила се и потреба да се подаци из релационе базе података представе помоћу XML докумената, као и да се XML подаци складиште у специјално дизајниране релационе базе података, и да се над тим подацима извршавају упити. Процес трансформације релационе базе података у XML би требало да буде аутоматски, и самим тим независан од утицаја корисника. Једна од најважнијих ствари о којој треба водити рачуна је очување свих ограничења и њихово задовољење приликом ажурирања базе података. У [Jum08] приказано је генерисање XML датотека на основу резултата упита из релационе базе података и обрнуто. Процес трансформације у оба смера обављају специјализовани модули засновани на правилима (*rule-based modules*). На основу релационе шеме базе података направљена је XML Schema која се користи у модулима за трансформацију. Тако добијена XML Schema има многе предности: може се користити за приказ XML погледа релационих података, дозвољава проверу да ли XML документ има аномалије пре ажурирања, узима у обзир хијерархијски поглед табела у бази података. У раду није описан поступак добијања XML Schema-е на основу релационе шеме базе података, већ је дат детаљан опис трансформације података у оба смера.

До сада су у радовима приказани различити начини трансформације релационе базе података у XML. Према [Kri03] XML подаци се складиште и посебно дизајнирану релациону базу података која је креирана према концептуалном моделу заснованом на XML структури стабла [Ata07]. У раду [Jum10] предлаже се складиштење у већ постојеће релационе базе података. У истом раду, предлаже се и екстракција података из релационе базе података и њихова репрезентација у облику XML–а. Тада је могуће извршавати XML упите над XML погледима релационих података [Bar99, Fer02, Sha01a]. Недостатак оваквог приступа је што XML погледи не подржавају ограничења, што представља проблем приликом писања XML упита.

Неке групе аутора предлажу трансформацију релационе шеме у XML шему. Метода групе аутора [Lee01] шеме релација и атрибуте из релационе базе података трансформише у елементе и атрибуте у DTD-у. Даље, исти аутори презентују још два алгоритма [Lee02, Lee06]. Још један алгоритам [Lv07] трансформише релациону шему базе података у DTD користећи функционалне зависности. За разлику од свих предложених алгоритама, у раду [Jum10]

предлаже се алгоритам који ће користити XML Schema-у и омогућити очување ограничења која су дефинисана у релационој бази података. Постоји још радова у којима је коришћена XML Schema. Релациона шема може прво да се трансформише у проширени ЕР модел, који се онда таква ЕР шема базе података трансформише у XSD граф, од којег коначно настаје XML Schema [Fon05]. У [Yan08] релациона шема се директно трансформише у XML шему, узимајући у обзир само зависности садржавања.

У [Jum10] описана је метода ефикасног генерисања XML Schema документа из релационог модела података. Метода се састоји из два корака, који имају за циљ да очувају ограничења и да избегну редундансе података тако што ће користити висок ниво угњеждавања. Први корак садржи алгоритам који раслојава све релације из базе података у различите нивое у зависности од постојања функционалних зависности и степена страног кључа. Други корак садржи алгоритам који, на основу скупа генеричких XML Schema докумената аутоматски генерише нове XML Schema документе према класификацији из првог корака. Метода је тестирана у бази података једног здравственог система који користи XML документе. Резултујућа шема садржи уредно угњеждене делове без редунданси, што може да омогући и аутоматско генерисање SQL упита који би извршавали промене над релационом базом података, која се налази испод овако генерисане шеме.

Уводе се два нова концепта: ниво релације и степен страног кључа. На нултом нивоу се налазе шеме релација које немају стране кључеве. Страни кључ нултог нивоа је страни кључ који се односи на шему релације на нултом нивоу. Шема релације на првом нивоу је она чији су сви страни кључеви првог нивоа. Страни кључ првог нивоа је страни кључ који се односи на шему релације на првом нивоу. Шема релације на N-том новоу је шема релације чији су сви страни кључеви M-тог степена, где је M<=N-1, а страни кључ M-тог степена је страни кључ који се односи на шему релације на M-том новоу.

Како се релациона шема базе података састоји од скупа шема релација, скупа атрибута сваке шеме релације, домена за сваки атрибут шеме релације, примарног кључа сваке шеме релације, скупа страних кључева, скупа ограничења (unique, not null), за сваку од ових компонената се мора дефинисати скуп шаблона који ће помоћи приликом генерисања одговарајућих делова XML Schema документа. У раду је дато девет шаблона.

Оно што је у раду значајно за ову дисертацију је начин представљања везе „родитељ-дете“. Описана су два начина за представљање страног кључа. Први начин је да се користи угњеждавање елемената и да се на тај начин одреди који елемент је надређени, а који подређени. Тада се у подређеном елементу не наводе атрибути који представљају страни кључ, а припадају надређеном елементу. Други начин представљања ове везе је да се користи keyref елемент у XML Schema-а документу, а онда елемент који је подређени није угњежден у оквиру надређеног елемента, и садржи све атрибуте који представљају страни кључ.

Нека је fk ограничење страног кључа дефинисано над скупом атрибута a_1, ... a_n шеме релације r, а који се односи на скуп атрибута a_{p1}, ..., a_{pn} у надређеној шеми релацији r_p. Према првом начину представљања страног кључа, шема релације r је може директно угњездити као поделемент у елементу који представља надређену шему релације r_p. Тада се скуп атрибута a_1, ..., a_n не наводи у скупу атрибута шеме релације r, што је приказано у Листинг 8.

```
<xs:element name="r_p">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="r" minOccurs="0" maxOccurs="unbounded">
    <xs:complexType>
     <!--Here are the attributes of r except those of fk-->
    </xs:complexType>
    <!—Here are the identity constraints of relation r-->
```

```
   </xs:element>
  <xs:sequence>
  <xs:attribute name="ap1" type="DT"/>
  ...
  <xs:attribute name="apn" type="DT"/>
  <!-- Here are the other attributes of rp -->
  </xs:complexType>
 <xs:key name="rp_pk">
  <xs:selector xpath=".//r" />
  <xs:field xpath="@ap1" />
  ...
  <xs:field xpath="@apn" />
 </xs:key>
 <!-- Here are the other identity constraints of rp -->
</xs:element>
```

Листинг 8 Шаблон за дефинисање подређеног елемента у оквиру надређеног

На овај начин могуће је пресликати само једно ограничење страног кључа. Остали страни кључеви морају бити пресликани помоћу елемента keyref за сваки страни кључ.

Нека је fk1 ограничење страног кључа дефинисано над скупом атрибута a_1, ... a_n шеме релације r, а који се односи на примарни кључ у надређеној шеми релације r_1. Тај примарни кључ пресликан је елементом key, а страни кључ fk_1 пресликан је елементом keyref, што је приказано у Листинг 9.

```
<xs:element name="r1" minOccurs="0" maxOccurs="unbounded">
 <xs:complexType>
  <xs:attribute name="a1" type="DT" />
   ...
  <xs:attribute name="an" type="DT" />
  <!--Here are the other attributes of relation r1 -->
 </xs:complexType>
 <xs:key name="r1_pk">
  <xs:selector xpath=".//r1" />
  <xs:field xpath="@a1" />
   ...
  <xs:field xpath="@an" />
 </xs:key>
 <!--Here other identity constraints of relation r-->
</xs:element>
<xs:element name="r" minOccurs="0" maxOccurs="unbounded">
 <xs:complexType>
  <xs:attribute name="a1" type="DT"/>
   ...
  <xs:attribute name="an" type="DT"/>
  <!-- Here are the other attributes of r -->
 </xs:complexType>
 <xs:keyref name="fk1" refer=" r1_pk">
  <xs:selector xpath=".//r" />
  <xs:field xpath="@a1" />
   ...
  <xs:field xpath="@an" />
 </xs:keyref>
 <!-- Here are the other identity constraints of rp -->
</xs:element>
```

Листинг 9 Шаблон за дефинисање надређеног и подређеног елемента

Када је потребно нагласити да неки атрибут не сме имати нула вредности, у елементу xs:attribute у XML Schema документу наводи се атрубут use чија је вредност required, што је искоришћено и у одељку 3.1.2, приликом дефиниције ограничења вредности атрибута.

```
<xs:attribute name="a" type="DT" use="required"/>
```

И у овом раду описују се само ограничења примарног и страног кључа, ограничење јединствености и дозвола или забрана нула вредности. У овој докторској дисертацији усвојено

је правило да се страни кључеви представљају помоћу елемента keyref и навођењем свих атрибута који треба да представљају страни кључ у подређеном елементу.

У [Dav07] развијен је језик за трансформацију XML докумената у релациону базу података без потребе за DTD-ом или шемом. Такође, дат је и алгоритам којим се проверава да ли су функционалне зависности, који произилазе из XML кључева, обухваћене овом трансформацијом. Резултати рада се могу уопштити тако да се добије комплетан скуп функционалних зависности из XML кључева.

2.9 Класификација типова ограничења у XML базама података

У релационом моделу података постоје различите таксономије ограничења. Више аутора дало је предлоге подела, па тако се у [Sil05] предлаже класификација: домен, кључ, везе и ограничење референцијалног интегритета. У [Dat03] подела је следећа: домен, атрибут, торка, транзиција стања (*state transition*), кључ, ограничење референцијалног интегритета и тренутак извршења акције (*check point*).

Мали број доступних радова предлаже таксономију ограничења за XML модел података. У раду [Bun02] аутори обрађују ограничење кључа, и дефинишу две катеогорије кључева: кључ, који је исти као и у релационом моделу података, и релативни кључ, који се састоји од кључа и израза путање у XPath-у. Аутори рада [Ben03] предлажу категорије: тип, кључ и инклузија, која је заснована на ограничењу страног кључа. У раду [Pav00] дати су: интегритет, који се састоји од категорија тип, путања и комплексна ограничења, и валидност података. Апсолутна и релативна ограничења, као и унарна и вишеатрибутска ограничења, су категорије предложене у [Are02]. Главни допринос овог рада је класификација ограничења према количини података која је потребна за верификацију ограничења, што неки аутори називају и опсег ограничења.

У [Fan03] главна категорија је названа XML основна ограничења, као што су кључ или страни кључ. Таксономија предложена у раду [Jac02] ограничења организује у мале специфичне групе, али не узима у обзир опсег ограничења и синтаксну структуру ограничења. Једна група ограничења је чак названа ''друга ограничења''.

Таксномија коју су предложили аутори рада [Hu04] уводи форму ограничења као нови појам, и разликује динамичка и статичка ограничења. Али ни у овом раду се не спомиње оспег ограничења.

Постоје радови у којима се обрађује појам ограничења домена [Bon02, Ogb00, Pro02], што је била полазна основа за рад [Laz05], у којем је ограничење домена проширено по угледу на ограничење домена у релационим базама података [Dat03, Elm03]. Међутим, ни у овом раду аутори не разматрају опсег ограничења, нити форму ограничења. Приказана је контрола XDC (XML Domain Constraint Control) која има за циљ да подржи ограничења домена која нису подржана у XML Schema-и. Предложен је XML језик за спецификацију ограничења домена, који је назван XDCL, као и парсер који валидира таква ограничења. XDCL омогућава спецификацију услова и извршавање акција, као што је то могуће у SQL-у. XDC је независан механизам који могу да користе корисници, апликације или XML базе података за валидацију ограничења домена.

Ограничење домена у базама података ограничава дозвољене вредности атрибута. Због хијерархијске структуре XML докумената, у раду [Laz05] претпостављено је следеће:

- тип ограничења домена обухвата структуралну компоненту, као што су поделементи, секвенце или избори, и кардиналитет поделемената,
- атрибут дефинише да дозвољене вредности могу бити прости елементи или атрибути,
- ограничење торке валидира садржај елемента или атрибута који припадају истом нивоу хијерархије у XML документу, то јест, све елементе и њихове атрибуте који су директни потомци елемента E, укључујући и атрибуте елемента E,
- ограничење базе података валидира елементе или атрибуте који припадају различитим нивоима хијерархије XML документа и
- ограничење транзиције стања разматра транзиције валидних вредности у просте елементе и атрибуте.

Таксономија ограничења за XML модел података описана је у раду [Rod07]. Разликује се неколико фасета ограничења. Предложени фасети базирани су на компонентама и концептима ограничења у релационом моделу података. Ограничења се састоје од пет компонената:

- објекти који се користе у спецификацији ограничења,
- објекти над којима се дефинише ограничење,
- логички израз који повезује ове две врсте објеката,
- тренутак када ограничење треба да буде проверено и
- акције које се спроводе када је ограничење нарушено.

Таксономија ограничења у овом раду је организована према следећим аспектима:

- вредност ограничења која обухвата рестрикције XML вредности,
- опсег ограничења одређује количину чворова XML документа који су укључени у спецификацију ограничења:
    - податак – садржај јединственог атрибута или простог елемента,
    - торка – садржај скупа атрибута и/или простог елемента из комплексног елемента,
    - елемент – садржај скупа атрибута и/или поделемената истог комплексног елемента,
    - репозиторијум – садржај скупа атрибута и/или простог елемента из различитих комплексних елемената,
- форма ограничења– користи се нотација за дефинисање ограничења:
    - заснован на boolean изразима, где се израчунава предикат и враћа израз true или false,
    - заснован на правилима, где су boolean изрази уграђени у изразе и који омогућавају или не омогућавају извршавање неке акције над XML документом, који гарантују да је интегритет података очуван,
- тренутак извршавања акције:
    - тренутан – ограничење се проверава у тренутку извршавања операције,
    - одложен – ограничење се проверава након извршавања операције,
- акција – акција која се спроводи када је ограничење нарушено:
    - информативно – приказује грешку и опозива операцију,
    - активно – извршава друге операције над XML документом како би био очуван интегритет података.

Аутори сматрају да је оваква таксономија ограничења добра основа за дефинисање језика за њихову спецификацију, као и формирање модула за XML базе података који се баве ограничењима.

2005. године патентиран је прошириви језик за ограничења (Extensible constraint markup language - XCML). XCML [Hu05] је богатији од до тада постојећих језика за дефинисање ограничења и има бољу подршку за спецификацију динамичких и међурелационих ограничења. За визуелну спецификацију користи се Unified Modeling Language (UML), а за аутоматско генерисање инстанце XCML докумената и XML Schema докумената, користи се Object Constraint Language (OCL).

XML документи могу бити валидирани у односу на DTD или XML Schema документе. То је синтаксна валидација, која спречава уношење погрешних података у XML документ. Међутим, веома су важна и неструктурална ограничења над XML подацима. На пример, присуство једног елемента може да зависи од вредности или присуства неког другог елемента, док опсег вредности елемента може да буде различит за различите документе. Иако је XML Schema много моћнија од DTD-а, не може да се користи за спецификацију неструктуралних ограничења. Због тога се и јавила потреба за новим проширивим, платформски неутралним и доменски независним начином специфицирања ограничења за XML документе.

Посебна тема којом су се аутори бавили је и валидација ограничења. Парсери за валидацију не могу да користе документе са ограничењима да би валидирали неструктурална ограничења. Уграђивање таквих ограничења у сам програм није било погодно решење, јер такви програми нису погодни за проширивање и промене, и не могу се поново користити у другачијим ситуацијама. Модерне XML технологије би требало да понуде генеричко окружење за аутоматску валидацију ограничења.

Док синтаксна XML ограничења описују статичку структуру XML документа, ограничења намећу статичка или динамичка ограничења на вредности или појављивање елемената или атрибута у XML документу.

Ограничење се може изразити у облику израза који има вредност тачно или нетачно, или помоћу условног правила са уграђеним изразима.

XCML синтакса је дефинисана у XML Schema документу. XCML документ садржи један коренски елемент Constraints, који садржи секвенцу од једног или више елемената Constraint. Елемент Constraint мора да одреди опсег важења ограничења у атрибуту context. Даље се опционално наводи секвенца елемената Parameter, који специфицирају назив, тип и опционално подразумевану вредност параметра који се прослеђује. Као поделементи елемента Constraint јављају се или елемент Rule или елемент Assertion. Елемент Rule је секвенца елемената If, Then и Else.

У Листинг 10 приказано је да у контексту елемента employee, вредност атрибута taxRatemора бити једнака параметру rate, који се динамички поставља у систему.

```
<Constraint context="employee">
 <Parameter>
 <name>rate</name>
 <type>decimal</type>
 <defaultValue>0.07</defaultValue>
</Parameter>
 <Assertion test=”taxRate=$rate”/>
</Constraint>
```

Листинг 10 Контекст елемента employee

У ограничењу је могуће задати и израз, тако да се у Листинг 11, у контексту елемента employee, вредност атрибута tax израчунава тако што се вредност елемента income множи са вредношћу параметра rate, који се поставља у програму.

```
<Constraint context=“employee”>
```

```
  <Parameter>
    <name>rate</name>
    <type>decimal</type>
    <defaultValue>0.07</defaultValue>
  </Parameter>
  <Assertion test="tax=income*$rate"/>
</Constraint>
```

Листинг 11 Рачунање вредности атрибута на основу вредности другог елемента

У Листинг 12 дат је и услов, на основу ког се израчунава вредност атрибута taxRate.

```
<Constraint context="employee">
  <Parameter>
    <name>level</name>
    <type>decimal</type>
  </Parameter>
  <If test="income<=$level"/>
  <Then test="taxRate=0.05"/>
</Constraint>
```

Листинг 12 Условно рачунање вредности атрибута

У раду се могу видети и остали примери задавања ограничења.

Исти аутори објавили су и рад [Hu06] у којем дају преглед постојећих језика за дефиницију XML ограничења, и класификују типове ограничења.

Ограничења се могу поделити у следеће категорије:

- ограничења која проверавају да ли су документи добро формирани,
- ограничења која проверавају структуру XML документа,
- ограничења која проверавају тип података или формат,
- ограничења која проверавају вредност атрибута или елемената, за елементе или атрибуте чија се вредност или оспег вредности не могу задати у DTD-у или шеми (таква ограничења могу бити статичка или динамичка),
- ограничења која проверавају присуство атрибута или елемената, као и број појављивања елемента, која такође могу бити статичка или диманичка,
- међурелациона ограничења између елемената или атрибута, када присуство или вредност елемента или атрибута зависи од присуства или вредности неког другог елемента или атрибута,
- ограничења конзистентности, где одговарајући елементи или атрибути имају конзистентне вредности.

Прве три категорије односе се на ограничења исказана структуром и она могу бити дефинисана у DTD-или у XML Schema-и и валидирана помоћу XML парсера за валидацију. Преостале четири категорије односе се на ограничења исказана путем посебног језика и израза у том језику. Ограничења у четвртој и петој категорији чешће се специфицирају као assertions, док се ограничења у шестој и седмој категорији специфицирају помоћу условних правила.

Ограничења приказана у овом раду добра су полазна основа за дефиницију ограничења. Иако уводе дефиниције условне зависности, елементе процесирања, као што су гранања, ова ограничења не могу да се искористе за комплексније проблеме.

Иако је XML Schema богатија од DTD-а у изражавању структура, типова података и формата података, ипак није довољно моћна да изрази ограничења. У [XFront] предложене су три опције за проширење XML Schema-е да би било омогућено изражавање ограничења:

- додавање новог језика за XML ограничења XML Schema-и,
- изражавање ограничења у програмском коду,
- изражавање ограничења помоћу XSLT-а и XPath-а.

Аутори предлажу прву опцију као најбољу, јер уколико се користи програмски код, у њему могу да се изразе сва ограничења, али се тешко повезује са XSLT технологијом, јер је свако ограничење посебна апликација. Код треће опције, свака апликација креира свој *stylesheet* за спецификацију и проверу ограничења, који је јединствен баш за ту апликацију. Такође, *stylesheet*-ови не могу поново да се искористе и нису лако читљиви (енг. *human-oriented*).

Предложене таксономије ограничења у XML моделу података могу бити основа за дефинисање таксономије у овој докторској дисертацији, с тим што се дефинишу типови ограничења који су дати у релационом моделу података, а неки аспекти, као што су опсег ограничења и време извршавања акције, биће усвојени из датих радова.

2.10 Тригери у XML базама података

Развој концепта тригера пратио је развој релационих система за управљање базама података. Чим се појавио појам ограничења, настала је потреба да се одреагује на нарушавање услова интегритета. У базама података сви корисници зависе од тачности или валидности података, а тригери се могу користити да спроведе провера валидности.

Концепт тригера уведен је у системе за управљање базама података, како би се аутоматски реаговало на нарушавање ограничења. W3C још увек није укључио тригере у стандард, али постоје радови који се баве тригерима и пружају неке идеје за њихову реализацију у XML базама података.

Како је број XML система за управљање базама података растао, јавила се потреба да се системи за управљање XML базама података развију на сличан начин као и системи за управљање релационим базама података.

У раду [Tat01] дат је скуп примитивних операција за ажурирање XML документа, које су предложене као проширење језика XQuery. Многи XML упитни језици су предлагани у претходном периоду, али аутори су се доминантно опредељивали за XQuery стандард. На почетку развоја XML-а, развијани су упитни језици XML-QL [Deu99] и XQuery [XQuery]. eXcelon XML [eXcelon] је један од ретких XML система који подржавају додавање и брисање коришћењем проширења XPath језика. Проширења упитног језика предложена у наведеном раду [Tat01] омогућавају додавање и брисање на вишеструким нивоима унутар хијерархије, како би биле подржане и комплексне структуре.

Предложене су операције за ажурирање структуре и садржаја XML документа. Такође, предложено је проширење XQuery језика. Дат је алгоритам за имплементацију ажурирања XML докумената сачуваних у релационој бази података као XML погледа.

XQuery користи XML упитни модел података [XDM] који XML документ посматра као стабло са обележеним чворовима и референцама. И у овом раду [Tat01] је коришћен XML упитни модел. Наведен је скуп операција ажурирања над XML документом, које могу да буду примењене не само на вредностима у чворовима-листовима, него и на комплекне структуре, комплексне XML Schema типове и њихове изведене подтипове. Такође, прави се разлика између IDREF типа података и садржаја атрибута, јер IDREF садржи структурну информацију и могу да се појаве унутар уређене IDREFS листе, док атрибути садрже конкретне вредности.

На пример, ако је у питању операција брисања, а брише се неки чвор дете, довољно је само обрисати га. Валидни типови за чвор дете су PCDATA, атрибут, IDREF унутар листе IDREFS, и елемент. Ако је чвор дете референца у оквиру IDREFS, брише се само тај конкретан чвор, а остатак листе IDREFS не мења се. Остале операције су описане у раду.

XQuery је проширен структуром FOR...LET...WHERE...UPDATE. У оквиру UPDATE клаузуле, специфицира се секвенца операција:

```
FOR $binding IN XPath-expr, ...
LET $binding := XPath-expr, ...
WHERE predicate1, ....
updateOp, ...
```

где је updateOp:

```
UPDATE $binding {subOp, {, subOp}*}
```

где је subOp:

```
DELETE $child |
RENAME $child TO name |
INSERT content [BEFORE | AFTER $child] |
REPLACE $child WITH $content |
FOR $binding IN XPath-subexpr, ...
WHERE predicate1, ... updateOp
```

Ово проширење упитног језика имплементирано је у оквиру релационе базе података, јер су XML подаци били складиштени у њој.

Један од предлога спецификације тригера за XML дат је у раду [Bon01]. Active XQuery је језик за XML базе података, који је заснован на претходно дефинисаном XQuery update моделу [Tat01]. У раду се даје синтакса и семантика језика. Проблем ажурирања XML докумената је и даље тема истраживања и постоје многи приступи који та ажурирања подржавају. Active XQuery нуди један општи приступ за изражавање промена над било којим подацима који су представљени као XML структура, без обзира на то да ли су ти подаци смештени у XML базу података, или се налазе у релационој бази података као XML погледи.

Главни доприноси рада [Tat01] су предлози проширења W3C стандарда за XQuery, где се користе BEFORE и AFTER тригери, као и нивои грануларности ROW и STATEMENT. Такође, предлаже се алгоритам за трансформацију великих израза у мање делове, над којима се извршавају ажурирања и покрећу тригери. Аутори су користили [Coc99] стандард као основу за дефинисање Active XQuery језика. Тај стандард специфицира синтаксу и тригере у релационом моделу података.

XQuery тригер се састоји од четири компоненте: операција која изазива тригер, грануларност тригера, услов и акција. Тригер се окида када се појави једна од операција која изазива тригер. Затим се проверава да ли је задовољен услов задат у тригеру, и извршава се акција.

Дат је следећи предлог синтаксе дефиниције тригера:

```
CREATE TRIGGER trigger-name
[WITH PRIORITY Signed-Integer-Number]
(BEFORE | AFTER)
 (INSERT | DELETE | REPLACE | UPDATE-CONTENT)+
 OF XPathExpression (, XPathExpression)*
[FOR EACH (NODE | STATEMENT)]
[XQuery-Let-Clause]
[WHEN XQuery-Where-Clause]
```

```
DO (XQuery-UpdateOp | ExternalOp)
```

Клаузула CREATE TRIGGER користи се у циљу дефинисања новог XML тригера с предложеним називом. Може се задати редослед правила у оквиру клаузуле WITH PRIORITY. Редослед се задаје било којим целим бројем, а ако се ова клаузула изостави, приоритет је по дефиницији нула. Клаузуле BEFORE/AFTER одређују време када ће тригер бити покренут у односу на неку операцију ажурирања. Сваки тригер је повезан са скупом операција ажурирања (INSERT, UPDATE, DELETE), а проширен је и операцијом UPDATE-CONTENT коју су увели аутори рада [Tat01] у проширењу језика XQuery.

Операција се односи на елементе који одговарају XPath изразу наведеном после кључне речи OF. Дозвољено је навести један или више предиката као XPath филтер, како би били елиминисани чворови који не задовољавају дати услов.

Израз FOR EACH NODE/STATEMENT је опционалан и изражава грануларност тригера. Тригер на нивоу израза (STATEMENT) извршава се једном за сваки скуп чворова који задовољавају услов у XPath изразу. Тригер на нивоу чвора (NODE) извршава се једном за сваки од чвор. У зависности од грануларности тригера, уведене су и следеће променљиве:

- ако је тригер креиран на нивоу чвора, променљиве OLD_NODE и NEW_NODE означавају XML елемент пре и после операције,
- ако је тригер креиран на нивоу израза, променљиве OLD_NODE и NEW_NODE означавају секвенцу XML елемената пре и после операције.

Опционални израз XQuery-Let-Clause користи се за дефинисање променљивих чији опсег покрива и услов и акцију тригера.

Израз WHEN представља услов тригера, а ако се изостави, подразумева се да је услов тачан.

Акција тригера дата је у изразу DO и може бити произвољна комплексна операција ажурирања.

Оваква дефиниција тригера погодна је и за потребе ове дисертације, тако да је коришћена у даљем раду за спецификацију тригера за проверу ограничења у бази података.

Event-Condition-Action (ECA) правила користе се у базама података, а у раду [Bai02] испитано је увођење ECA правила у контексту XML структура података. Ова правила аутоматски извршавају акције које се спроводе када наступи одговарајући догађај, а наведени услов је задовољен. У раду аутори наводе да је у плану да се ECA правила користе за наведену функционалност и у XML базама података, пре свега за проверу ограничења кључа и осталих ограничења над XML документима, али и за опоравак базе, уколико је неко од ограничења нарушено. Дефинисан је језик за спецификацију ECA правила за XML.

Због полуструктуиране природе XML-а, разматране су и следеће теме:

- грануларност догађаја: док је у релационом моделу података грануларност јасно одређена, јер се догађаји за додавање, брисање и промену дешавају када се у релацију додаје нова торка, или се брише или мења постојећа торка, у XML-у не постоји овако јасна подела, а који део документа ће бити избрисан или где ће бити додат нови део документа, зависи од локације која је задата у оквиру XPath израза;
- грануларност акције: у релационом моделу података, додавање, брисање или промена могу се догодити само над торкама једне релације; у XML-у ове акције могу да

манипулишу целим подстаблом, а додавање или брисање дела документа може да изазове различите догађаје.

Овај језик разматран је и у радовима који су претходно наведени, а односе се на спецификацију упитног језика за XML.

У раду [Gri05] дата је дефиниција XML тригера, који је заснован на XQuery језику и претходно дефинисаном језику за ажурирање. Такође дате су и методе за подршку тригерима у XML базама података. Методе описане у раду имплементиране су у XML бази података Sedna, [Sedna].

Као резултати овог рада, наведени су: дефинисање XML тригера који имају семантику и синтаксу сличну семантици и синтакси SQL тригера, универзална метода за подршку тригера у XML системима за управљање базама података, методе за подршку тригера приликом ажурирања.

Дефинисањем тригера у XML базама података, аутори су наишли на неколико проблема. Пре свега, сви проблеми су се јавили због произвољне и хијерархијске структуре XML података, који су онемогућили да се једноставно усвоји концепт SQL тригера.

Како се тригери у релационим базама података користе за подршку ограничењима, аутори су предложили, као један од праваца даљег развоја, да се испитају могућности тригера за подршку разних врста ограничења у XML базама података.

Да тригери били направљени по угледу на тригере у релационим базама података, мора се узети у обзир хијерархијска структура XML докумената. У раду [Lan14] уведен је до тада нови концепт, опсег тригера, као и грануларност на нивоу путање. Грануларност тригера је битна да би се знало на који део документа утиче нека операција ажурирања.

Грануларност тригера је у радовима скоро комплетно преузета из релационог модела података. Оно што је у релационом моделу грануларност на новоу појединачне врсте или целе наредбе, у XML моделу података је грануларност на нивоу чвора или документа [Bai02]. Неки приступи говоре о грануларности на нивоу документа као о скуповној грануларности [Bon01a] или грануларности на нивоу упита [Bon02]. Неки радови, због једноставности, спомињу само грануларност на нивоу чвора [Rek05, Bon01b]. Иако се користе различити изрази, сви појмови се односе на исто: тригер на нивоу чвора је еквивалентан тригеру на нивоу врсте, а тригер на нивоу документа је еквивалентан тригеру на нивоу упита, у XML и релационим базама, редом. Док су аутори у свом претходном раду увели појам грануларности путање [Lan07], у овом раду први пут се спомиње грануларност путање која је повезана са опсегом тригера.

Једна од тема обрађиваних у неколико радова су и промељиве old и new (или old-node и new-node), које се односе на чворове на које је утицала нека операција, и то пре и после операције [Bon01a, Bon01b, Bon02]. Један од приступа описује само једну променљиву $delta [Bai02]. Ова променљива представља скуп чворова који су додати или избрисани једним једноставним XPath изразом. Може се користити и у услову и у акцији која се спроводи.

Тип променљиве зависи и од типа тригера. Слично као и у релационим базама података, уколико се тригер односи на операцију INSERT, постоји само променљива :new.

Опсег тригера је проширење контекста у оквиру којег се примењује тригер. До тада дефинисани тригери могли су бити примењени на цео документ и контекст је био цео документ. Такође, подршка за рад са више докумената за сада није могућа. То значи да тригер не може бити примењен на колекцију докумената. Проширење се односи на примену тригера на шему или на све документе у бази.

У доступној литератури везаној за тригере у XML базама података, сви тригери се односе на ограничења која потичу од неких правила пословања. Ограничења која представљају еквивалент ограничењима из релационог модела података до сада нису била имплементирана, што отвара простор за тему ове дисертације.

У неким радовима аутори посматрају податке из релационе базе података као XML погледе. Тако у раду [Sha05] аутори представљају програм у којем се тригери могу дефинисати над XML погледима релационих података. Технике за обраду тригера над XML погледима усклађене су са постојећом подршком за SQL тригере у комерцијалним релационим базама података. Упити над XML погледима релационих података специфицирани су у XQuery-ју. У раду [Sha01] обрађене су две теме. Прва се односи на дизајнирање окружења за обраду произвољних комплексних XQuery упита, укључујући и упите који садрже угњеждене изразе. Друга тема, која не спада у план разматрања ове дисертације, је презентација техника за ефикасну евалуацију XQuery упита над XML погледима над релационим подацима. Једна од приказаних техника је композиција XML погледа, у којој се елиминишу сви делови XML документа који се не јављају у коначном резултату упита. Још један од резултата овог рада је опис проширења система, које ће омогућити да се у релационим базама података обрађују велике класе XQuery упита.

Како је XML постао доминантни стандард за размену података на Интернету, настала је и потреба да се подаци из релационих база података представе у облику XML докумената. На тај начин, подаци могу да се чувају у релационим базама података, а да Интернет апликације приступају тим истим подацима у XML формату путем XML погледа. На пример, произвођач податке о својом производима чува у релационој бази података. Да би представио каталог производа купцима путем XML веб сервиса, произвођач може да креира XML поглед каталога производа и да тај поглед представи као веб сервис. Текући систем који су аутори користили је пасиван и може да подржи само упите које иницира сам корисник. То значи да постојећи систем омогућава купцу из претходног примера, само да експлицитно иницира захтев да претражи каталог производа. У овом раду аутори предлажу нов систем, који ће бити активан, који омогућава кориснику да специфицира тригере над XML погледима. У претходном примеру, то би значило да купац може да постави тригер који ће га обавештавати увек када се у каталог дода нови производ, или када се жељени производ распрода, без потребе да сваки пут понавља упит над XML погледом који ће детектовати промене.

Представљен је начин превођења тригера над XML погледима релационих података у SQL тригере, као и превођење операција ажурирања у релационим базама података у одговарајуће операције ажурирања у XML структурама.

Тригери и упити су механизми за валидацију ограничења. Ограничења дефинисана у овој дисертацији биће на примерима валидирана и путем XQuery функција и путем тригера у изабраној XML бази података.

2.11 Складиштење XML докумената и Native XML базе података

XML документи се могу складиштити на три начина: у фајл систему, у релационој бази података или у XML native бази података [Bou10].

Складиштење XML докумената у фајл систему је прихватљиво само за мали скуп докумената. Постоје алати за претраживање, ажурирање, идексирање и трансакциону обраду података. Документима се управља помоћу неког API-ја, на пример, DOM API [DOM].

Складиштење у релациону базу података нуди низ погодности које ове базе имају над фајл системом: трансакције, вишекориснички приступ, безбедност, али, како су XML и релациони модел суштински различити, нема додатних предности. XML документи су хијерархијски, док су у релационом моделу подаци смештени у више табела. Чворови XML документа имају елементе и/или атрибуте, док обележја у табелама имају само једну вредност. Елементи могу бити угњеждени, док су вредности обележја недељиве. Шема не мора да постоји, док је у релационом моделу обавезна. Ако шема постоји, елементи имају тачно утврђен редослед, а редослед врста и колона у табели није битан. XML документи се у релациону базу података могу сместити на три начина: CLOB (Character Large Object), чисто релациони приступ, и XML enabled релационе базе података.

XML native базе података као основну логичку јединицу имају XML документ и подацима манипулишу користећи XML стандарде. То значи да осим смештаја, трансакција, вишекорисничког приступа и безбедности, ове базе података управљају подацима у њиховом изворном облику, а не трансформишу их у релациони или други облик. Упитни језици XPath и XQuery су стандардни језици у XML native базама података.

Најпознатије имплементације *native* XML система за управљање базама података су: eXist, Sedna, BaseX, MarkLogic, и друге, мада данас сви велики произвођачи нуде подршку за XML.

Sedna [Sedna, Gri06] је *native* XML СУБП који пружа велики број основних сервиса: упити, ажурирање, складиштење, ACID трансакције, оптимизација упита, итд. Подржава W3C XQuery имплементацију. У наведеном раду описан је физички слој Sedna XML система за управљање базом података.

Sedna је јавно доступна на сајту [Sedna] и примери који су направљени у овој дисертацији, тестирани су у овој бази података. Sedna омогућава извршавање XQuery упита и тригера, као и складиштење XML докумената.

eXist [eXist] је још један *native* XML СУБП. Омогућава складиштење XML докумената, као и постављање упита помоћу XQuery-ја. Тригери се могу дефинисати и путем програмског језика Java, и путем посебних израза који садрже XQuery упите. Међутим, приликом извршавања, тригер утиче на цео документ, па је потребно поредити изглед и садржај документа пре и после операције. То је веома компликовано, ако је документ обиман.

BaseX [BaseX] је *native* XML СУБП који пружа потпуну подршку за XQuery упите. Има клијент/сервер архитектуру, подршку за ACID трансакције и управљање корисницима. Међутим, овај систем не подржава имплементацију тригера, па није даље разматран.

Због свега наведеног, тригери су имплементирани у XML бази података Sedna.

2.12 Структуирање XML документа

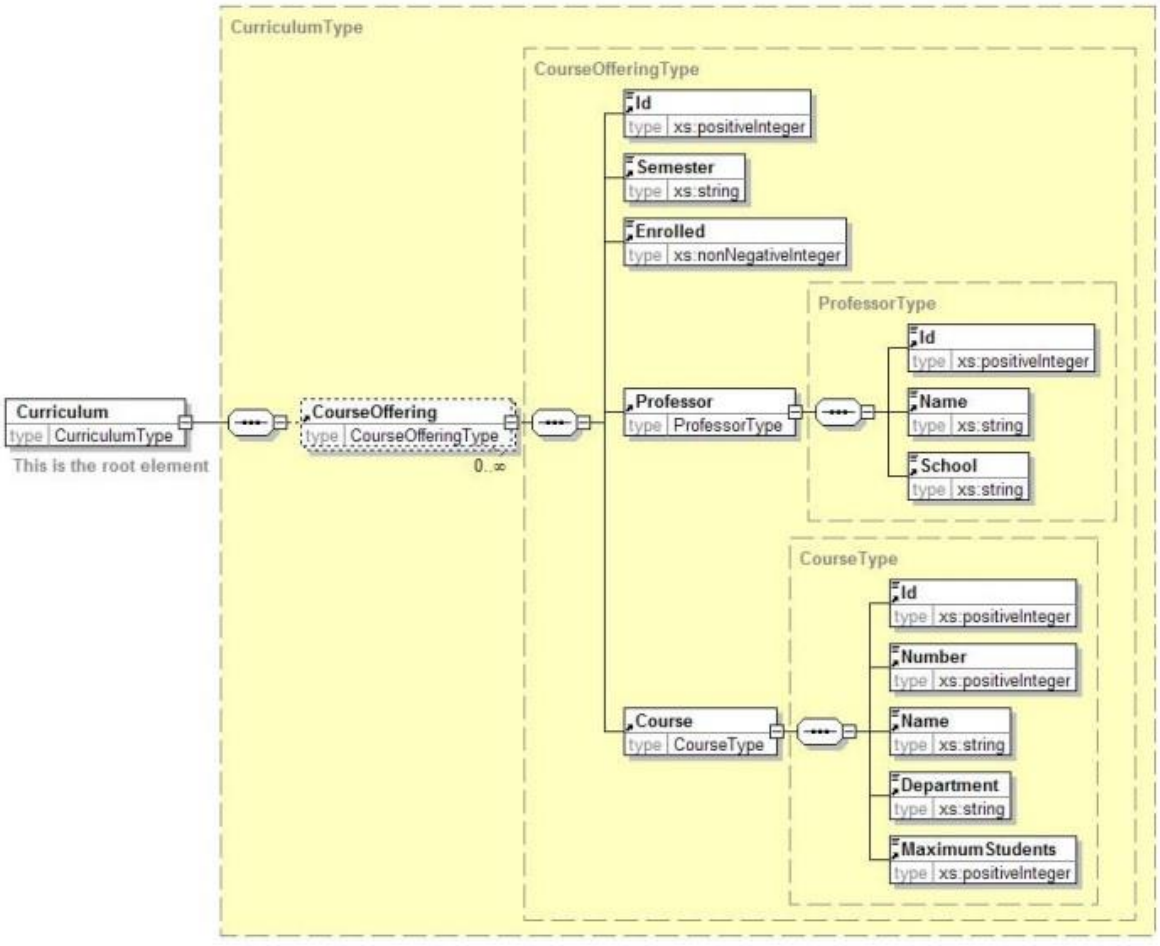
Структура XML документа је потпуно произвољна. Сваки XML документ мора да буде добро формиран, а ако је направљен у складу са неким DTD-ом или XML Schema документом, онда мора бити и валидан. Структуирање елемената испод коренског требало би да укаже на зависност једног елемента од надређеног. Слично графу затварања у релационом моделу [Mog00], елементи који су подређени неком елементу, зависе од елемента који се налази изнад њих у хијерархији.

У раду, [Nei06] разматрано је којим приступом треба формирати XML шему, хијерархијским или релационим, а тако да одговара и W3C XML Schema спецификацији и потребама и

правилима агенције EPA. У овом извештају, дата су два типа XML Schema докумената, хијерархијски и релациони.

Појам хијерархијски односи се на шеме код којих је однос између ентитета реализован само преко угњеждених структура. Веза између елемената у XML документу дефинисана је њиховом физичком локацијом унутар документа. У случају када је веза између елемената са кардиналитетом „више-више“, долази до понављања одређених елемента, што доводи до редундантности. У примеру који је наведен у раду, а овде је дат на Слика 2 и у Листинг 13, за сваку информацију о држању наставе морају бити наведени сви подаци о предмету, као и подаци о професору који га предаје. Ту долази до понављања информација, то јест до редундансе података.

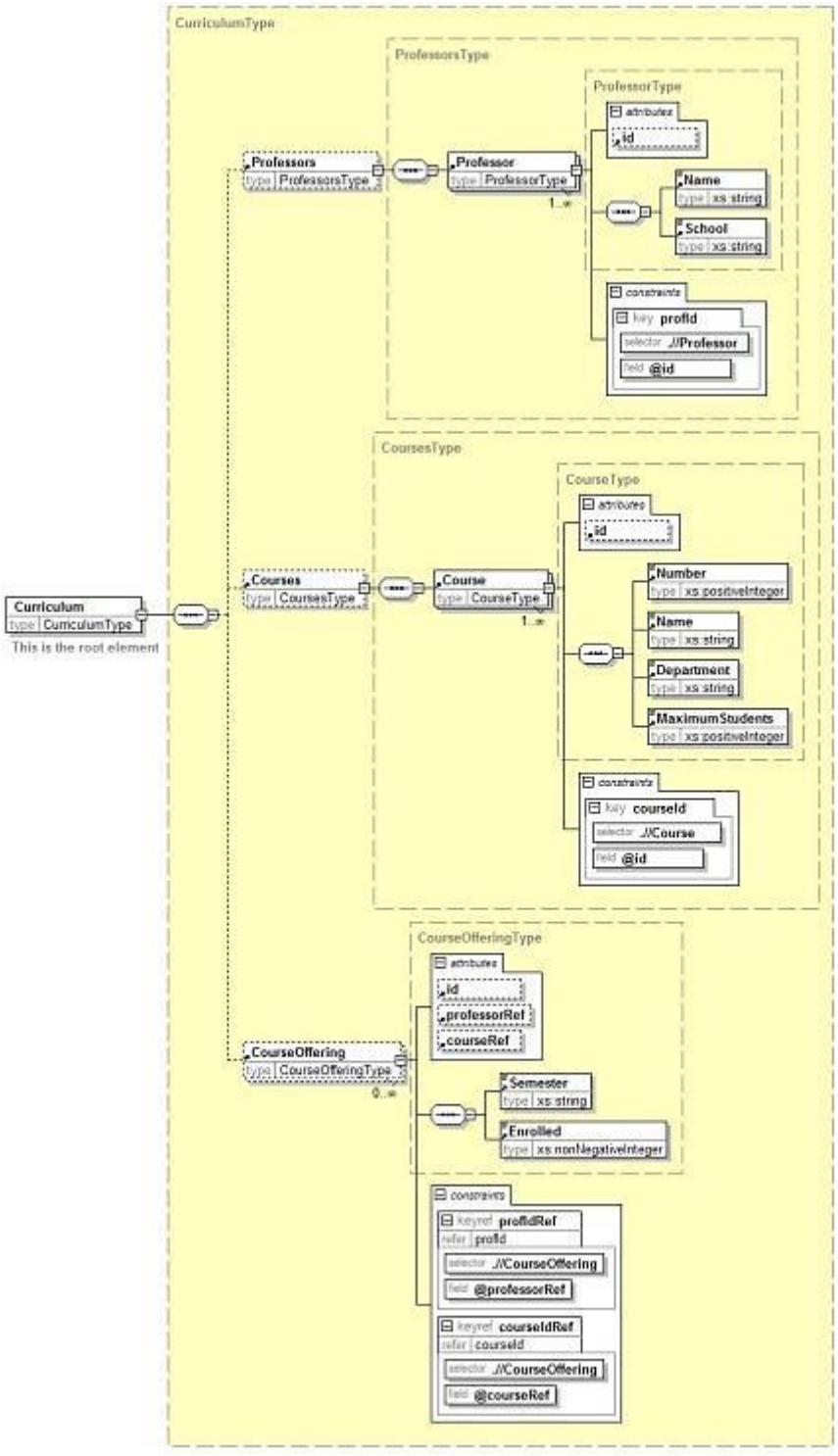
Када је у питању релациони приступ, XML Schema документ се формира тако што се за везе између елемената не користи угњеждавање, него кључеви и страни кључеви. Сваки елемент има свој идентификатор, који се касније користи за референцирање у другим елементима помоћу key и keyref елемената. У примеру датом у раду, а овде на Слика 3 и Листинг 14, професори и предмети су дефинисани посебним елементима, а њихови кључеви се прослеђују као референце у елемент CourseOffering, који садржи податке о томе који професор предаје који предмет. Професори и предмети представљени су само преко својих кључева, а сви остали подаци налазе се само на једном месту.



Слика 2 XML Schema документ Curriculum формиран хијерархијским приступом

```
<?xml version="1.0" encoding="UTF-8"?>
<Curriculum xmlns:xsi="http:/ www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ecos_h.xsd">
<CourseOffering>
 <Id>1</Id>
  <Semester>Fall</Semester>
 <Enrolled>128</Enrolled>
 <Professor>
  <Id>1</Id>
  <Name>John Brown</Name>
  <School>UNC</School>
 </Professor>
  <Course>
  <Id>3</Id>
  <Number>150</Number>
  <Name>Biology I</Name>
  <Department>Health Sciences</Department>
  <MaximumStudents>150</MaximumStudents>
 </Course>
 </CourseOffering>
 <CourseOffering>
 <Id>2</Id>
 <Semester>Fall</Semester>
 <Enrolled>25</Enrolled>
 <Professor>
  <Id>1</Id>
  <Name>John Brown</Name>
  <School>UNC</School>
 </Professor>
 <Course>
  <Id>1</Id>4
  <Number>114</Number>
  <Name>Computer Science I</Name>
  <Department>Computer Science</Department>
  <MaximumStudents>30</MaximumStudents>
 </Course>
</CourseOffering>
</Curriculum>
```

Листинг 13 Део XML документа Curriculum формиран хијерархијским приступом



Слика 3 XML Schema документ Curriculum формиран релационим приступом

```
<?xml version="1.0" encoding="UTF-8"?>
<Curriculum xmlns:xsi="http:/ www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ecos_r.xsd">
<Professors>
 <Professor id="1">
  <Name>John Brown</Name>
  <School>UNC</School>
 </Professor>
 <Professor id="2">
  <Name>Elanor Fisher</Name>
  <School>Harvard</School>
```

```
 </Professor>
 <Professor id="3">
  <Name>David Darwin</Name>
  <School>Yale</School>
 </Professor>
</Professors>
<Courses>
 <Course id="1">
  <Number>114</Number>
  <Name>Computer Science I</Name>
  <Department>Computer Science</Department>
  <MaximumStudents>30</MaximumStudents>
 </Course>
 <Course id="2">
  <Number>214</Number>
  <Name>Computer Science II</Name>
  <Department>Computer Science</Department>
  <MaximumStudents>30</MaximumStudents>
 </Course>
</Courses>
<CourseOffering id="1" professorRef="1" courseRef="2">
 <Semester>Fall</Semester>
 <Enrolled>128</Enrolled>
</CourseOffering>
<CourseOffering id="2" professorRef="1" courseRef="1">
 <Semester>Fall</Semester>
 <Enrolled>25</Enrolled>
</CourseOffering>
<CourseOffering id="3" professorRef="3" courseRef="1">
 <Semester>Fall</Semester>
 <Enrolled>25</Enrolled>
</CourseOffering>
</Curriculum>
```

Листинг 14 Део XML документа Curriculum формиран релационим приступом

Оба приступа имају и предности и мане. Који од њих ће бити примењен зависи од конкретних захтева самог система. Један од критеријума је интегритет података, при чему се проверава да ли елементи могу бити јединствено идентификовани, да ли важи ограничење референцијалног интегритета, и да ли постоје аномалије ажурирања. Како нормализација није применљива на XML документ који је валидан у односу на хијерархијску шему, не може бити примењено ни ограничење референцијалног интегритета. Без нормализације, не постоје ни везе „родитељ-дете“ базиране на кључу. Могу да постоје само физичке везе „родитељ-дете“.

У XML документу формираном на основу релационе шеме, користе се атрибути key и keyref у циљу обезбеђења јединствености елемента у документу. Овај механизам већ приликом валидације проверава да ли је вредност кључа јединствена. Такође, и ограничење референцијалног интегритета се проверава приликом валидације, јер свака вредност атрибута у keyref елементу мора да одговара некој постојећој вредности атрибута у key елементу.

Неки од закључака из овог рада су да је релациони начин формирања XML документа бољи уколико је потребно подржати ограничење референцијалног интегритета и јединственост вредности кључа. Документи формирани на овај начин су мањи, избегава се редундантност података, али хијерархијски начин формирања XML документа омогућава прегледност и читљивост документа.

Подаци у XML документу могу се структуирати на три начина:

- сви подаци се придружују елементу,
- сви подаци се придружују искључиво атрибутима елемента и
- неки подаци се придружују елементу, а остали атрибутима.

Не постоји прецизно дефинисано правило за формирање XML докумената. Такође, приликом трансформације шеме базе података у XML документ потребно је одлучити:

- да ли ће елементи који представљају шеме релације које зависе од неких других шема релација, бити поделементи својих надређених елемената, а са осталим елементима бити повезани путем референци [Jum10],
- да ли ће постојати један коренски елемент који представља шему базе података под којим ће се, на истом нивоу хијерархије, налазити поделементи који представљају шеме релација и
- да ли ће обележја шеме релације бити представљена као атрибути или поделементи елемента који представља шему релације.

У овој дисертацији усвојено је правило да свака шема релације из релационог модела буде представљена као елемент, а да обележја шема релација буду представљена као атрибути.

Уколико је XML документ представљен као стабло у којем везе између надређеног и подређеног елемента одређују и зависност подређеног од надређеног елемента, онда се у подређеном елементу не појављују атрибути који представљају стране кључеве. Веза ''родитељ-дете'' представљена је само физичким позиционирањем.

XML документ се може приказати у облику графа, који је сличан графу затварања у релационом моделу података [Mog00]. Гране оваквог графа представљају референцијалне интегритете који постоје између елемената у XML документу. Чворови графа су елементи XML документа. Између надређеног и подређеног елемента постоји ограничење референцијалног интегритета. Уколико подређени елемент зависи од још неког елемента, потребно је тај елемент референцирати.

Елемент Dete идентификационо зависи од елемента Roditelj. Ако се направи хијерархијска структура (Листинг 15), елемент Dete се налази испод елемента Roditelj. Позиција елемента Dete указује на зависност тог елемента од надређеног елемента. Кључ елемента Dete би требало да буде id_deteta+mbr. Атрибуту mbr, који се налази само у елементу Roditelj, није могуће приступити из елемента Dete, али се физичким позиционирањем одређује идентификациона зависност.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
     <xs:element name="Koren">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Roditelj" maxOccurs="unbounded">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="Dete" maxOccurs="unbounded">
        <xs:complexType>
         <xs:attribute name="id_deteta" type="xs:string" use="required"/>
         <xs:attribute name="ime" type="xs:string" use="required"/>
        </xs:complexType>
       </xs:element>
      </xs:sequence>
      <xs:attribute name="mbr" type="xs:string" use="required"/>
      <xs:attribute name="ime" type="xs:string" use="required"/>
      <xs:attribute name="prezime" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:key name="Roditelj_PK">
   <xs:selector xpath="Roditelj"/>
```

```
    <xs:field xpath="@mbr"/>
   </xs:key>
   <!—posebno izdvojeno za komenatre-->
  </xs:element>
</xs:schema>
```

Листинг 15 XML Schema са елементима Roditelj и Dete

Ако се кључ елемента Dete дефинише на следећи начин (Листинг 16):

```
<xs:key name="Dete_PK">
 <xs:selector xpath="Dete"/>
 <xs:field xpath="@id_deteta"/>
</xs:key>
```

Листинг 16 Релативни кључ елемента Dete

тада је Dete_PK релативни кључ и сваки атрибут id_deteta јединствен је само у оквиру свог надређеног елемента Roditelj.

XML документ који је формиран у односу на ову шему дат је у Листинг 17.

```
<?xml version="1.0" encoding="UTF-8"?>
<Koren xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="RoditeljDete-nonID.xsd">
 <Roditelj prezime="Peric" mbr="R1" ime="Pera">
  <Dete id_deteta="D1" ime="Maja"/>
  <Dete id_deteta="D2" ime="Ana"/>
 </Roditelj>
 <Roditelj prezime="Maric" mbr="R2" ime="Mara">
  <Dete id_deteta="D1" ime="Marko"/>
 </Roditelj>
 <Roditelj prezime="Ilic" mbr="R3" ime="Ilija">
  <Dete id_deteta="D1" ime="Ivan"></Dete>
 </Roditelj>
 <Roditelj prezime="Antic" mbr="R4" ime="Ana">
  <Dete id_deteta="D1" ime="Aca"></Dete>
  <Dete id_deteta="D2" ime="Maja"></Dete>
 </Roditelj>
</Koren>
```

Листинг 17 XML документ са елементима Roditelj и Dete

Оваква хијерархијска структура је погодна за формирање идентификационо зависних елемената. Међутим, ако би било потребно направити ограничење инверзног референцијалног интегритета, потребан је атрибут у елементу Dete на који би био референциран кључ из надређеног елемента Roditelj. У оваквом примеру то није могуће.

Претпоставимо да дете не зависи идентификационо од родитеља и да се свако дете може идентификовати само преко свог атрибута id_deteta. Тада свака вредност атрибута id_deteta мора бити јединствена у оквиру целог XML документа. Потребно је направити апсолутни кључ. Свако дете припада само једном родитељу, на пример, раднику у некој фирми, ради здравственог осигурања. И у овом случају могуће је направити хијерархијски однос између елемената, али примарни кључ ечемента дете мора означавати јединствене вредности у целом документу. Ако је путања xpath у елементу xs:selector израз Roditelj/Dete, онда је опсег важења тог кључа цео документ и не могу се понављати вредности атрибута id_deteta. Овакав примарни кључ приказан је у Листинг 18.

```
<xs:key name="Dete_PK">
   <xs:selector xpath="Roditelj/Dete"/>
   <xs:field xpath="@id_deteta"/>
</xs:key>
```

Листинг 18 Апсолутни кључ елемента Dete

XML документ који задовољава овако дефинисан кључ приказан је у Листинг 19. Ни у оваквом примеру није могуће дефинисати ограничење инверзног референцијалног интегритета, јер не постоји страни кључ у подређеном елементу.

```
<?xml version="1.0" encoding="UTF-8"?>
<Koren xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="RoditeljDete.xsd">
 <Roditelj prezime="Peric" mbr="R1" ime="Pera">
  <Dete id_deteta="D1" ime="Ana"/>
  <Dete id_deteta="D2" ime="Maja"/>
 </Roditelj>
 <Roditelj prezime="Savic" mbr="R2" ime="Sava">
  <Dete id_deteta="D3" ime="Tanja"/>
  <Dete id_deteta="D4" ime="Sanja"/>
 </Roditelj>
</Koren>
```

Листинг 19 XML документ са елементима Roditelj и Dete

Уколико неки елемент треба да има више страних кључева, из више елемената, није довољно само физичко позиционирање у хијерархијској структури, него морају да постоје атрибути, који ће означавати те стране кључеве. Једна од веза би могла бити представљена физичким позиционирањем, али за остале стране кључеве морају постојати атрибути који их представљају у подређеном елементу. Међутим, да би сви документи били формирани на исти начин, и овде ће подређени елемент садржати атрибуте који представљају стране кључеве, а ограничење страног кључа биће специфицирано keyref елементима. У Листинг 20 дата је шема са коренским елементом Fakultet. Елемент Ispit има два страна кључа, која су представљена атрибутима BRI и SPR.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Fakultet">
  <xs:annotation>
   <xs:documentation>Comment describing your root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Student" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="BRI" type="xs:string" use="required"/>
      <xs:attribute name="IME" type="xs:string" use="required"/>
      <xs:attribute name="PREZIME" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Predmet" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="SPR" type="xs:string" use="required"/>
      <xs:attribute name="NAZIV" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Ispit" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="BRI" type="xs:string" use="required"/>
      <xs:attribute name="SPR" type="xs:string" use="required"/>
      <xs:attribute name="DATUM" type="xs:date" use="required"/>
      <xs:attribute name="OCENA" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:integer">
         <xs:minInclusive value="5"/>
         <xs:maxInclusive value="10"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
     </xs:complexType>
```

```
    </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:key name="Student_PK">
   <xs:selector xpath="Student"/>
   <xs:field xpath="@BRI"/>
  </xs:key>
  <xs:key name="Predmet_PK">
   <xs:selector xpath="Predmet"/>
   <xs:field xpath="@SPR"/>
  </xs:key>
  <xs:key name="Ispit_PK">
   <xs:selector xpath="Ispit"/>
   <xs:field xpath="@BRI"/>
   <xs:field xpath="@SPR"/>
  </xs:key>
  <xs:keyref refer="Student_PK" name="Ispit_FK1">
   <xs:selector xpath="Ispit"/>
   <xs:field xpath="@BRI"/>
  </xs:keyref>
  <xs:keyref refer="Predmet_PK" name="Ispit_FK2">
   <xs:selector xpath="Ispit"/>
   <xs:field xpath="@SPR"/>
  </xs:keyref>
 </xs:element>
</xs:schema>
```

Листинг 20 XML Schema Fakultet

У Листинг 21 дат је XML документ који је валидан у односу на шему дату у Листинг 20.

```
<?xml version="1.0" encoding="UTF-8"?>
<Fakultet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="ispiti.xsd">
 <Student BRI="s1" IME="Ana" PREZIME="Vasic"/>
 <Student BRI="s2" IME="Iva" PREZIME="Maric"/>
 <Predmet SPR="p1" NAZIV="Baze podataka 1"/>
 <Predmet SPR="p2" NAZIV="Baze podataka 2"/>
 <Ispit SPR="p1" BRI="s1" DATUM="2014-06-05" OCENA="10"/>
 <Ispit SPR="p1" BRI="s2" DATUM="2014-06-21" OCENA="9"/>
 <Ispit SPR="p2" BRI="s2" DATUM="2014-09-05" OCENA="9"/>
</Fakultet>
```

Листинг 21 XML документ Fakultet

За нека ограничења која постоје у релационом моделу, а у овој дисертацији су специфицирана и за XML модел података, неопходно је да и у подређеном елементу постоји атрибут који представља страни кључ. Због тога, а и због смањења редундантности, у овој дисертацији је усвојено правило да се сви елементи налазе на истом нивоу хијерархије испод корена, као што је описано у [Cha06]. Уколико између неких елемената треба да постоји веза са значењем ''родитељ-дете'', онда у подређеном елементу мора да постоји атрибут који представља страни кључ. Ипак, хијерархијску структуру могуће је користити за потребе овог рада, али подређени елементи увек морају да садрже атрибуте који означавају стране кључеве.

За спецификацију неких ограничења, као што су ограничење домена, ограничење торке, ово правило није битно, али да би сви примери били формирани на сличан начин, правило је примењено на све.

2.13 Закључак

У овом поглављу дат је преглед стања у области истраживања. Приказани су типови ограничења у релационом моделу података у одељку 2.2.1, као и типови ограничења у XML моделу података у одељку 2.2.2. Описани су типови ограничења у XML Schema језику, а затим

је у одељцима 2.5, 2.6 и 2.7 дат преглед типова ограничења, до сада описаних у радовима из области истраживања. Како су у дисертацији коришћени и тригери за имплементацију неких типова ограничења, у одељку 2.10 дат је и преглед до сада предложених имплементација тригера. Такође, у одељку 2.11 проучено је и неколико јавно доступних XML система за управљање базама података, од којих је један, Sedna, изабран за имплементацију ограничења обрађених у овој дисертацији помоћу тригера, док је eXist изабран за имплементацију ограничења употребом XQuery функција.

У великом броју радова, чији је преглед дат у овом поглављу, посвећена је пажња дефиницији апсолутног и релативног кључа, који проширују дефиницију кључа из DTD-а, а касније из XML Schema-е. На основу дефиниције кључа у тим радовима, ограничење које је затим једино обрађивано је ограничење референцијалног интегритета. Нису обрађени типови ограничења који постоје у релационом моделу, а јављају се у пракси, као што су ограничење инверзног референцијалног интегритета и проширено ограничење референцијалног интегритета. У овој дисертацији ова два ограничења су идентификована, дата је њихова спецификација у поглављу 3, и реализован код који врши њихову валидацију, у поглављу 5. Радови који описују апсолутни и релативни кључ, као и начин структуирања XML документа, представљају основу за спецификацију типова ограничења, која ће бити приказана у овој дисертацији. Након разматрања наведене две дефиниције кључа, а на основу усвојене структуре XML документа, кључеви који су употребљени у овој дисертацији су апсолутни кључеви. Увођење типова ограничења који постоје у релационом моделу података, а до сада нису били дефинисани у XML моделу података, не омогућава употребу релативних кључева, јер се страни кључ мора навести, уколико је то потребно.

За проширено ограничење торке постоје радови који су увели гранање и петље у језик за валидацију. Проблем са оваквим приступом је што тако дефинисан језик за валидацију није довољно моћан као регуларан програмски језик, попут XQuery-ја. Такође, гранања и петље представљају процедурални начин за исказивање логичког услова ограничења, док XQuery пружа могућност да се логички услови искажу на функционалан начин. Сваки покушај проширивања језика за валидацију беспотребно компликује такав језик, а заправо већ постоји специјализован језик за XML модел података - XQuery. У овој дисертацији код који валидира проширено ограничење торке задат је у XQuery-ју.

Ограничења домена, вредности атрибута и јединствене вредности постоје и у релационом моделу података и у XML моделу података. Ограничења торке, кључа и референцијалног интегритета јесу специфицирана у XML моделу података и на одређени начин јесу имплементирана у постојећим XML СУБП. Ипак, неке сложеније провере, као и акције које се спроводе приликом валидације ових ограничења нису специфициране у XML моделу података, нити имплементиране у XML СУБП, што је такође оставило простор за истраживање у овој дисертацији.

На основу типизације ограничења у релационом моделу података, описане у одељку 2.2.1, дата је типизација ограничења у XML моделу података, описана у поглављу 3.

3 Типови ограничења у XML моделу података

По угледу на ограничења која су дефинисана у релационом моделу података, у овој дисертацији дат је предлог следећих типова ограничења у XML моделу података:

- ограничење домена,
- ограничење торке,
- проширено ограничење торке,
- ограничење вредности атрибута,
- ограничења кључа,
- ограничење јединствене вредности,
- ограничење референцијалног интегритета,
- ограничење инверзног референцијалног интегритета, и
- проширено ограничење референцијалног интегритета.

Неки од ових типова ограничења већ су дефинисани у XML моделу података, о чему је писано у поглављу 2 (Тренутно стање у области истраживања), а за нека је потребно извршити одређена проширења у XML Schema-и. Нека до сада нису дефинисана и приказана су у овој дисертацији. Сва ограничења су дефинисана за XML Schema-у, за разлику од већине до сада објављених радова, у којима су ограничења дефинисана у присуству DTD-а.

3.1 Тип ограничења у XML моделу података

На основу доступне литературе везане за таксономије ограничења у XML моделу података, а највише [Rod07], као и спецификације типа ограничења датог у [Luk07a], усвојени су неки предлози типизације ограничења у XML моделу података. Предлажу се следеће карактеристике типа ограничења у XML моделу података:

- област дефинисаности – TOd, која садржи информације о објектима који се користе у спецификацији ограничења и над којима се дефинише ограничење,
- област интерпретације – TOi, која садржи информације о објектима над којима се ограничење интерпретира,
- формализам за записивање (дефиницију) – TFz, који специфицира формулу за записивање конкретног типа ограничења и
- правило за интерпретацију (валидацију) – TPi, које дефинише када је ограничење задовољено.

Тип ограничења је, у складу са наведеним карактеристикама, дефинисан на следећи начин:

$$TipOgraničenja(TOd, TOi, TFz, TPi)$$

Област дефинисаности TOd може бити један елемент или више елемената, али може бити и без елемената, ако је у питању ограничење домена.

$$TOd \in \{1, n, 0\}$$

Према области интерпретације TOi, тип ограничења може бити тип ограничења вредности, тип ограничења торке, тип ограничења елемента, или тип ограничења између више елемената.

$$TOi \in \{v, t, e, me\}$$

Формализам за записивање TFz даје граматику за записивање ограничења датог типа.

Правило за интерпретацију TPi даје формулу која дефинише када је ограничење задовољено. Ограничење се имплементира као задовољено или није задовољено.

Спецификација конкретног ограничења у XML моделу података, дата је на следећи начин:

$$OgrNaziv(OgrTip, OgrF, T)$$

где ознаке имају следећа значења:

- OgrNaziv назив конкретног ограничења,
- OgrTip је ознака типа ограничења и може имати једну од следећих вредности {DomCon, TupleCon, ExTupleCon, AttValCon, KeyCon, UnCon, RICon, IRICon, ExRICon},
- OgrF је формула за запис конкретног ограничења, написана помоћу синтаксе задате путем правила TFz,
- T је логичка структура над којом се дефинише конкретно ограничење, уз коју се наводе критичне операције које могу довести до нарушавања ограничења, као и могуће акције којима се обезбеђује очување валидности базе података.

Логичка структура T за конкретно ограничење садржи називе елемената који учествују у ограничењу, улогу сваког од елемената, као и критичне операције и могуће акције за сваку од тих улога. Дефиниција структуре дата је на следећи начин:

$$(E_i, r_i, \{(op_{ij}, act_{ij})\} | j \geq 1)$$

где ознаке имају следећа значења:

- E_i назив елемента који учествује у ограничењу,
- r_i је улога тог елемента у ограничењу,
- op_i је критична операција за дату улогу и може имати вредности {insert, update, delete},
- act_i је одабрана акција из одговарајућег скупа могућих акција {cascade, restrict}
    - cascade – акција се спроводи каскадно и на подређене елементе,
    - restrict – акција се забрањује.

Слично дефиницији DTD стуктуре датих у [Fan00, Bun01, Are02a], предлаже се дефиниција XML структуре. XML шема са ограничењима може се посматрати као пар (S, Σ), где је S структура шеме, а Σ скуп ограничења, и дата је на следећи начин:

$$S_{con} = (S, \Sigma)$$

Опсег важења свих типова ограничења је цео документ, јер је таква структура XML документа усвојена у овој дисертацији.

Скуп ограничења Σ садржи ограничења оних типова који су дефинисани у овој дисертацији, а који су елементи скупа типова ограничења {DomCon, TupleCon, ExTupleCon, AttValCon, KeyCon, UniqueCon, RICon, IRICon, ExRICon}.

XML документ је валидан у односу на XML шему са ограничењима, ако и само ако задовољава сва правила дата у XML шеми и задовољава сва ограничења дата у Σ.

Уводе се и следеће ознаке:

- E – тип елемента,
- e – елемент типа E,
- e@X – вредност (скупа) атрибута X у елементу e.

Такође, дефинише се оператор спајања елемената, слично као у релационом моделу [Mog96, Dat03].

Дати су типови елемената E_1 и E_2. Посматрајмо скуп атрибута $X \in E_1$ и скуп атрибута $Y \in E_2$. Спој елемената e_1, који је типа E_1, и e_2, који је типа E_2, по атрибутима X и Y је функција $\bowtie: E_1 x E_2 \rightarrow E$, где је E нови тип елемента, дефинисана на следећи начин:

$$e_1 \bowtie e_2 = \{e \in root \mid e@X \in e_1 \wedge e@Y \in e_2 \text{ i } X, Y \in E\},$$

где је root коренски елемент XML документа. Спој елемената је скуп нових елемената e, типа E, који су подређени коренском елементу, и садрже скупове атрибута X и Y, који припадају, редом, типовима елемената E_1 и E_2.

У поглављу 5 дати су примери модификованих XML Schema на основу којих је генерисан код који реализује наведена ограничења.

3.1.1 Ограничење домена

Ограничење домена дефинише се за сваки атрибут XML документа. Наводи се тип податка, дужина, и опционално, услов који вредности из тог домена треба да задовоље. Формализам за запис овог ограничења дат је на следећи начин:

$$id(D) = (Tip, Dužina, Uslov)$$

То је ограничење вредности које се дефинише без елемената. Интерпретира се за сваку могућу вредност, за коју се проверава тип податка, дужина типа податка, и додатни услов који мора да задовољи. Како ниједан елемент не учествује у овом ограничењу, T је Ø. Ограничење домена је, према спецификацији датој у одељку 3.1, записано на следећи начин:

DomCon(0,
v,
id(D),
id(D)(d)=(Tip, Dužina, Uslov)(d)=Tip(d) ∧ Dužina(d) ∧ Uslov(d),
Ø)

Ограничење домена је у целини подржано XML Schema-ом, а преглед свих могућности за реализацију ограничења домена, дат је у одељку 5.1.1.

3.1.2 Ограничење вредности атрибута

Ово ограничење дефинише се у оквиру једног типа елемента, за сваки атрибут. Поред услова дефинисаних у ограничењу домена тог атрибута, наводи се и да ли су за атрибут дозвољене или забрањене нула вредности. Формула за записивање ограничења вредности обележја дата је на следећи начин:

$$\tau(E, A) = (id(D), NullSpec)$$

Ово ограничење је ограничење вредности, и интерпретира се за сваку могућу вредност атрибута d. Дефиниција ограничења AttValCon описана је у следећем реду:

AttValCon(1,
v,
τ(E, A),
τ(E, A)(d) = (id(D), NullSpec)(d) = id(D)(d) ∧ NullSpec(d),
(-, nebitna uloga, {(insert, Restrict), (update, Restrict)}))

Ограничење вредности обележја није подржано XML Schema-ом на начин како је то обезбеђено у релационом моделу података. Нула вредности не могу се доделити атрибутима, већ само елементима. Као еквивалент вредности *null*, у XML Schema-и може се атрибуту use доделити вредност optional, чиме би било дозвољено да се вредност атрибута не наведе. Имплементација на конкретном примеру дата је у одељку 5.1.2. Експлицитно навођење вредности *null* је могуће за елементе, када се елементу додаје атрибут *xsi:nillable=true*.

3.1.3 Ограничење торке

Ограничење торке се дефинише за сваки тип елемента. За сваки атрибут у типу елемента може се дефинисати додатни услов Con(E), који вредности тог атрибута морају да задовоље. Формула за записивање овог ограничења дата је на следећи начин:

$$\tau(E) = (\{\tau(E, A) \mid A \in E\}, Con(E))$$

Ово ограничење се дефинише за један тип елемента и за све његове атрибуте. То је ограничење торке, па је област интерпретације торка, то јест један конкретан елемент одређеног типа. За сваки атрибут у том елементу треба проверити да ли задовољава ограничење вредности атрибута, као и додатни услов који је задат у ограничењу торке. Додатни услов повезује вредности атрибута у торци, тако што вредност једног атрибута зависи од вредности једног или више других атрибута. Улога елемента није битна, а приликом уноса новог елемента или модификације постојећег елемента, треба проверити да ли нове вредности такође задовољавају ограничење торке. Дефиниција ограничења TupleCon дата је на следећи начин:

TupleCon(1,
t,
τ(E),
τ(E)(e) = ({τ(E, A) | A∈E}, Con(E))(e) = (∀A∈E)(τ(E, A)@A ∧ Con(E)(e)),
(e, nebitna uloga, {(insert, Restrict), (update, Restrict)}))

На основу дефиниције ограничења, прелаже се проширење XML Schema-е, као што је дато у Листинг 22.

```
<xc:constraint xmlns:xc="http://www.doktorat.org/constraints"
 type="xc:tupleCon">
 <xc:condition from="<xPath>" function="<naziv_funkcije>"/>
</xc:constraint>
```

Листинг 22 Проширење шеме за ограничења торке

У XML Schema-и могуће је ограничење задати регуларним изразом, али велики број ограничења из праксе није могуће описати на тај начин. Не постоји могућност да се приликом провере ограничења користе гранање и итерација. Проширење шеме које је овде предложено почиње елементом xc:constraint. Вредност xc:tupleCon атрибута type означава да је у питању дефиниција ограничења торке. Елемент xc:condition у том случају садржи следеће атрибуте: from и function. Атрибут from садржи XPath израз којим се селектује одговарајући чвор у XML документу. Атрибут function означава XQuery функцију која треба да буде позвана, а тој

функцији ће као аргумент бити прослеђена торка. Функција враћа *boolean* вредност, где *true* означава да је услов задовољен, а *false* да услов није задовољен.

Приликом провере ограничења торке, за нови елемент који треба да се унесе или модификује, мора се проверити да ли задовољава услов, што се реализује позивом функције наведене у атрибуту function елемента xc:condition. Ако је услов задовољен, унос или модификација се дозвољава, иначе се забрањује. Синтакса XQuery језика не дозвољава вредност *null* у логичким изразима, тако да није могуће у потпуности повући паралелу са релационим моделом података. Псеудокод за проверу ограничења торке дат је у Листинг 23, а провера модификације је аналогна провери која је описана у овом листингу.

```
ulaz: novi element e∈E
BEGIN unos
  IF(uslov(e))
  THEN dozvoli unos
  ELSE zabrani unos
END
```

Листинг 23 Псеудокод за проверу ограничења торке приликом уноса новог елемента

3.1.4 Проширено ограничење торке

Проширено ограничење торке дефинише се за више типова елемената. Атрибути тих типова елемената треба да буду повезани како би задовољили задати услов. Формула за записивање овог ограничења је:

$$\tau_{ex}(E1 \bowtie E2)=\text{Constraint}(E1 \bowtie E2)\text{: Condition.}$$

Ово ограничење је ограничење торке, јер се за сваки конкретан елемент проверава да ли задовољава задати услов. Логички услов Condition се дефинише над скупом атрибута који припадају унији атрибута елемената који учествују у овом ограничењу. Улоге повезаних елемената нису битне. Проширено ограничење торке је записано на следећи начин:

ExTupleCon(n,
t,
$\tau_{ex}(E1 \bowtie E2)=\text{Constraint}(E1 \bowtie E2)$:Condition,
$\tau_{ex}(E1 \bowtie E2)(e)=\text{Constraint}(E1 \bowtie E2)(e)$,
(e, nebitna uloga, {(insert, Restrict), (update, Restrict)}))

Ограничење се интерпретира над елементом који настаје спајањем елемената који учествују у проширеном ограничењу торке. Он мора да задовољи услов који је дат изразом Condition.

Додавање или модификација елемента може да наруши услов проширеног ограничење торке, али се акција мора спречити.

Проширено ограничење торке не постоји у XML Schema-и, па се на основу дефиниције ограничења, прелаже проширење XML Schema-е, које је дато у Листинг 24.

```
<xc:constraint xmlns:xc="http://www.doktorat.org/constraints" type="xc:exTupleCon">
 <xc:condition from="<XPath>" to="<XPath>" keyref="atributi" additional="uslov" />
</xc:constraint>
```

Листинг 24 Проширење шеме за проширено ограничење торке

Вредност xc:exTupleCon атрибута type означава да је у питању дефиниција проширеног ограничења торке. Елемент xc:condition у том случају садржи следеће атрибуте: from, to, keyref и additional. Атрибути from и to садрже XPath изразе којима се селектују одговарајући чворови у XML документу. Атрибут keyref садржи називе атрибута који представљају страни кључ, а

којим се повезују два елемента добијена у from и to XPath изразима. Атрибут additional садржи услов по којем се врши повезивање ових елемената.

У Листинг 25 дат је псеудокод за проверу овог ограничења приликом уноса новог елемента, који је у проширењу шеме означен са to. Ограничење се имплементира помоћу XQuery функције или помоћу тригера.

```
ulaz: e_to∈E_to
BEGIN unos_to
   IF(∃e_from∈E_from)(e_from@PK=e_to@FK)∧provera_dodatnog_uslova=true
   THEN dozvoli unos
   ELSE prijavi grešku
END
```

Листинг 25 Псеудокод за контролу проширеног ограничења торке приликом уноса новог елемента to

У Листинг 26 дат је псеудокод за проверу проширеног ограничења торке приликом модификације подређеног елемента, који је у проширењу шеме означен са to.

```
ulaz: e_to∈E_to
BEGIN modifikacija_to
   IF(∃e_to'∈E_to)(e_to'@PK=e_to@PK)∧(∃e_from∈E_from)(e_to@FK=e_from@PK)∧
                                 provera_dodatnog_uslova=true
   THEN dozvoli modifikaciju
   ELSE prijavi grešku
END
```

Листинг 26 Псеудокод за контролу проширеног ограничења торке приликом модификације новог елемента to

Псеудокод за модификацију надређеног елемента, који је у проширењу шеме означен са from, дат је у Листинг 27.

```
ulaz: novi element e_from∈E_from
BEGIN unos_from
   IF(∃e_from'∈E_from)(e_from'@PK=e_from@PK)∧(¬∃e_to∈E_to)(e_to@FK=e_from'@PK)∧
                                 provera_dodatnog_uslova=false
   THEN dozvoli unos
   ELSE prijavi grešku
END
```

Листинг 27 Псеудокод за контролу проширеног ограничења торке приликом unosa новог елемента from

3.1.5 Ограничење кључа

За сваки елемент дефинише се ограничење кључа. Скуп атрибута X је кључ типа елемента E, ако за свака два елемента e1 и e2, који су истог типа E важи да ако је e1@X=e2@X, онда су и елементи e1 и e2 једнаки. Формула за записивање ограничења кључа је:

$$Key(E, X),\ X \subseteq E,\ (\forall A \in E)(Null(A)=\perp)$$

Ограничење кључа се дефинише за један тип елемента. Интерпретира се у оквиру једног елемента на следећи начин:

1. $(\forall e_1, e_2 \in E)(e_1@X=e2@X \Rightarrow e_1=e_2)$
2. $(\forall X' \subset X)$ ne važi 1.

Услов 1 је услов јединствене вредности кључа, а услов 2 је услов минималности.

Улога елемента који учествује у ограничењу кључа није битна. Дефиниција овог ограничења дата је на следећи начин:

KeyCon(1,
e,
Key(E, X), X⊆E,
uslovi 1. i 2. pravila za interpretaciju,
(e, nebitna uloga, {(insert, Restrict), (update, Restrict)})

У XML Schema-и кључ се задаје елементом xs:key. Промена вредности кључа је могућа, па је потребно дефинисати тригер који ће то забранити.

На основу дефиниције ограничења, прелаже се проширење XML Schema-е у Листинг 28.

```
<xc:constraint xmlns:xc="http://www.doktorat.org/constraints" type="xc:keyCon">
 <xc:condition from="<XPath>" pk="atributi"/>
</xc:constraint>
```

Листинг 28 Проширење шеме за ограничење кључа

XML Schema не забрањује промену вредности кључа. Због тога је уведено проширење XML Schema-е, које се састоји од додавања елемента xc:constraint, чија вредност атрибута type је xc:keyCon. У елементу xc:condition налазе се атрибути from и pk. Атрибут from садржи XPath израз којим се селектује одговарајући чвор у XML документу. Атрибут pk садржи скуп атрибута који представља кључ елемента селектованог у XPath изразу. Ако је примарни кључ и страни кључ у неком другом елементу, онда ће ово ограничење постати део ограничења референцијалног интегритета, што ће бити описано у одељку 3.1.7.

Приликом покушаја модификације кључа, стари елемент је могуће заменити новим, само ако су им вредности кључа исте. У супротном се модификација забрањује. Псеудокод за контролу модификације примарног кључа дата је у Листинг 29.

```
ulaz: e_new∈E, e_old∈E
BEGIN modifikacija_PK
   IF(e_new@PK=e_old@PK)
   THEN dozvoli modifikaciju
   ELSE zabrani modifikaciju
END
```

Листинг 29 Псеудокод за проверу ограничења кључа

3.1.6 Ограничење јединствене вредности

Ограничење јединствене вредности се односи на вредности одређеног атрибута. Када се за неки атрибут дефинише да важи ово ограничење, онда све вредности тог атрибута морају бити јединствене. Формула за записивање овог ограничења је:

$$Unique(E, X), X \subseteq E$$

Ограничење јединствене вредности се дефинише у оквиру једног елемента за неки атрибут, чије вредности треба да буду јединствене. Интерпретира је у оквиру елемента на следећи начин:

1. $(\forall e_1, e_2 \in E)((e_1@X \neq null \wedge e2@X \neq null) \Rightarrow (e_1@X = e2@X \Rightarrow e_1 = e_2)$
2. $(\forall X' \subset X)$ ne važi 1.

Улога елемента у овом ограничењу није битна. Дефиниција ограничења јединствене вредности дата је на следећи начин:

UniqueCon(1,
e,
Unique(E, X), X⊆E,
uslovi 1. i 2. pravila za interpretaciju,
(e, nebitna uloga, {(insert, Restrict), (update, Restrict)}))

У XML Schema-и додаје се елемент xs:unique, којим се дефинише који атрибут има јединствене вредности у оквиру типа елемента за који се дефинише. Ово ограничење у XML моделу података не може да се реализује као у релационом моделу, кад се односи на нула вредности. У релационом моделу података, обележје за које је дефинисано unique ограничење, може имати null вредност више пута; свака од њих је јединствена и различита. У XML моделу података је у декларацији атрибута, који треба да има јединствене вредности, могуће атрибуту use доделити вредност optional, па ако се атрибут не наведе у конкретном елементу, може се сматрати да он нема вредност. Остале вредности које се појављују морају бити јединствене.

3.1.7 Ограничење референцијалног интегритета

Ограничење референцијалног интегритета између два елемента e_1, који је типа E_1, и e_2, који је типа E_2, важи, ако је скуп атрибута Y кључ типа елемента E_2, и важи да је $e_1@X \subseteq e_2@Y$, где су скупови атрибута X и Y унијски компатибилни. Формула за записивање овог ограничења је:

$E_1@X \subseteq E_2@Y$, где је Y кључ елемента E_2

Ограничење референцијалног интегритета се дефинише за два типа елемента, од којих један има улогу референтног, а други референцирајућег елемента. Интерпретира се између два елемента, тако што скуп вредности атрибута X типа елемента E1 треба да буде подскуп скупа вредности атрибута Y типа елемента E2. Дефиниција ограничења гласи:

RICon(2,
me,
E1@X⊆E2@Y,
e1@X⊆e2@Y, e1∈E1, e2∈E2, Key(E2, Y),
(e_1, referencing, {(insert, noAction), (update, noAction)}),
(e_2, referenced, {(delete, noAction, cascade), (update, noAction, Cascade)}))

Брисање надређеног елемента се не може извршити, или се може форсирати каскадно брисање, где ће пре брисања надређеног елемента бити обрисани и сви подређени елементи, који се референцирају на њега. Слично важи и за модификацију надређеног елемента, која може бити забрањена, или се може спровести каскадно. Иако ограничење референцијалног интегритета постоји у XML Schema-и, наведена функционалност није подржана, па се, на основу дефиниције ограничења, предлаже проширење XML Schema-е на следећи начин (Листинг 30):

```
<xc:constraint xmlns:xc="http://www.doktorat.org/constraints" type="xc:refIntCon">
 <xc:condition from="<XPath>" to="<XPath>" keyref="atributi" onupdate="xc:cascade"
              ondelete="xc:cascade"/>
</xc:constraint>
```

Листинг 30 Проширење шеме за ограничење референцијалног интегритета

Вредност xc:refInt атрибута type означава да је у питању дефиниција ограничења референцијалног интегритета. Елемент xc:condition у том случају садржи следеће атрибуте: from, to, keyref, onupdate и ondelete. Атрибути from и to садрже XPath изразе којим се селектује одговарајући чвор у XML документу. Атрибут keyref садржи називе атрибута који представљају страни кључ у референцирајућем елементу. Атрибути onupdate и ondelete могу имати једну од

две вредности: xc:cascade, када се промена вредности примарног кључа пропагира и на зависне елементе, и xc:restrict, када се не дозвољава промена вредности примарног кључа, ако она постоји у референцирајућем елементу.

Ограничење референцијалног интегритета може бити нарушено приликом брисања или модификације референтног елемента. Акција се може спречити или каскадно извршити.

У случају забране акције, брисање референтног елемента се дозвољава само ако не постоји ниједан његов референцирајући елемент. Псеудокод за ову акцију дат је у Листинг 31.

```
ulaz: e_from∈E_from
BEGIN brisanje_from_before
   IF(¬∃e_to∈E_to)(e_to@FK=e_from@PK)
   THEN dozvoli brisanje
   ELSE prijavi grešku
END
```

Листинг 31 Псеудокод за проверу ограничења референцијалног интегритета приликом брисања елемента from (NoAction)

У случају каскадног спровођења акције, брисање надређеног референтног елемента се спроводи тако што се избришу његови подређени референцирајући елементи. Псеудокод за ову акцију дат је у Листинг 32.

```
ulaz: e_from_old∈E_from, e_from_new∈E_from
BEGIN brisanje_from_after
   (∀e_to∈E_to)(e_to@FK=e_from_old@PK)
       briši e_to
END
```

Листинг 32 Псеудокод за проверу ограничења референцијалног интегритета приликом брисања елемента from (Cascade)

У случају забране акције, модификација референтног елемента се дозвољава само ако не постоји ниједан његов референцирајући елемент. Псеудокод за ову акцију дат је у Листинг 33.

```
ulaz: e_from∈E_from
BEGIN modifikacija_from_before
   IF(¬∃e_to∈E_to)(e_to@FK=e_from@PK)
   THEN dozvoli modifikaciju
   ELSE prijavi grešku
END
```

Листинг 33 Псеудокод за проверу ограничења референцијалног интегритета приликом модификације елемента from (NoAction)

У случају каскадног спровођења акције, модификација надређеног референтног елемента се спроводи тако што се модификују његови подређени референцирајући елементи. Псеудокод за ову акцију дат је у Листинг 34.

```
ulaz: e_from_old∈E_from, e_from_new∈E_from
BEGIN modifikacija_from_after
   (∀e_to∈E_to)(e_to@FK=e_from_old@PK)
       modifikuj e_to set e_to@FK:=e_from_new@PK
END
```

Листинг 34 Псеудокод за проверу ограничења референцијалног интегритета приликом модификације елемента from (Cascade)

3.1.8 Ограничење инверзног референцијалног интегритета

Ограничење инверзног референцијалног интегритета између два типа елемента E_1 и E_2 важи, ако између истих типова елемената важи референцијални интегритет $E_1@X \subseteq E_2@Y$, где је скуп

атрибута Y кључ елемента E_2, а скупови обележја X и Y су унијски компатибилни, и постоји егзистенцијална зависност између реалних ентитета који су представљени типовима елемената E_1 и E_2. Формула за запис ограничења инверзног референцијалног интегритета је:

$$E_2@Y \subseteq E_1@X, Key(E_2, Y)$$

Типови елемената који учествују у ограничењу инверзног референцијалног интегритета имају различите улоге. Један тип елемента је референцирајући, а други је референтан. Ограничење се интерпретира за два елемента. За сваки елеменат E_2, постоји један елемент E_1, такав да је $E_1@Y=E_2@X$. Дефиниција ограничења инверзног референцијалног интегритета је:

IRICon(2,
me,
$E_2@Y \subseteq E_1@X$, $Key(E_2, Y)$,
$e_2@Y \subseteq e_1@X$, $e_1 \in E_1$, $e_2 \in E_2$, $Key(E_2, Y)$,
((e_2, referencing, {(insert, noAction, cascade), (update, noAction, cascade)}),
(e_1, referenced, {(delete, noAction, cascade), (update, noAction, cascade)})))

Провера важења ограничења инверзног референцијалног интегритета мора бити одложена. Она се дешава након операција уноса, модификације или брисања из референцирајућег и референтног елемента. Брисање или модификација референтног елемента су забрањене, или се дозвољава каскадна акција, која ће исто урадити и у референцирајућем елементу. Додавање или модификација референцирајућег елемента је забрањена, ако не постоји одговарајући референтни елемент, или је каскадна, ако се провера спроводи одложено, након операције.

Ограничење инверзног референцијалног интегритета није подржано XML Schema-ом. Због тога се, на основу дефиниције ограничења, прелаже проширење XML Schema-е као у Листинг 35.

```
<xc:constraint xmlns:xc="http://www.doktorat.org/constraints" type="xc:invRefInt">
 <xc:condition from="<XPath>" to="<XPath>" keyref="atributi"/>
</xc:constraint>
```

Листинг 35 Проширење шеме за ограничење инверзног референцијалног интегритета

Вредност xc:invRefInt атрибута type означава да је у питању дефиниција ограничења инверзног референцијалног интегритета. Елемент xc:condition у том случају садржи следеће атрибуте: from, to и keyref. Атрибути from и to садрже XPath изразе којим се селектује одговарајући чвор у XML документу. Атрибут keyref садржи називе атрибута који представљају страни кључ.

Ограничење инверзног референцијалног интегритета може бити нарушено приликом уноса или модификације надређеног елемента, у ограничењу означеног са from, као и приликом брисања или модификације подређеног елемента, означеног са to.

Приликом уноса елемента from може доћи до међусобног закључавања елемената from и to, ако се ограничење инверзног референцијалног интегритета привремено не онемогући. То се може постићи искључивањем тригера који проверава ово ограничење. Затим се унесу надређени и подређени елементи (from и to), а онда се тригер поново укључи и провери да ли је ограничење задовољено. Псеудокод за контролу ограничења приликом уноса надређеног елемента (from) дат је у Листинг 36.

```
ulaz: e_from∈E_from
BEGIN unos_from
   IF(isključen triger)
   THEN dozvoli unos
   ELSE prijavi grešku
END
```

Листинг 36 Псеудокод за проверу ограничења инверзног референцијалног интегритета приликом уноса елемента from

Одмах након уноса елемента from, треба унети и подређени елемент to. Приликом уноса овог елемента, ограничење референцијалног интегритета ће дозволити да се унесу само постојеће вредности кључа надређеног елемента. Псеудокод за унос елемента to дат је у Листинг 37.

```
ulaz: e_to∈E_to
BEGIN unos_to
   IF(RICon(e_to))
   THEN dozvoli unos
   ELSE prijavi grešku
END
```

Листинг 37 Псеудокод за проверу ограничења инверзног референцијалног интегритета приликом уноса елемента to

Након оба уноса потребно је укључити тригер који ће проверити да ли је задовољено ограничење инверзног референцијалног интегритета. То значи да треба да провери да ли за сваки надређени референтни елемент постоји подређени, који се на њега референцира. У Листинг 38 дат је псеудокод тригера који проверава да ли је задовољено ограничење.

```
BEGIN proveri_IRICon_triger
   (∀e_from∈E_from)
      IF(¬∃e_to∈E_to)(e_from@to_keyref=e_to@to_keyref)
      THEN obriši e_from
END
```

Листинг 38 Псеудокод тригера који се окида након уноса елемената

Ако се брише подређени елемент, у ограничењу означен са to, потребно је проверити да ли је он последњи елемент на који указује његов надређени. Уколико јесте, брисање треба забранити. Псеудокод за контролу ограничења инверзног референцијалног интегритета приликом брисања подређеног елемента дат је у Листинг 39.

```
ulaz: e_to∈E_to
BEGIN brisanje_to
   (∀e∈E_to)
      e@to_keyref=e_to@to_keyref
      IF(count(e)=1)
      THEN zabrani brisanje
      ELSE dozvoli brisanje
END
```

Листинг 39 Псеудокод за проверу ограничења инверзног референцијалног интегритета приликом брисања елемента to

Модификација подређеног елемента to може да наруши ограничење инверзног референцијалног интегритета, ако је елемент који се брише последњи који је повезан са својим надређеним елементом. Тада треба спречити модификацију. Псеудокод ове контроле дат је у Листинг 40.

```
ulaz: e_to, e_to_old∈E_to
BEGIN modifikacija_to
   (∀e∈E_to)
      e@to_keyref=e_to_old@to_keyref
      IF(count(e)=1)
      THEN zabrani modifikaciju
```

```
      ELSE dozvoli modifikaciju
END
```

Листинг 40 Псеудокод за проверу ограничења инверзног референцијалног интегритета приликом модификације елемента to

3.1.9 Проширено ограничење референцијалног интегритета

У XML документу могу да постоје елементи који се не могу директно повезати референцијалним интегритетом. Посматрајмо елементе e_1, који је типа E_1, и e_2, који је типа E_2, и њихове скупове атрибута $X \in E_1$ и $Y \in E_2$. Ако се неки од атрибута елемента E_2 не налази у елементу E_1, а постоји елемент E_a који тај атрибут садржи, онда треба спојити елементе E_1 и E_a, како би се дошло до траженог атрибута. Скуп обележја који се добија спајањем елемената обележен је са Z. Формула за запис проширеног ограничења референцијалног интегритета, дата је на следећи начин:

$$\bowtie(E_1, E_a)@Z \subseteq E_2@Z$$

Ово ограничење се дефинише за више типова елемената, а интерпретира се међу елементима који учествују у ограничењу. Скуп атрибута у споју елемената, мора припадати скупу атрибута елемента са десне стране ове инклузије. Улоге елемената у овом ограничињу су, слично као код референцијалног интегритета, референцирајући и референтни елемент. Референцирајућих елемената може бити више од једног. Проширено ограничење референцијалног интегритета се записује на следећи начин:

ExRICon(n,
me,
$\bowtie(E_1, E_a)@Z \subseteq E_2@Z$,
$\bowtie(e_1, e_a)@Z \subseteq e_2@Z$,
((e_1, referencing, {(insert, noAction), (update, noAction)}),
(e_2, referenced, {(delete, noAction, cascade), (update, noAction, cascade)})))

Брисање или модификација надређеног, референтног елемента се не може извршити, или се може форсирати каскадно брисање, где ће пре брисања надређеног елемента бити обрисани и сви подређени елементи, који се референцирају на њега. Проширено ограничење референцијалног интегритета није подржано XML Schema-ом, па се, на основу дефиниције ограничења, прелаже проширење XML Schema-е као у Листинг 41.

```
<xc:constraint xmlns:xc="http://www.doktorat.org/constraints" type="xc:exRefInt">
 <xc:condition from="<XPath>" fromPK="atribut" linkKeyref="atribut"  to="<XPath>"
  link="<XPath>" linkPK="atribut"/>
</xc:constraint>
```

Листинг 41 Проширење шеме за проширено ограничење референцијалног интегритета

Вредност xc:exRefInt атрибута type означава да је у питању дефиниција проширеног ограничења референцијалног интегритета. Елемент xc:condition у том случају садржи следеће атрибуте: from, to, link, fromPK, linkKeyref и linkPK. Атрибути from, to и link садрже XPath изразе којима се селектују одговарајући чворови у XML документу. Атрибути fromPK, linkKeyref и linkPK садрже називе атрибута који представљају примарне и стране кључеве, који учествују у проширеном ограничењу торке. Атрибут linkKeyref садржи кључ који се не појављује у елементу to, а потребан је да би се повезали елементи from и to. Атрибут fromPK садржи кључ елемента from. Атрибут linkPK садржи кључ елемента link.

Елемент означен са link, додат је у ово ограничење да би био надокнађен страни кључ који недостаје. Унос новог елемента означеног са link, не нарушава ограничење проширеног референцијалног интегритета, јер је он у овом случају независан од осталих елемената који

учествују у ограничењу. Његова модификација би могла да наруши проширено ограничење референцијалног интегритета. Псеудокод за контролу овог ограничења приликом модификације референцирајућег елемента, дат је у Листинг 44.

```
ulaz: e_link, e_link_old∈E_link
BEGIN modifikacija_link
   IF(e_link_old@linkPK=e_link@linkPK)
   THEN(∀e_to∈E_to)(e_to@linkPK=e_link@linkPK)
          (∀e_from∈E_from)(e_from@linkKeyref=e_link@linkKeyref ∧ e_from@fromPK=e_to@fromPK)
          IF count(e_to)=count(e_from)
          THEN dozvoli modifikaciju
          ELSE prijavi grešku
   ELSE prijavi grešku
END
```

Листинг 42 Псеудокод за проверу проширеног ограничења референцијалног интегритета приликом модификације елемента link

Референцирајући елемент, који је у ограничењу означен са to, представља подређени елемент који је требало да учествује у референцијалном интегритету, али су недостајали атрибути који су уведени помоћу елемента означеног са link. Унос и модификација овог елемента би могли да наруше проширено ограничиње референцијалног интегритета. Псеудокод за контролу ограничења приликом уноса дата је у Листинг 43, а приликом модификације у Листинг 44.

```
ulaz: novi element e_to∈E_to
lokalna promenljiva: e_link∈E_link
BEGIN unos_to
   e_link@fromPK=e_to@fromPK
   IF(∃e_from∈E_from)(e_from@fromPK=e_to@fromPK ∧ e_from@linkKeyref=e_link@linkKeyref)
   THEN dozvoli operaciju
   ELSE prijavi grešku
END
```

Листинг 43 Псеудокод за проверу проширеног ограничења референцијалног интегритета приликом уноса елемента to

```
ulaz: novi element e_to∈E_to
lokalna promenljiva: e_link∈E_link
BEGIN modifikacija_to
   e_link@fromPK=e_to@fromPK
   IF(∃e_from∈E_from)(e_from@fromPK=e_to@fromPK ∧ e_from@linkKeyref=e_link@linkKeyref)
   THEN dozvoli operaciju
   ELSE prijavi grešku
END
```

Листинг 44 Псеудокод за проверу проширеног ограничења референцијалног интегритета приликом модификације елемента to

Елемент који је означен са from је надређени елемент за елемент to. Део његовог кључа ће као страни кључ бити укључен у кључ елемента to. Због тога брисање и модификација овог елемента могу довести до нарушавања проширеног ограничења референцијалног интегритета. Псеудокод за контролу овог ограничења приликом брисања елемента from дат је у Листинг 45, а приликом модификације у Листинг 46.

```
ulaz: e_from∈E_from
BEGIN brisanje_from
   IF((∃e_link∈E_link)(e_link@linkKeyref=e_from@linkKeyref)∧
      (∃e_to∈E_to)(e_to@fromPK=e_from@fromPK ∧e_to@linkPK=e_link@linkPK))
   THEN prijavi grešku
   ELSE dozvoli brisanje
END
```

Листинг 45 Псеудокод за проверу проширеног ограничења референцијалног интегритета приликом брисања елемента from

```
ulaz: e_from, e_from old∈E_from
BEGIN modifikacija_from
   IF(e_from@PK=e_from old@PK)
   THEN dozvoli modifikaciju
   ELSE zabrani modifikaciju
END
```

Листинг 46 Псеудокод за проверу проширеног ограничења референцијалног интегритета приликом модификације елемента from

3.2 Закључак

У овом поглављу дата је таксономија типова ограничења у XML моделу података, по угледу на таксономију ограничења у релационом моделу дату у [Luk03, Luk07]. Обрађени су типови ограничења који постоје у XML Shema-и, и то су ограничење домена, ограничење вредности атрибута и ограничење јединствене вредности. Обрађени су и типови ограничења који постоје у XML Schema-и, али не задовољавају у потпуности функционалност из релационог модела, те је потребно проширити XML Schema-у. То су ограничење торке, ограничење кључа и ограничење референцијалног интегритета. Дати су и нови типови ограничења, који постоје у пракси, немају еквивалент у XML Schema-и, и нису имплементирани ни у једном систему за управљање XML базом података. Нови типови ограничења дефинисани су по угледу на типове ограничења у релационом моделу података. Одабрана су три репрезентативна типа ограничења, која се могу наћи у пракси. То су проширено ограничење торке, ограничење инверзног референцијалног ограничења и проширено ограничење референцијалног интегритета.

Методологијом приказаном у овом поглављу, могуће је обрадити и остале типове ограничења која постоје у релационом моделу података, као што је, на пример, селективно ограничење референцијалног интегритета.

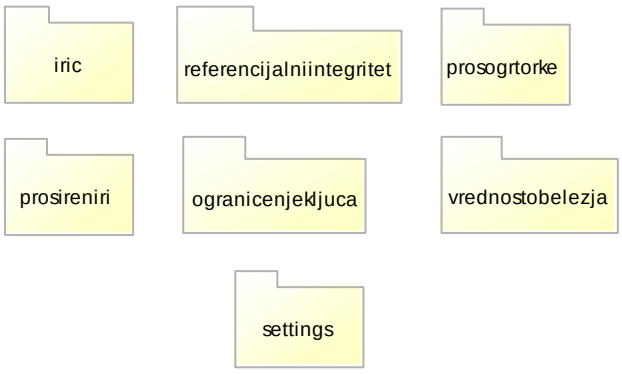
Приказани типови ограничења не обухватају све типове ограничења који постоје у релационом моделу података, што оставља простор за будућа истраживања у овој области. Ако би била усвојена нотација предложена у овом поглављу, пројектанти би формално и на декларативан начин могли да запишу све наведене типове ограничења, што до сада није био случај. На основу спецификација типова ограничења датих у овом поглављу, реализован је генератор кода, који је детаљније описан у поглављу 4. Генерисани код проверава да ли су задовољена ограничења приликом обављања основних операција са подацима, као што су додавање новог, измена и брисање постојећег елемента у XML бази података. Генератор може да произведе два типа програма. Први је XQuery функција, а други је тригер. У зависности од конкретног XML система за управљање базом података, биће коришћено прво или друго решење. Детаљан опис имплементације провере ограничења дат је у поглављу 5.

4 Генератор кода за реализацију ограничења у XML моделу података

У претходном поглављу дата је типизација ограничења у XML моделу података. За сваки тип ограничења дат је предлог проширења XML Schema-е. У овом поглављу биће описан генератор кода који, на основу проширене XML Schema-е за конкретан XML документ, генерише или тригер или XQuery функцију која имплементира задато ограничење. Генератор кода је реализован у програмском језику Java [Java].

4.1 Спецификација генератора кода

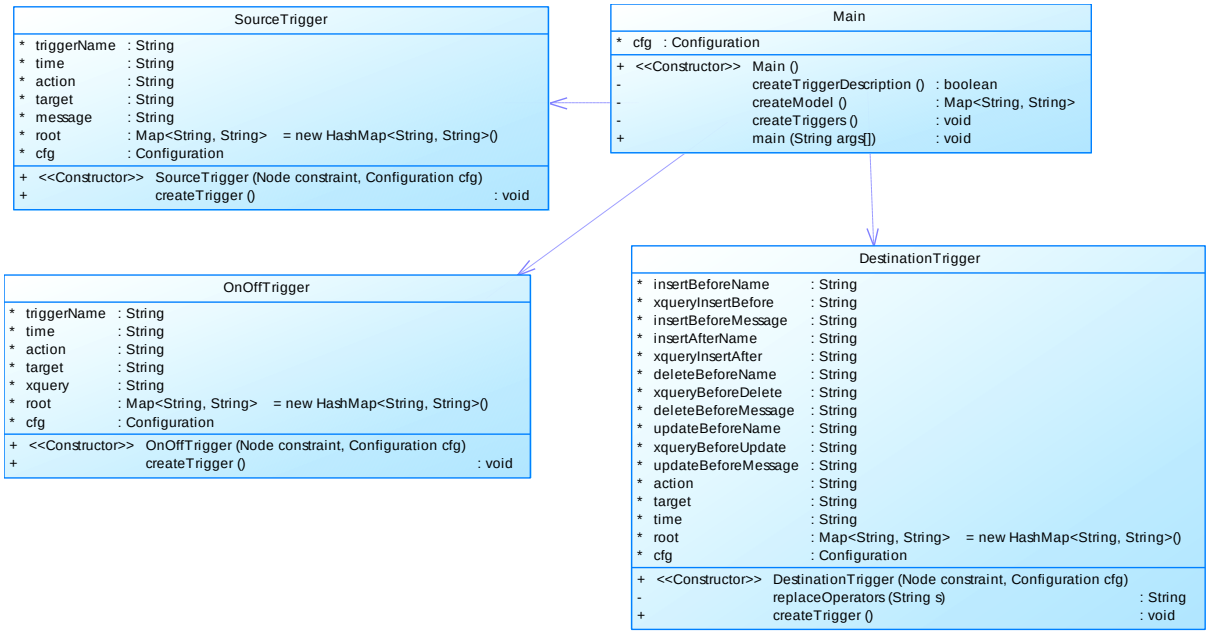
Генератор кода је Java апликација која је организована модуларно. За сваку врсту ограничења направљен је посебан пакет и посебан скуп шаблона.



Слика 4 Дијаграм класа са пакетима у генератору кода

За свих шест типова ограничења, за које је предложено проширење у XML моделу података, направљен је одговарајући пакет. На пример, за ограничење инверзног референцијалног интегритета користи се пакет iric. Дијаграм класа са пакетима дат је на Слика 4. Како су пакети формирани на исти начин, у наставку ће бити описан један од њих.

На Слика 5 приказан је дијаграм класа пакета који обрађује ограничење инверзног референцијалног интегритета.



Слика 5 Дијаграм класа пакета који реализује ограничење инверзног референцијалног интегритета

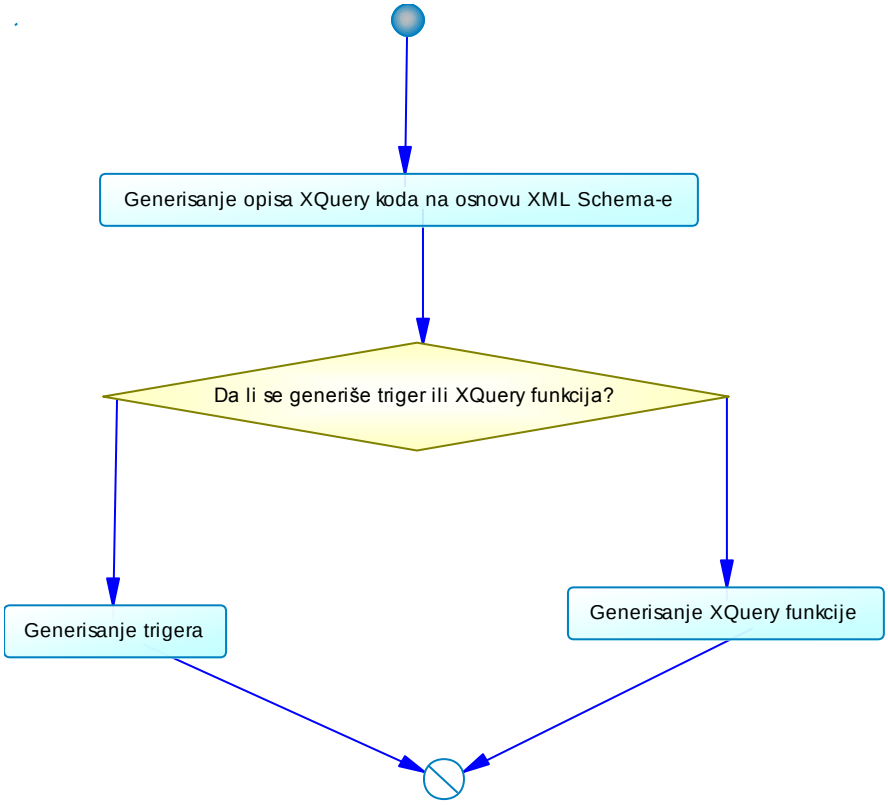
Главна класа у пакету је класа Main. Метода createModel чита проширену XML Schema-у, дату у Листинг 47 и попуњава шаблон за генерисање описа XQuery кода, који је дат у Листинг 48. Метода createTriggers чита генерисани опис XQuery кода и на основу атрибута type, описаног у Листинг 48, инстанцира одговарајућу класу из овог пакета. Ако је вредност атрибута type једнака onOff, креира се објекат класе OnOffTrigger. За вредност атрибута type source, креира се објекат класе SourceTrigger, а вредност destination, објекат класе DestinationTrigger. Све три класе имају методу createTrigger, која на основу унетих параметара попуњава шаблон и генерише или тигер или XQuery функцију.

Класа OnOffTrigger генерише тригер или функцију који се активирају приликом поновног укључивања тригера. Ова класа је специфична баш за ограничење инверзног референцијалног интегритета, јер је потребно привремено исључити рад тригера, док се не обави унос надређеног и подређеног елемента. Затим се тригер поново укључује и проверава се да ли је ово ограничење задовољено.

Класа SourceTrigger генерише тригер или функцију који се активирају пре додавања новог надређеног елемента. Класа DestinationTrigger генерише тригер или функцију који се активирају приликом рада са подређеним елементима. Детаљан опис конкретних тригера и функција за проверу ограничења инверзног референцијалног интегритета дат је у одељку 5.3.3.

Предложен начин генерисања кода је проширив, јер се за свако ново ограничење може направити пакет са својим скупом шаблона.

На Слика 6 приказан је дијаграм активности који описује процес генерисања кода за XQuery функције или тригере, на основу описа типова ограничења и проширења XML Schema-е.



Слика 6 Дијаграм активности који описује процес генерисања кода на основу описа типова ограничења

4.2 Пример генерисања кода за проверу ограничења инверзног референцијалног интегритета

Генерисање кода за проверу ограничења инверзног референцијалног интегритета почиње читањем XML Schema-е. На основу описа ограничења у елементу xc:constraint, генерише се опис XQuery кода који ће реализовати ограничење. У Листинг 47 дат је пример XML Schema документа који садржи проширење везано за опис ограничења инверзног референцијалног интегритета, а на којем ће бити описан процес генерисања кода.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="Database">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Fakultet" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="FacId" type="xs:string" use="required"/>
          <xs:attribute name="FacNaziv" type="xs:string" use="required"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="Departman" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="DepId" type="xs:string" use="required"/>
          <xs:attribute name="DepNaziv" type="xs:string" use="required"/>
          <xs:attribute name="FacId" type="xs:string" use="optional"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:key name="Fakultet_PK">
    <xs:selector xpath="Fakultet"/>
```

```
    <xs:field xpath="@FacId"/>
  </xs:key>
  <xs:key name="Departman_PK">
    <xs:selector xpath="Departman"/>
    <xs:field xpath="@DepId"/>
    <xs:field xpath="@FacId"/>
  </xs:key>
  <xs:keyref name="Departman_FK" refer="Fakultet_PK">
    <xs:selector xpath="Departman"/>
    <xs:field xpath="@FacId"/>
  </xs:keyref>
  <xc:constraint type="xc:invRefInt">
    <xc:condition from="/Database/Fakultet" to="/Database/Departman"
                  keyref="FacId"/>
  </xc:constraint>
 </xs:element>
</xs:schema>
```

Листинг 47 Пример XML Schema документа за ограничење инверзног референцијалног интегритета

Опис XQuery кода је генерисан употребом FreeMarker Java Template Engine библиотеке [FreeMarker]. Ова библиотека се ослања на шаблоне написане у FreeMarker нотацији које се затим попуњава конкретним подацима и на основу тога се генерише излазна датотека. У Листинг 48 приказан је FreeMarker шаблон на основу ког се генерише опис XQuery кода.

```
<xc:triggers xmlns:xc="http://www.doktorat.org/constraints"
 type="xc:invRefInt">
 <xc:trigger type="source" name="IRICInsert${fromName}"
  time="xc:before" action="xc:insert" target="${from}"
  message="${fromName} ne moze da se ubaci bez ${toName}" />

 <xc:trigger type="onOff" name="IRICUpdateTrigger"
  time="xc:after" action="xc:update" target="${from}"
  xquery="${docName}${from}[@${keyref}=${docName}${from}/@${keyref}[not(. =
          ${docName}${to}/@${keyref})]]" />

 <xc:trigger type="destination" name="IRICInsert${toName}Before"
  action="xc:insert" target="${to}" time="xc:before"
  xquery="exists(${docName}${from}[@${keyref}=$NEW/@${keyref}])"
  message="${toName} ne moze da se ubaci,
           posto ne postoji ${fromName} na koji se referencira"/>

 <xc:trigger type="destination" name="IRICInsert${toName}After"
  action="xc:insert" target="${to}" time="xc:after"
  xquery="${docName}${from}[@${keyref}=${docName}${from}/@${keyref}[not(. =
          ${docName}${to}/@${keyref})]]" />

 <xc:trigger type="destination" name="IRICDelete${toName}Before"
  action="xc:delete" target="${to}" time="xc:before"
  xquery="exists(${docName}${from}[@${keyref}=$OLD/@${keyref}]) and
          count(${docName}${to}[@${keyref}=$OLD/@${keyref}]) ?le; 1"
  message="${toName} ne moze da se obrise,
           posto je poslednji za svoj ${fromName}" />

 <xc:trigger type="destination" name="IRICUpdate${toName}Before"
  action="xc:update" target="${to}" time="xc:before"
  xquery="count(${docName}${to}[@${keyref}=$OLD/@${keyref}]) ?le; 1"
  message="${toName} ne moze da se preveze na drugi ${fromName},
           posto je poslednji za svoj ${fromName}" />
</xc:triggers>
```

Листинг 48 FreeMarker шаблон за генерисање описа XQuery кода

Елемент xc:triggers може да садржи више елемената xc:trigger. Сваки елемент xc:trigger описује XQuery код који ће бити активиран приликом извршавања конкретне акције. Атрибути time и action елемента xc:trigger дефинишу време и тип конкретне акције. Вредност атрибута time може бити xc:before или xc:after, у зависности од тога да ли се тригер или функција извршавају

пре или после задате акције. Вредност атрибута action може бити xc:insert, xc:update или xc:delete. Атрибут xquery садржи XQuery код који ће се извршавати у тригеру или функцији и који ће обављати проверу испуњености услова задатог ограничења. Атрибут message садржи текст поруке која се исписује кориснику ако ограничење није задовољено. Атрибут type одређује којим шаблоном ће бити креиран тригер.

Вредности означене изразом ${naziv} представљају промењљиве које ће FreeMarker Template Engine заменити конкретним вредностима. Те вредности се добијају из генератора кода, на основу анализе XML Schema-е.

Генератор кода, на основу XML Schema-е и шаблона, генерише опис XQuery кода који ће бити коришћен или у тригеру или у XQuery функцији. Опис XQuery кода је XML документ чији пример је дат у Листинг 49.

```
<xc:triggers xmlns:xc="http://www.doktorat.org/constraints"
 type="xc:invRefInt">
 <xc:trigger type="source" name="IRICInsertFakultet"
  time="xc:before" action="xc:insert" target="/Database/Fakultet"
  message="Fakultet ne moze da se ubaci bez Departman" />

 <xc:trigger type="onOff" name="IRICUpdateTrigger"
  time="xc:after" action="xc:update" target="/Database/Fakultet"
  xquery="doc('fakultetDepartman.xml')/
          Database/Fakultet[@FacId=doc('fakultetDepartman.xml')/
          Database/Fakultet/@FacId[not(. =
          doc('fakultetDepartman.xml')/Database/Departman/@FacId)]]" />

 <xc:trigger type="destination" name="IRICInsertDepartmanBefore"
  action="xc:insert" target="/Database/Departman" time="xc:before"
  xquery="exists(doc('fakultetDepartman.xml')/
                 Database/Fakultet[@FacId=$NEW/@FacId])"
  message="Departman ne moze da se ubaci,
           posto ne postoji Fakultet na koji se referencira"/>

 <xc:trigger type="destination" name="IRICInsertDepartmanAfter"
  action="xc:insert" target="/Database/Departman" time="xc:after"
  xquery="doc('fakultetDepartman.xml')/
          Database/Fakultet[@FacId=doc('fakultetDepartman.xml')/
          Database/Fakultet/@FacId[not(. =
          doc('fakultetDepartman.xml')/Database/Departman/@FacId)]]" />

 <xc:trigger type="destination" name="IRICDeleteDepartmanBefore"
  action="xc:delete" target="/Database/Departman" time="xc:before"
  xquery="exists(doc('fakultetDepartman.xml')/
                 Database/Fakultet[@FacId=$OLD/@FacId]) and
                 count(doc('fakultetDepartman.xml')/
                 Database/Departman[@FacId=$OLD/@FacId]) ?le; 1"
  message="Departman ne moze da se obrise, posto je poslednji za svoj Fakultet" />

 <xc:trigger type="destination" name="IRICUpdateDepartmanBefore"
  action="xc:update" target="/Database/Departman" time="xc:before"
  xquery="count(doc('fakultetDepartman.xml')/
                Database/Departman[@FacId=$OLD/@FacId]) ?le; 1"
  message="Departman ne moze da se preveze na drugi Fakultet,
           posto je poslednji za svoj Fakultet" />
</xc:triggers>
```

Листинг 49 Опис XQuery кода за реализацију ограничења инверзног референцијалног интегритета

У Листинг 49 види се да је FreeMarker Template Engine заменио вредности промењљивих конкретним вредностима из XML Schema-е. На основу овог документа генерише се или тригер или XQuery функција употребом *FreeMarker Template Engine* библиотеке. Шаблони за тригер и XQuery функцију направљени су на основу псеудокода датог у одељку 3.1.8, у Листинг 39. Шаблон за тригер приказан је у Листинг 50.

```
CREATE TRIGGER "${deleteBeforeName}"
BEFORE DELETE
ON ${collectionName}${target}
FOR EACH NODE
DO {
 if (${xqueryBeforeDelete})
 then
  error(xs:QName("${deleteBeforeName}"),"${deleteBeforeMessage}")
 else
   ($OLD);
```

Листинг 50 Шаблон за тригер

Променљива ${deleteBeforeName} садржи име тригера, и попуњава се из атрибута name елемента xc:trigger. Променљива ${collectionName} је глобална променљива која садржи путању до колекције у коју је смештен XML документ. Она је глобална, јер се не налази у конкретним XML Schema документима, па се не може из њих прочитати. XPath путања до чвора на који утиче тригер, налази се у променљивој ${target}. Променљива ${xqueryBeforeDelete} садржи XQuery код наведен у опису XQuery кода, конкретно у атрибуту xquery елемента xc:trigger. Слично томе, променљива ${deleteBeforeMessage} подешава се на основу атрибута message у елементу xc:trigger.

У Листинг 51 дат је пример тригера који је добијен на основу претходно описаног шаблона.

```
CREATE TRIGGER "IRICDeleteDepartmanBefore"
BEFORE DELETE
ON collection('IRIC')/Database/Departman
FOR EACH NODE
DO {
 if (exists(fn:doc('FakultetDepartman', 'IRIC')/
            Database/Fakultet[@FacId=$OLD/@FacId]) and
            count(fn:doc('FakultetDepartman', 'IRIC')/
            Database/Departman[@FacId=$OLD/@FacId]) <= 1)
 then
  error(xs:QName("IRICDeleteDepartmanBefore"),
        "Departman ne moze da se obrise, posto je poslednji za svoj Fakultet")
 else
   ($OLD);
}
```

Листинг 51 Тригер добијен на основу шаблона

Ако је потребно генерисати XQuery функцију, тада се користи други шаблон. Тај шаблон је дат у Листинг 52.

```
declare function local:canDelete${toName}($OLD as element(${toName}))
    as xs:boolean {
    let $res := not (${xquery})
    return $res
};

declare function local:delete${toName}($OLD as element(${toName}))
    as xs:boolean {
        let $CanDelete := local:canDelete${toName}($OLD)
        let $res := if($CanDelete)
                    then update delete ${docName}${to}[${deleteWhere}]
                    else true()
        return $CanDelete
};
```

Листинг 52 Шаблон за XQuery функцију

Променљива ${docName} је глобална променљива која садржи назив документа. Променљива ${toName} представља име елемента у XML докуемнту, за који се генерише XQuery функција и добија се из XML Schema-е. Променљива ${xquery} попуњава се из истоименог атрибута из

xc:condition елемента приказаног у Листинг 49. Променљива ${to} садржи XPath израз којим се селектује чвор који је потребно обисати. Ова вредност се налази у проширеној XML Schema-и, приказаној у Листинг 47. Променљива ${deleteWhere} садржи XPath израз који представља услов за селекцију чвора који се брише. Садржај ове променљиве се такође добија из проширене XML Schema-е.

Генерисана XQuery функција приказана је у Листинг 53.

```
declare function local:canDeleteDepartman($OLD as element(Departman))
    as xs:boolean {
    let $res :=
        not(exists(doc('fakultetDepartman.xml')/
            Database/Fakultet[@FacId=$OLD/@FacId]) and
            count(doc('fakultetDepartman.xml')/
            Database/Departman[@FacId=$OLD/@FacId]) <= 1)
    return $res
};

declare function local:deleteDepartman($OLD as element(Departman))
    as xs:boolean {
        let $CanDelete := local:canDeleteDepartman($OLD)
        let $res := if($CanDelete)
                    then update delete doc('fakultetDepartman.xml')/
                         Database/Departman[@DepId=$OLD/@DepId and
                                            @FacId=$OLD/@FacId]
                    else true()
        return $CanDelete
};
```

Листинг 53 XQuery функција добијена на основу шаблона

4.3 Закључак

У овом поглављу приказане су спецификација и имплементација генератора кода. Генератор кода генерише програмски код за проверу свих типова ограничења, која су наведена у поглављу 3. За демонстрацију генерисања кода одабрано је ограничење инверзног референцијалног интегритета. Није наведен комплетан генерисан код, већ је дат пример генерисаног кода за проверу ограничења инверзног референцијалног интегритета приликом брисања елемента. Детаљан приказ кода за овај тип ограничења, као и дискусија како ово ограничење „делује“ у садејству с ограничењем референцијалног интегритета и како се реализује његова провера, дати су у одељку 5.3.3.

Основна намена овог генератора кода је да аутоматизује процес провере ограничења под условом да се усвоји нотација за дефиницију типова ограничења описана у поглављу 3. На тај начин, програмерима би био олакшан посао употребом генерисаног кода уместо ручног програмирања провере ограничења.

Основа за реализацију генератора кода је псеудокод, приказан у поглављу 3. На основу псеудокода генерисан је шаблон, који се попуњава на основу спецификације типа ограничења дате проширеном XML Schema-ом.

Генератор поизводи две врсте кода: XQuery функције и тригере. На основу истраживања приказаног у поглављу 2, одабрана су два XML система за управљање базом података, eXist [eXist] и Sedna [Sedna]. СУБП eXist има веома слабу подршку за тригере, те се за њега генерише код као XQuery функција. СУБП Sedna у потпуности подржава тригере, те се за њу могу искористити генерисани тригери.

У прегледаној литератури, ниједан од радова није приказао генератор кода, који се базира на формалној спецификацији ограничења. Анализирани радови приказују спецификације ограничења, али имплементација верификације тих ограничења није аутоматизована.

5 Имплементација ограничења у XML моделу података

У одељку 2.2.1 наведена су ограничења у релационом моделу података која су имплементирана путем декларативних механизама у већини система за управљање базама података, као и нека која се не могу имплементирати на тај начин, већ искључиво путем процедуралних механизама, који укључују тригере, процедуре и функције. Нека од тих ограничења су подржана и имплементирана у XML моделу података, а за нека је потребно направити додатну спецификацију и имплементацију. Нека ограничења могу бити исказана структуром и већ су описана у одељку о XML Schema-и, а односе се на начин појављивања једног елемента у оквиру другог. На основу анализе у одељку 2.12, одабрана је структура XML документа код које су сви елементи директно подређени корену, а везе између њих реализоване су путем ref/keyref механизма. Овакав начин формирања XML документа је бољи уколико је потребно подржати ограничење референцијалног интегритета и јединственост вредности кључа. Документи формирани на овај начин су мањи и избегава се редундантност података.

XML модел података подржава поступке за дефинисање различитих типова ограничења. У одељку 5.1 прво ће бити описана имплементација ограничења, која су у потпуности подржана у XML Schema-и. Затим ће, у одељку 5.2, бити описана имплементација ограничења за која се предлаже проширење XML Schema-е. У последњем одељку, 5.3, биће дата имплементација ограничења која не постоје у XML Schema-и, а за која се такође предлаже проширење стандарда. Уз последња ограничења, биће дати примери из релационог модела података, као и примери у XML моделу података, који приказују конкретну примену дефинисаних ограничења.

За све примере генерисане су XQuery функције и тригери. За XML СУБП eXist користе се XQuery функције, а за XML СУБП Sedna користе се тригери. Ова подела је условљена могућностима наведених СУБП. СУБП eXist не подржава у довољној мери тригере, а СУБП Sedna их подржава. Функције су генерисане тако да, осим извршења одређене операције ажурирања, врше и проверу одговарајућег ограничења. На тај начин се, уместо позива основне операције ажурирања, позива XQuery функција. Ако је ограничење задовољено, функција извршава ту основну операцију. С друге стране, тригери врше само проверу одговарајућег ограничења, и у зависности од ограничења које реализују, дозвољавају или забрањују операцију ажурирања.

5.1 Имплементација ограничења која постоје у XML Schema-и

У овом одељку дат је преглед ограничења из релационог модела података, која већ постоје у XML Schema-и. То су ограничење домена, ограничење вредности атрибута и ограничење јединствене вредности. Спецификације ових ограничења дате су у одељцима 3.1.1, 3.1.2 и 3.1.6, редом.

5.1.1 Ограничење домена

DTD, као један од мета језика XML модела података подржава веома скроман број атомичних типова података и не подржава извођење нових атомичних типова путем ограничавања дозвољених вредности постојећих типова. XML Schema подржава велик број уграђених

атомичних типова података, а омогућава и извођење нових атомичних типова података путем ограничавања опсега могућих вредности постојећих типова. Ограничење скаларних типова података може се поредити са ограничењем домена у релационом моделу података.

На самом почетку пројектовања шеме базе података, потребно је специфицирати интегритет или ограничење домена, као скупа могућих вредности будућих обележја шеме базе података. У релационом моделу података ограничење домена је ванрелациони тип ограничења и дефинише се над постојећим или кориснички дефинисаним доменима.

У XML моделу података постоје много веће могућности за дефинисање ограничења домена. Могуће је користити уграђене типове података, као и изведене и кориснички дефинисане. У XML шеми [XMLSch], ограничење домена може се дефинисати и за атрибуте и за елементе. Елемент може узимати вредности из неког домена уколико није дефинисан као празан елемент. Тип податка у елементу или атрибуту дефинише се помоћу атрибута *type*.

Међу типовима података у XML шеми могуће је разликовати атомичне типове података, листе и уније.

Атомични типови података су они чије вредности су недељиве. Могу бити примитивни или изведени. Примитивни типови су:

- string
- boolean
- decimal
- float
- double
- dutarion
- dateTime
- time
- date
- gYearMonth
- gYear
- gMonthDay
- gDay
- gMonth
- hexBinary
- base64Binary
- anyURI
- QName
- NOTATION

Изведени типови података су:

- normalizedString
- token
- language
- NMTOKEN
- NMTOKENS
- Name
- NCName
- ID
- IDREF
- IDREFS
- ENTITY
- ENTITIES
- Integer
- nonPositiveInteger
- negativeInteger
- long
- int
- short
- byte
- nonNegativeInteger
- unsignedLong
- unsignedInt
- unsignedShort
- unsignedByte
- positiveInteger

У наставку су дати описи и листа вредности и унија, али они неће бити коришћени у спецификацији ограничења домена, јер листа вредности може да садржи више вредности, што се коси с правилом прве нормалне форме да вредности буду једноелементне, а уније могу садржати вредности различитих типова података, што се не може дефинисати ограничењем домена.

Листе вредности су увек изведени типови података. Најчешће су представљене као колекције и садрже скуп коначног броја атомичних вредности. Тип податка елемената листе, дефинише се у оквиру атрибута itemType елемента xs:list у шеми. Листе имају вредности које се састоје од коначног низа (може бити и празан) вредности атомичних типова података.

У Листинг 54 дефинисана је листа вредности velicina која може да садржи децималне бројеве. Постоји само један елемент velicinaElemenata који је типа velicina. У XML документу садржај елемента velicinaElemenata је листа децималних бројева, а тај документ је дат у Листинг 55.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Lista">
  <xs:annotation>
   <xs:documentation>Comment describing your root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="velicinaElemenata" type="velicina" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:simpleType name="velicina">
  <xs:list itemType="xs:decimal"/>
 </xs:simpleType>
</xs:schema>
```

Листинг 54 XML Schema Lista

```
<?xml version="1.0" encoding="UTF-8"?>
<Lista                              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="MojaLista.xsd">
 <velicinaElemenata>2 4.5 3 2.56</velicinaElemenata>
</Lista>
```

Листинг 55 XML документ Lista

У Листинг 56 приказан је прост тип listaStringova, а елемент који ће бити тог типа може садржати више стрингова. Иако су у Листинг 57 стрингови написани у три реда (раздвојени знаком за крај реда), елемент elementi у ствари садржи 9 стрингова.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Lista">
  <xs:annotation>
   <xs:documentation>Comment describing your root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="elementi" type="listaStringova"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:simpleType name="listaStringova">
  <xs:list itemType="xs:string"/>
 </xs:simpleType>
</xs:schema>
```

Листинг 56 XML Schema Lista

```
<?xml version="1.0" encoding="UTF-8"?>
<Lista                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="MojaLista1.xsd">
 <elementi>
  element liste 1
  element liste 2
  element liste 3
 </elementi>
</Lista>
```

Листинг 57 XML документ Lista

У Листинг 58 елемент elementi нема дефинисан тип податка у шеми. Међутим, дефинисан је прост тип mojaSkracenaLista који се може користити у XML документу. Уколико се елемент elementi наведе без типа податка, могуће је унети било који садржај, као што је овде случај: abcd. Уколико се у XML документу наведе да је тип податка у елементу elementi баш mojaSkracenaLista, као у Листинг 59, тада садржај тог елемента може бити формиран само према утврђеном патерну, а који овде дозвољава унос цифара 123, иза којих следи произвољан број цифара и празнина, за којима се наводе цифре 456.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Lista3">
  <xs:annotation>
   <xs:documentation>Comment describing your root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="elementi" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:simpleType name="mojaLista">
  <xs:list itemType="xs:integer"/>
 </xs:simpleType>
 <xs:simpleType name="mojaSkracenaLista">
  <xs:restriction base="mojaLista">
   <xs:pattern value="123 (\d+\s)*456"/>
  </xs:restriction>
 </xs:simpleType>
</xs:schema>
```

Листинг 58 XML Schema Lista3

```
<?xml version="1.0" encoding="UTF-8"?>
<Lista3                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="MojalLista2.xsd">
 <elementi xsi:type="mojaSkracenaLista">123 456</elementi>
 <elementi xsi:type="mojaSkracenaLista">123 987 456</elementi>
 <elementi xsi:type="mojaSkracenaLista">123 4 5 6 456</elementi>
 <elementi>abcd</elementi>
</Lista3>
```

Листинг 59 XML документ Lista3

Унија је тип податка који представља унију вредности једног или више других типова података. У Листинг 60 приказан је елемент size који је дефинисан као унија вредности из домена xs:integer и xs:string. У XML документу у Листинг 61, као вредност елемента size могу се навести и бројеви и стрингови.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
```

```
 <xs:element name="Unija">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="size" maxOccurs="unbounded">
     <xs:simpleType>
      <xs:union memberTypes="xs:integer xs:string"/>
     </xs:simpleType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Листинг 60 XML Schema Unija

```
<?xml version="1.0" encoding="UTF-8"?>
<Unija                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Unija.xsd">
 <size>1</size>
 <size>large</size>
</Unija>
```

Листинг 61 XML документ Unija

Примитивни типови података су они који се не дефинишу уз помоћ других типова података, док су изведени они који се дефинишу преко неког другог типа података и то помоћу рестрикције, листе или уније. На пример, float је тип података који се не може дефинисати уз помоћ других типова података, док је тип података integer специјалан случај општијег типа података decimal и изведен је из њега.

Тип података је изведен рестрикцијом из другог типа података када се користе фасети за ограничавање вредности обележја која припадају том типу података. Вредност обележја припада основном типу података (base type), који може бити примитивни или изведен. У зависности од основног типа података, примењују се неки од наведених фасета:

- енумерације (enumeration), чиме се дефинише листа дозвољених вредности,
- број децимала (fractionDigits), чиме се дефинише максималан дозвољен број децималних места,
- дужина (length), чиме се дефинише тачан број карактера или елемената листе,
- maxExclusive, дефинише се горња граница за нумеричке вредности, при чему вредност мора бити мања од максимално дозвољене,
- maxInclusive, дефинише се горња граница за нумеричке вредности, при чему вредност може бити мања или једнака максимално дозвољеној,
- minExclusive, дефинише се доња граница за нумеричке вредности, при чему вредност мора бити већа од минимално дозвољене,
- minInclusive, дефинише се доња граница за нумеричке вредности, при чему вредност може бити већа или једнака минимално дозвољеној,
- maxLength, дефинише се максималан број карактера или елемената листе,
- minLength, дефинише се минималан број карактера или елемената листе,
- pattern, дефинише се тачан опис дозвољене вредности,
- totalDigits, дефинише се тачан број дозвољених цифара за нумеричке вредности,
- whiteSpace, дефинише се начин обраде празнина (white space).

Тип података је изведен помоћу листе вредности неког другог типа података (itemType) тако што се креира коначна листа вредности тог типа података. Тип података је изведен помоћу уније тако што се врши унија вредности једног или више типова података.

Уграђени типови података су они коју се дефинисани у спецификацији XML шеме и могу бити примитивни или изведени. Кориснички дефинисани типови података су изведени типови података које дефинише корисник.

Да би шема релације у релационом моделу података била у првој нормалној форми, сва обележја из универзалног скупа обележја морају бити елементарна. Елементарна обележја су она чији домен садржи само атомичне вредности, што значи да не представљају торке реда већег од 1, или се даља структура таквог елемента више не декомпонује. То значи да је обележје елементарно ако се не може даље растављати на компоненте које такође представљају обележја или ако посматрана примена не захтева декомпозицију.

За потребе ове дисертације, претпоставка је да за типове података треба бирати просте типове (simpleType) или неки од уграђених примитивних типова података или уграђених изведених типова података рестрикцијом. Не би требало користити комплексне типове података (complexType), који дефинишу поделеменете неког елемента и тако праве хијерархијску структуру, и листе вредности код којих у оквиру једног елемента постоји листа вредности неког другог типа податка. Унија дефинише елементе који могу да садрже вредности више различитих типова података. Такви типови података не могу учествовати у дефиницији ограничења кључа и референцијалног интегритета, а слично тако ни код осталих типова ограничења.

У Листинг 62 елемент Adresa садржи поделементе Ulica, Broj, PostanskiBroj и Mesto. Елемент Adresa је комплексни тип, па не задовољава услов да сви атрибути или елементи у XML шеми морају бити елементарни. У Листинг 63 дат је XML документ валидан у односу на шему из Листинг 62.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Osoba">
  <xs:annotation>
   <xs:documentation>Comment describing your root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Ime" type="xs:string"/>
    <xs:element name="Prezime" type="xs:string"/>
    <xs:element name="Adresa">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="Ulica" type="xs:string"/>
       <xs:element name="Broj" type="xs:positiveInteger"/>
       <xs:element name="PostanskiBroj">
        <xs:simpleType>
         <xs:restriction base="xs:positiveInteger">
          <xs:totalDigits value="5"/>
         </xs:restriction>
        </xs:simpleType>
       </xs:element>
       <xs:element name="Mesto" type="xs:string"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
    <xs:element name="BrojTelefona">
     <xs:complexType>
      <xs:sequence>
       <xs:element name="PozivniBroj">
        <xs:simpleType>
         <xs:restriction base="xs:string">
          <xs:pattern value="0[1-9][0-9]"/>
         </xs:restriction>
```

```
        </xs:simpleType>
       </xs:element>
       <xs:element name="Broj" type="xs:positiveInteger"/>
      </xs:sequence>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Листинг 62 XML Schema Osoba

```
<?xml version="1.0" encoding="UTF-8"?>
<Osoba                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="KompleksniElementi.xsd">
 <Ime>Ana</Ime>
 <Prezime>Antic</Prezime>
 <Adresa>
  <Ulica>Cvetna</Ulica>
  <Broj>25</Broj>
  <PostanskiBroj>21000</PostanskiBroj>
  <Mesto>Novi Sad</Mesto>
 </Adresa>
 <BrojTelefona>
  <PozivniBroj>021</PozivniBroj>
  <Broj>1234546</Broj>
 </BrojTelefona>
</Osoba>
```

Листинг 63 XML документ Osoba

Према усвојеној структури XML документа у овој дисертацији, сви елементи су прости и немају вредности, а за сваки атрибут се дефинишу типови података. Ограничење домена у XML моделу података је семантички богатије од домена у релационом моделу података, јер XML Schema пружа веће могућности за дефинисање ограничења домена него релациони модел података.

Пример 1

Према спецификацији ограничења домена датој у одељку 3.1.1, дефинисано је ограничење домена DomenOcena. Обележја која припадају домену DomenOcena, могу бити само цели бројеви између 5 и 10:

DomenOcena(DomCon, id(DomenOcena)(d)=(Integer, 2, d≥5 i d≤10), Ø).

Како у XML Schema-и постоји тип података Integer, ово ограничење се лако може имплементирати. Ограничење да су вредности обележја из тог домена између 5 и 10, може се реализовати на неколико начина, на пример, набрајањем вредности помоћу енумерације (Листинг 64) или ограничавањем опсега (Листинг 65).

```
<xs:element name="Ocena">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:enumeration value="5"/>
   <xs:enumeration value="6"/>
   <xs:enumeration value="7"/>
   <xs:enumeration value="8"/>
   <xs:enumeration value="9"/>
   <xs:enumeration value="10"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Листинг 64 Рестрикција елемента Ocena помоћу енумерација

```
<xs:element name="Ocena">
 <xs:simpleType>
  <xs:restriction base="xs:integer">
   <xs:minInclusive value="5"/>
   <xs:maxInclusive value="10"/>
  </xs:restriction>
 </xs:simpleType>
</xs:element>
```

Листинг 65 Рестрикција елемента Ocena помоћу ограничавања вредности

Пример 2

Домен неког атрибута се може имплементирати и као simpleType, у којем је наведено ограничење домена. Ограничење домена DomenPrezime, према спецификацији датој у одељку 3.1.1, приказано је на следећи начин:

DomenPrezime(DomCon, id(DomenPrezime)(d)=(String, 30, /), Ø)

У Листинг 66 дата је имплементација ограничења домена помоћу simpleType елемента, а у Листинг 67 и пример како се овако дефинисан домен користи у атрибуту.

```
<xs:simpleType name="Prezime">
  <xs:restriction base="xs:string">
   <xs:maxLength value="30"/>
</xs:restriction>
```

Листинг 66 Ограничење домена помоћу елемента simpleType

```
<xs:attribute  name="prezime" type="Prezime"/>
```

Листинг 67 Тип атрибута

5.1.2 Ограничење вредности атрибута

Ограничење вредности атрибута дефинише услов који атрибут из неког домена мора да задовољи. Поред тога, дефинише и да ли су за атрибут дозвољене или забрањене нула вредности. Ово ограничење је већ подржано у XML Schema-и. Спецификација овог ограничења дата је у одељку 3.1.2.

Пример 3

Ограничење вредности атрибута се дефинише за сваки атрибут неког елемента. У следећем примеру приказан је елемент Osoba, који има атрибуте Ime, Prezime, Pol, DatumRodjenja, Zaposlen, DatumZaposlenja и Plata. Сваки од ових атрибута припада одређеном домену, а поред тога, мора се навести да ли су дозвољене или забрањене нула вредности.

Ограничења домена су:

DomenIme(DomCon, id(DomenIme)(d)=(String, 30, /), Ø)
DomenPrezime(DomCon, id(DomenPrezime)(d)=(String, 30, /), Ø)
DomenPol(DomCon, id(DomenPol)(d)=(String, 1, d∈{M, Z}), Ø)
DomenDatumRodjenja(DomCon, id(DomenDatumRodjenja)(d)=(Date, /, /), Ø)
DomenZaposlen(DomCon, id(DomenZaposlen)(d)=(Boolean, /, /), Ø)
DomenDatumZaposlenja(DomCon, id(DomenDatumZaposlenja)(d)=(Date, /, /), Ø)
DomenPlata(DomCon, id(DomenPlata)(d)=(Decimal, 8, d≥20000 i d≤100000), Ø)

Ограничења вредности обележја су:

AttValConIme(AttValCon, τ(Osoba, Ime) = (DomenIme, NotNull))
AttValConPrezime(AttValCon, τ(Osoba, Prezime) = (DomenPrezime, NotNull))

AttValConPol(AttValCon, τ(Osoba, Pol) = (DomenPol, NotNull))
AttValConDatumRodjenja(AttValCon, τ(Osoba, DatumRodjenja) = (DomenDatumRodjenja, NotNull))
AttValConZaposlen(AttValCon, τ(Osoba, Zaposlen) = (DomenZaposlen, NotNull))
AttValConDatumZaposlenja(AttValCon, τ(Osoba, DatumZaposlenja) = (DomenDatumZaposlenja, Null))
AttValConPlata(AttValCon, τ(Osoba, Plata) = (DomenPlata, Null))

У Листинг 68 приказан је елемент Особа чији атрибути припадају атомичним типовима података. Ако је нула вредност забрањена за неки атрибут, за њега се наводи да употреба обавезна (required), а ако је нула вредност дозвољена, онда је употреба опционална (optional), а у XML документу се тај атрибут не наводи. У Листинг 69 и Листинг 70 дати су XML документи формирани у односу на шему из Листинг 68, у којем се виде конкретне вредности наведених типова података.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Osoba">
  <xs:annotation>
   <xs:documentation>Comment describing your root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:attribute name="Ime" use="required">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:maxLength value="30"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="Prezime" use="required">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:maxLength value="30"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="Pol" use="required">
    <xs:simpleType>
     <xs:restriction base="xs:string">
      <xs:enumeration value="M"/>
      <xs:enumeration value="Z"/>
     </xs:restriction>
    </xs:simpleType>
   </xs:attribute>
   <xs:attribute name="DatumRodjenja" type="xs:date" use="required"/>
   <xs:attribute name="Zaposlen" type="xs:boolean" use="required"/>
   <xs:attribute name="DatumZaposlenja" type="xs:gYearMonth" use="optional"/>
   <xs:attribute name="Plata" type="xs:decimal" use="optional"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Листинг 68 XML Schema Osoba

```
<?xml version="1.0" encoding="UTF-8"?>
<Osoba                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Osoba.xsd"
  Ime="Ana"
  Prezime="Antic"
  Pol="Z"
  DatumRodjenja="1980-06-01"
  Zaposlen="true"
  DatumZaposlenja="2010-05"
  Plata="55000.00"/>
```

Листинг 69 XML документ Osoba

```
<?xml version="1.0" encoding="UTF-8"?>
<Osoba                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Osoba.xsd"
 Ime="Marko"
 Prezime="Maric"
 Pol="M"
 DatumRodjenja="1990-02-02"
 Zaposlen="false"/>
```

Листинг 70 XML документ Osoba

5.1.3 Ограничење јединствене вредности

Ограничење јединствене вредности обележја се дефинише за атрибуте који имају јединствену вредност на нивоу типа елемента у XML документу. Дефинише се помоћу елемента xs:unique. Овај елемент има два поделемента: xs:selector, који дефинише тип елемента на који се примењује ограничење, чиме уједно дефинише и опсег ограничења, и xs:field, који одређује атрибут унутар претходно одређеног типа, на које се примењује ограничење. Вредност тог атрибута ће бити јединствена. Спецификација овог ограничења дата је у одељку 3.1.6.

Атрибути у XML Schema-и не могу имати декларисане нула вредности. Могуће је само одредити да ли атрибут мора или не мора да се појави у XML документу. Ако се појављује, свака вредност тог атрибута мора бити јединствена. Ово је недостатак саме XML Schema спецификације.

Пример 4

Посматрајмо елемент Student, који има атрибуте brojIndeksa, maticniBroj, ime и prezime. Атрибут maticniBroj има јединствене вредности у XML документу. Ограничење јединствене вредности је задато на следећи начин:

Unique(Student, maticniBroj)

У Листинг 71, елемент Student има кључ елемент brojIndeksa. Такође, матични број студента, maticniBroj, има јединствене вредности за сваког студента. Испод коренског елемента је дефинисан елемент xs:unique, који се односи на елемент maticniBroj. Како је елемент maticniBroj типа MBR, његова укупна дужина је тачно 13 цифара, па вредност елемента maticniBroj у одговарајућем XML документу не сме имати нула вредности.

Међутим, ако је потребно извршити и проверу да ли матични број задовољава и неки тачно утврђен формат, онда је потребно направити функцију која ће унету вредност проверавати. То је описано у одељку 4.2.1 јер се слична ситуација јавља и код ограничења торке.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:simpleType name="BrojIndeksa">
  <xs:restriction base="xs:string">
   <xs:pattern value="[A-Z]{2}-[0-9]{3}"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="MBR">
  <xs:restriction base="xs:integer">
   <xs:totalDigits value="13"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="Ime">
  <xs:restriction base="xs:string">
```

```
    <xs:maxLength value="30"/>
   </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Prezime">
   <xs:restriction base="xs:string">
    <xs:maxLength value="30"/>
   </xs:restriction>
  </xs:simpleType>
  <xs:element name="Studenti">
   <xs:complexType>
    <xs:sequence>
     <xs:element name="Student" maxOccurs="unbounded">
      <xs:complexType>
       <xs:attribute name="brojIndeksa" type="BrojIndeksa" use="required"/>
       <xs:attribute name="maticniBroj" type="MBR" use=" optional"/>
       <xs:attribute name="ime" type="Ime" use="required"/>
       <xs:attribute name="prezime" type="Prezime" use="required"/>
      </xs:complexType>
     </xs:element>
    </xs:sequence>
   </xs:complexType>
   <xs:key name="StudentPK">
    <xs:selector xpath="Student"/>
    <xs:field xpath="@brojIndeksa"/>
   </xs:key>
   <xs:unique name="StudentUN">
    <xs:selector xpath="Student"/>
    <xs:field xpath="@maticniBroj"/>
   </xs:unique>
  </xs:element>
</xs:schema>
```

Листинг 71 XML Schema Studenti

Документ који је валидан у односу на шему дат је у Листинг 72. Сваки матични број мора имати јединствену вредност. Како је атрибут maticniBroj опционалан, може и да се не наведе, а све остале вредности тог атрибута морају бити јединствене.

```
<?xml version="1.0" encoding="UTF-8"?>
<Studenti                      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="StudentiAtt.xsd">
 <Student brojIndeksa="IT-123" prezime="Peric" ime="Pera"
          maticniBroj="1234567891234"/>
 <Student brojIndeksa="IT-124" prezime="Maric" ime="Mira"/>
 <Student brojIndeksa="IT-125" prezime="Savic" ime="Sava"/>
 <Student brojIndeksa="IT-126" prezime="Mitic" ime="Ana"
          maticniBroj="1234567891235"/>
</Studenti>
```

Листинг 72 XML документ Studenti

5.2 Имплементација ограничења за која се предлаже проширење XML Schema-е

У овом одељку приказана су ограничења која постоје у XML Schema-и, али не поседују исту функционалност као у релационим СУБП. То су:

- ограничење торке,
- ограничење кључа и
- ограничење референцијалног интегритета,

чија је спецификација дата у одељцима 3.1.3, 3.1.5 и 3.1.7, редом. У овим одељцима предложено је проширење XML Schema-е, на основу којег генератор кода, описан у поглављу

4, генерише XQuery функције и тригере, који у потпуности врше контролу наведених ограничења.

5.2.1 Ограничење торке

У релационом моделу података ограничење торке је ограничење вредности обележја на нивоу једне торке, у појави над шемом релације. Дефинише се за свако обележје сваке шеме релације. Обавезне компоненте овог ограничења су ограничење домена и дозвола или забрана нула вредности. Опционална компонента је логички услов који повезује вредности различитих обележја у истој торци. Дефинисањем назива домена у оквиру ограничења торке врши се придруживање домена обележју, што даље значи да обележје не може узимати вредност која не задовољава ограничење домена. Провера важења ограничења торке спроводи се за сваку торку релације посебно. Уколико се дефинише логички услов, он се за сваку торку над скупом обележја интерпретира као тачан или нетачан.

У XML моделу података, ограничење торке се односи на ограничење које сваки атрибут из одређеног елемента мора да задовољи. Према усвојеном изгледу XML документа, ограничење торке се дефинише за атрибуте једног елемента. За сваки атрибут се проверава да ли задовољава ограничење домена, да ли има дозвољене или забрањене нула вредности, то јест да ли је обавезан или не, и, уколико постоји, да ли задовољава неки додатни логички услов. Ограничење торке се проверава тако што се у оквиру шеме задаје регуларни израз који ограничава вредности обележја неког типа, или се задаје назив XQuery функције, чијим израчунавањем се проверава да ли вредности обележја задовољавају услов задат у функцији. У пракси се појављује потреба за таквим логичким условом, који се не може исказати у постојећој XML Schema-и. Због тога дат је предлог проширења XML Schema-е које може да обухвати и такве случајеве. Спецификација ограничења торке дата је у одељку 3.1.3.

Пример 5

Скуп атрибута елемента Radnik обележен је са R. Један од атрибута елемента Radnik је jmbg, који треба да буде формиран према тачно утврђеном правилу. Ограничење торке се задаје на следећи начин:

TupleConRadnik(TupleCon, τ(Radnik)=({τ(Radnik, A) | A∈R}, checkJMBG))

где су ограничења вредности обележја:

AttValConIme(AttValCon, τ(Radnik, Ime) = (DomenIme, NotNull))
AttValConPrezime(AttValCon, τ(Radnik, Prezime) = (DomenPrezime, NotNull))
AttValConJMBG(AttValCon, τ(Radnik, JMBG) = (DomenJMBG, NotNull))
AttValConJMBG(AttValCon, τ(Radnik, Datrodj) = (DomenDatrodj, NotNull))

Ограничење домена за атрибуте ime и prezime су већ дефинисани, а ограничење домена за атрибут jmbg дато је на следећи начин:

DomenJMBG(DomCon, id(DomenJMBG)(d)=(String, 13, /), Ø).

Ограничење домена за атрибут datrodj дато је на следећи начин:

DomenDatrodj(DomCon, id(DomenDatrodj)(d)=(Date, /, /), Ø).

Функција која проверава исправност унете вредности атрибута jmbg зове се checkJMBG. Јединствени матични број грађанина добија се на основу датума рођења, тако што су прве две цифре добијене из дана у датуму рођења, друге две цифре добијене су из месеца у датуму

рођења, а наредне три цифре из године у датуму рођења. Валидност се проверава на основу поклапања датума рођења из торке и из вредности атрибута jmbg, као према алгоритму за проверу контролне (последње) цифре у атрибуту jmbg, а који је дат у XQuery функцији checkJMBG. Шема је проширена елементом xc:constraint који садржи услов xc:contidion. У услову је дат XQuery израз у којем је позвана функција checkJMBG, која проверава да ли је вредност атрибута jmbg формирана према правилу.

У Листинг 73 дат је XML документ Radnici који садржи елементе Radnik. Сваки елемент Radnik има атрибуте jmbg, ime, prezime и datrodj. Листинг 74 приказује XML Schema документ који документ Radnici, а који је проширен изразом за проверу вредности атрибута jmbg.

```
<Radnici>
    <Radnik jmbg="1509010800041" ime="Marko" prezime="Markovic"
            datrodj="2010-09-15"/>
</Radnici>
```

Листинг 73 XML документ Radnici

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Radnici">
  <xs:annotation>
   <xs:documentation>Comment describing your root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Radnik" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="jmbg" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:length value="13"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="ime" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:maxLength value="30"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="prezime" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:maxLength value="30"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="datrodj" type="xs:date" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:key name="Radnik_PK">
   <xs:selector xpath="Radnik"/>
   <xs:field xpath="@jmbg"/>
  </xs:key>
  <xc:constraint type="xc:tupleCon">
   <xc:condition from="/Radnici/Radnik" function="checkJMBG"/>
  </xc:constraint>
 </xs:element>
</xs:schema>
```

Листинг 74 XML Schema документ Radnici

5.2.1.1 Контрола ограничења торке помоћу XQuery функција

Када се додаје нови радник, прво се функцијом checkJMBG проверава да ли вредност атрибута jmbg задовољава правило, а ако је то тачно, функцијом doInsertRadnik се нови елемент Radnik додаје испред свих постојећих. Додавање новог радника помоћу XQuery функције описано је у Листинг 75.

```
declare function local:checkJMBG($NEW as element(Radnik))
    as xs:boolean{
        let $year := if(xs:integer(fn:substring($NEW/@jmbg, 5, 1)) = 0)
                        then fn:concat('2', fn:substring($NEW/@jmbg, 5, 3))
                        else fn:concat('1', fn:substring($NEW/@jmbg, 5, 3))
        let $controlD := fn:year-from-date($NEW/@datrodj) = xs:integer($year)
                            and fn:month-from-date($NEW/@datrodj) =
xs:integer(fn:substring($NEW/@jmbg, 3, 2))
                            and fn:day-from-date($NEW/@datrodj) =
xs:integer(fn:substring($NEW/@jmbg, 1, 2))
        let $m := xs:integer(fn:substring($NEW/@jmbg, 13, 1))
        let $control := 11 - ( 7*(xs:integer(fn:substring($NEW/@jmbg, 1, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 7, 1))) +
                                  6*(xs:integer(fn:substring($NEW/@jmbg, 2, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 8, 1))) +
                                  5*(xs:integer(fn:substring($NEW/@jmbg, 3, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 9, 1))) +
                                  4*(xs:integer(fn:substring($NEW/@jmbg, 4, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 10, 1))) +
                                  3*(xs:integer(fn:substring($NEW/@jmbg, 5, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 11, 1))) +
                                  2*(xs:integer(fn:substring($NEW/@jmbg, 6, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 12, 1))))
                           mod 11
        return if($control<10)
               then ($m = $control and $controlD = true())
               else ($m = 0 and $controlD = true())
};

declare function local:doInsertRadnik($NEW as element(Radnik))
    as xs:boolean{
        let $doInsert := update insert $NEW preceding
doc('RadnikJMBG.xml')/Radnici/Radnik[1]
    return true()
};

declare function local:insertRadnik($NEW as element(Radnik))
    as xs:boolean{
        let $canInsert := if (local:checkJMBG($NEW))
            then local:doInsertRadnik($NEW)
            else false()
    return $canInsert
};
```

Листинг 75 XQuery функција за додавање новог радника

Када се врши измена вредности неког од атрибута у елементу Radnik, потребно је стари елемент заменити новим, који садржи нове вредности неких атрибута. Функцијом checkJMBG проверава се да ли је вредност атрибута jmbg у новом елементу задовољава услов, наведен на почетку овог примера. XQuery функција за промену елемента Radnik дата је у Листинг 76.

```
declare function local:checkJMBG($NEW as element(Radnik))
    as xs:boolean{
        let $year := if(xs:integer(fn:substring($NEW/@jmbg, 5, 1)) = 0)
                        then fn:concat('2', fn:substring($NEW/@jmbg, 5, 3))
                        else fn:concat('1', fn:substring($NEW/@jmbg, 5, 3))
        let $controlD := fn:year-from-date($NEW/@datrodj) = xs:integer($year)
                            and fn:month-from-date($NEW/@datrodj) =
xs:integer(fn:substring($NEW/@jmbg, 3, 2))
```

```
                            and fn:day-from-date($NEW/@datrodj) =
xs:integer(fn:substring($NEW/@jmbg, 1, 2))
        let $m := xs:integer(fn:substring($NEW/@jmbg, 13, 1))
        let $control := 11 - ( 7*(xs:integer(fn:substring($NEW/@jmbg, 1, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 7, 1))) +
                                  6*(xs:integer(fn:substring($NEW/@jmbg, 2, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 8, 1))) +
                                  5*(xs:integer(fn:substring($NEW/@jmbg, 3, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 9, 1))) +
                                  4*(xs:integer(fn:substring($NEW/@jmbg, 4, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 10, 1))) +
                                  3*(xs:integer(fn:substring($NEW/@jmbg, 5, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 11, 1))) +
                                  2*(xs:integer(fn:substring($NEW/@jmbg, 6, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 12, 1))))
                           mod 11
        return if($control<10)
               then ($m = $control and $controlD = true())
               else ($m = 0 and $controlD = true())
};

declare function local:doUpdateRadnik($OLD as element(Radnik), $NEW as
element(Radnik))
    as xs:boolean{
        let $doInsert := update replace
doc('RadnikJMBG.xml'))/Radnici/Radnik[@jmbg=$OLD/@jmbg] with $NEW
    return true()
};

declare function local:updateRadnik($OLD as element(Radnik), $NEW as
element(Radnik))
    as xs:boolean{
        let $canUpdate := if (local:checkJMBG($NEW))
            then local:doUpdateRadnik($OLD, $NEW)
            else false()
    return $canUpdate
};
```

Листинг 76 XQuery функција за промену елемента Radnik

5.2.1.2 Контрола ограничења торке помоћу тригера

Израз за проверу вредности атрибута jmbg налази се у телу DO петље тригера који се извршава пре додавања новог радника. Тригер VrednostObelezjaBeforeInsert дат је у Листинг 77. Тренутна имплементација тригера не допушта позиве функција, као ни употребу локалних променљивих. Због тога је садржај функције checkJMBG додат у спољашњу if-then-else грану, а сама функција не користи локалне променљиве.

```
CREATE TRIGGER "VrednostObelezjaBeforeInsert"
BEFORE INSERT
ON collection('VrednostObelezja')/Radnici/Radnik
FOR EACH NODE
DO {
 if(if(11 - ( 7*(xs:integer(fn:substring($NEW/@jmbg, 1, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 7, 1))) +
                                  6*(xs:integer(fn:substring($NEW/@jmbg, 2, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 8, 1))) +
                                  5*(xs:integer(fn:substring($NEW/@jmbg, 3, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 9, 1))) +
                                  4*(xs:integer(fn:substring($NEW/@jmbg, 4, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 10, 1))) +
                                  3*(xs:integer(fn:substring($NEW/@jmbg, 5, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 11, 1))) +
                                  2*(xs:integer(fn:substring($NEW/@jmbg, 6, 1))+
                                  xs:integer(fn:substring($NEW/@jmbg, 12, 1))))
                           mod 11<10))
```

```
     then (xs:integer(fn:substring($NEW/@jmbg, 13, 1)) = 11 -
         (7*(xs:integer(fn:substring($NEW/@jmbg, 1, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 7, 1))) +
                              6*(xs:integer(fn:substring($NEW/@jmbg, 2, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 8, 1))) +
                              5*(xs:integer(fn:substring($NEW/@jmbg, 3, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 9, 1))) +
                              4*(xs:integer(fn:substring($NEW/@jmbg, 4, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 10, 1))) +
                              3*(xs:integer(fn:substring($NEW/@jmbg, 5, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 11, 1))) +
                              2*(xs:integer(fn:substring($NEW/@jmbg, 6, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 12, 1))))
                         mod 11 and
        (fn:year-from-date($NEW/@datrodj) =
        xs:integer(if(xs:integer(fn:substring($NEW/@jmbg, 5, 1)) = 0)
                   then fn:concat('2', fn:substring($NEW/@jmbg, 5, 3))
                   else fn:concat('1', fn:substring($NEW/@jmbg, 5, 3)))
        and fn:month-from-date($NEW/@datrodj) =
               xs:integer(fn:substring($NEW/@jmbg, 3, 2))
        and fn:day-from-date($NEW/@datrodj) =
               xs:integer(fn:substring($NEW/@jmbg, 1, 2))) = true())
     else (xs:integer(fn:substring($NEW/@jmbg, 13, 1)) = 0
         and (fn:year-from-date($NEW/@datrodj) =
              xs:integer(if(xs:integer(fn:substring($NEW/@jmbg, 5, 1)) = 0)
                         then fn:concat('2', fn:substring($NEW/@jmbg, 5, 3))
                         else fn:concat('1', fn:substring($NEW/@jmbg, 5, 3)))
         and fn:month-from-date($NEW/@datrodj) =
              xs:integer(fn:substring($NEW/@jmbg, 3, 2))
         and fn:day-from-date($NEW/@datrodj) =
              xs:integer(fn:substring($NEW/@jmbg, 1, 2))) = true())
 then
  ($NEW)
 else
  error(xs:QName("VrednostObelezjaBeforeInsert"),"Neispravan JMBG");
}
```

Листинг 77 Тригер VrednostObelezjaBeforeInsert

Унос радника у Листинг 78 је могућ, јер вредност атрибута jmbg задовољава услов у тригеру.

```
UPDATE INSERT
<Radnik jmbg="1509010800050" ime="Milan" prezime="Milic" datrodj="2010-09-15"/>
INTO fn:doc("RadnikJMBG", "VrednostObelezja")/Radnici
```

Листинг 78 Успешно додавање новог радника

Унос радника у Листинг 79 је није могућ, јер вредност атрибута jmbg не задовољава услов у тригеру.

```
UPDATE INSERT
<Radnik jmbg="1509010800042" ime="Milan" prezime="Milic" datrodj="2010-09-16"/>
INTO fn:doc("RadnikJMBG", "VrednostObelezja")/Radnici
```

Листинг 79 Неуспешно додавање новог радника

Промена вредности атрибута jmbg такође се контролише помоћу тригера, који се покреће пре те промене. Тригер VrednostObelezjaBeforeUpdate дат је Листинг 80. Објашењење је аналогно објашњењу у претходном примеру, па је овде наведен само код тригера и пример модификације елемента Radnik. Такође, могуће је променити и вредности осталих атрибута који нису кључеви, а у овом примеру то су атрибути ime и prezime радника.

```
CREATE TRIGGER "VrednostObelezjaBeforeUpdate"
BEFORE REPLACE
ON collection('VrednostObelezja')/Radnici/Radnik
FOR EACH NODE
```

```
DO {
 if(if(11 - ( 7*(xs:integer(fn:substring($NEW/@jmbg, 1, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 7, 1))) +
                              6*(xs:integer(fn:substring($NEW/@jmbg, 2, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 8, 1))) +
                              5*(xs:integer(fn:substring($NEW/@jmbg, 3, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 9, 1))) +
                              4*(xs:integer(fn:substring($NEW/@jmbg, 4, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 10, 1))) +
                              3*(xs:integer(fn:substring($NEW/@jmbg, 5, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 11, 1))) +
                              2*(xs:integer(fn:substring($NEW/@jmbg, 6, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 12, 1))))
                         mod 11<10))
     then (xs:integer(fn:substring($NEW/@jmbg, 13, 1)) = 11 -
         (7*(xs:integer(fn:substring($NEW/@jmbg, 1, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 7, 1))) +
                              6*(xs:integer(fn:substring($NEW/@jmbg, 2, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 8, 1))) +
                              5*(xs:integer(fn:substring($NEW/@jmbg, 3, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 9, 1))) +
                              4*(xs:integer(fn:substring($NEW/@jmbg, 4, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 10, 1))) +
                              3*(xs:integer(fn:substring($NEW/@jmbg, 5, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 11, 1))) +
                              2*(xs:integer(fn:substring($NEW/@jmbg, 6, 1))+
                              xs:integer(fn:substring($NEW/@jmbg, 12, 1))))
                         mod 11 and
        (fn:year-from-date($NEW/@datrodj) =
        xs:integer(if(xs:integer(fn:substring($NEW/@jmbg, 5, 1)) = 0)
                    then fn:concat('2', fn:substring($NEW/@jmbg, 5, 3))
                    else fn:concat('1', fn:substring($NEW/@jmbg, 5, 3)))
        and fn:month-from-date($NEW/@datrodj) =
               xs:integer(fn:substring($NEW/@jmbg, 3, 2))
        and fn:day-from-date($NEW/@datrodj) =
               xs:integer(fn:substring($NEW/@jmbg, 1, 2))) = true())
     else (xs:integer(fn:substring($NEW/@jmbg, 13, 1)) = 0
         and (fn:year-from-date($NEW/@datrodj) =
              xs:integer(if(xs:integer(fn:substring($NEW/@jmbg, 5, 1)) = 0)
                         then fn:concat('2', fn:substring($NEW/@jmbg, 5, 3))
                         else fn:concat('1', fn:substring($NEW/@jmbg, 5, 3)))
         and fn:month-from-date($NEW/@datrodj) =
              xs:integer(fn:substring($NEW/@jmbg, 3, 2))
         and fn:day-from-date($NEW/@datrodj) =
              xs:integer(fn:substring($NEW/@jmbg, 1, 2))) = true())
 then
  ($NEW)
 else
  error(xs:QName("VrednostObelezjaBeforeUpdate"),"Neispravan JMBG");
}
```

Листинг 80 Тригер VrednostObelezjaBeforeUpdate

```
UPDATE REPLACE $r in
  fn:doc("RadnikJMBG", "VrednostObelezja")/Radnici/Radnik[@jmbg="1509010800050"]
with
   <Radnik jmbg="1509010800041" ime="Milan1" prezime="Milic1"
           datrodj="2010-09-15"/>
```

Листинг 81 Успешна замена елемента Radnik новим

5.2.2 Ограничење кључа

Ограничење кључа обезбеђује јединствену идентификацију сваког елемента у оквиру XML документа. Приликом дефинисања кључа битно је одредити опсег у којем тај кључ важи. У релационом моделу података, опсег важења кључа је релација. У XML документу опсег важења кључа може бити цео документ или неки његов део, и у складу са тим дефинисани су

апсолутни и релативни кључеви. У овој дисертацији је усвојено да је опсег важења кључа цео документ. Кључ је, према правилу за формирање XML документа у овој дисертацији, скуп атрибута неког елемента.

Кључ се може дефинисати коришћењем атрибута ID. У Листинг 82 атрибут mbr представља кључ елемента Radnik. Тип овог атрибута је xs:ID. На овај начин, кључ може бити само једновредносан, то јест, може садржати само један атрибут. Опсег важења овако дефинисаног кључа је цео документ. Вредност обележја које представља кључ обавезно мора почињати словом. Како и елемент Projekat има кључ који је типа ID, вредности атрибута mbr и idp морају бити различите, иако се не налазе у истој класи елемената.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Root">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Radnik" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="mbr" type="xs:ID" use="required"/>
      <xs:attribute name="ime" type="xs:string" use="required"/>
      <xs:attribute name="prezime" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Projekat">
     <xs:complexType>
      <xs:attribute name="idp" type="xs:ID" use="required"/>
      <xs:attribute name="naziv" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

Листинг 82 XML Schema документ Root - кључеви као ID атрибути

У наставку су дата два XML документа формирана у односу на шему из примера. Први XML документ је валидан (Листинг 83), јер у сваком елементу Radnik атрибут mbr има јединствену вредност. Вредност кључа елемента Projekat се разликује од свих вредности кључева елемената Radnik.

```
<?xml version="1.0" encoding="UTF-8"?>
<Root                           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="KljucID.xsd">
 <Radnik mbr="R1" prezime="Petrovic" ime="Petar"/>
 <Radnik prezime="Markovic" mbr="R2" ime="Marko"/>
 <Radnik prezime="Ilic" mbr="R3" ime="Ilija"/>
 <Radnik prezime="Savic" mbr="R4" ime="Sava"/>
</Root>
```

Листинг 83 Валидан XML документ

Други XML документ није валидан (Листинг 84), јер је у четвртом елементу Radnik поновљена већ постојећа вредности атрибута mbr. Други проблем у овом документу је што један од радника има вредност кључа 5, а вредност кључа би морала да почиње словом. Такође, проблем је и што је вредност кључа елемента Projekat већ постојећа вредност кључа елемента Radnik, иако та два елемента нису никако повезана.

```
<?xml version="1.0" encoding="UTF-8"?>
<Root                           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="KljucID.xsd">
 <Radnik mbr="R1" prezime="Petrovic" ime="Petar"/>
```

```
 <Radnik prezime="Markovic" mbr="R2" ime="Marko"/>
 <Radnik prezime="Ilic" mbr="R3" ime="Ilija"/>
 <Radnik prezime="Savic" mbr="R3" ime="Sava"/>
 <Radnik prezime="Antic" mbr="5" ime="Ana"/>
 <Projekat idp="P1" naziv="R1"/>
</Root>
```

Листинг 84 Невалидан XML документ

Такође, кључ је могуће дефинисати и помоћу елемента xs:key. У том случају не треба навести и да је атрибут који представља кључ типа xs:ID. Овакав начин дефинисања кључа је једини могућ уколико се кључ састоји од више атрибута. Један елемент не може садржати више поделемената или атрибута који су типа xs:ID.

Елемент xs:key има два поделемента и оба поделемента имају атрибут xpath. У елементу xs:selector задаје се путања од корена до чвора под којим се налази кључ, али се назив коренског елемента не наводи у тој путањи. Он дефинише тип елемента на који се примењује ограничење, чиме се одређује опсег ограничења. У атрибуту xpath елемента xs:field наводи се атрибут који представља кључ. Испред назива атрибута у xpath изразу xs:field елемента, наводи се знак @. Уколико елемент има више атрибута у кључу, тада се у xs:key елементу наводи онолики број поделемената xs:field колико је атрибута у кључу.

У Листинг 85, за елемент Radnik дефинисан је кључ помоћу елемента xs:key. Кључ је атрибут mbr елемента Radnik.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Root">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Radnik" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="mbr" type="xs:string" use="required"/>
      <xs:attribute name="ime" type="xs:string" use="required"/>
      <xs:attribute name="prezime" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Projekat" maxOccurs="unbounded">
      <xs:complexType>
       <xs:attribute name="idp" type="xs:string" use="required"/>
       <xs:attribute name="naziv" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:key name="RadnikPK">
   <xs:selector xpath="Radnik"/>
   <xs:field xpath="@mbr"/>
  </xs:key>
  <xs:key name="ProjekatPK">
   <xs:selector xpath="Projekat"/>
   <xs:field xpath="@idp"/>
  </xs:key>
 </xs:element>
</xs:schema>
```

Листинг 85 XML Schema Root - keyref

XML документ који је валидан у односу на претходно дату шему је приказан у Листинг 86. У овом примеру један радник и пројекат имају исте шифре, што је сада дозвољено, јер су кључеви специфицирани преко xs:key елемента. Такође, вредност овако задатог кључа могу почињати цифром.

```
<?xml version="1.0" encoding="UTF-8"?>
<Root                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="KljucKEY.xsd">
 <Radnik mbr="R1" prezime="Petrovic" ime="Petar"/>
 <Radnik prezime="Markovic" mbr="R2" ime="Marko"/>
 <Radnik prezime="Ilic" mbr="R3" ime="Ilija"/>
 <Radnik prezime="Savic" mbr="4" ime="Sava"/>
 <Projekat idp="R1" naziv="Projekat 1"/>
</Root>
```

Листинг 86 Валидан XML документ

У Листинг 87, коренски елемент Studenti може да садржи више елемената Student. Сваки студент јединствено се идентификује у оквиру коренског елемента помоћу два обележја броја индекса и године уписа. Овакав кључ, који се састоји од два обележја, могуће је дефинисати једино помоћу xs:key елемента.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Studenti">
  <xs:annotation>
   <xs:documentation>Comment describing your root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Student" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="brojIndeksa" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:integer">
         <xs:totalDigits value="3"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="godinaUpisa" type="xs:gYear" use="required"/>
      <xs:attribute name="ime" type="xs:string" use="required"/>
      <xs:attribute name="prezime" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:key name="StudentPK">
   <xs:selector xpath="Student"/>
   <xs:field xpath="@brojIndeksa"/>
   <xs:field xpath="@godinaUpisa"/>
  </xs:key>
 </xs:element>
</xs:schema>
```

Листинг 87 XML Schema Studenti - key

XML документ који је валидан у односу на претходну шему, дат је у Листинг 88. Вредности делова кључа се могу понављати, али цео кључ има јединствене вредности.

```
<?xml version="1.0" encoding="UTF-8"?>
<Studenti                      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Kljuc2Obelezja.xsd">
 <Student brojIndeksa="123" godinaUpisa="2012" prezime="Antic" ime="Ana"/>
 <Student brojIndeksa="123" godinaUpisa="2013" prezime="Maric"  ime="Marko"/>
 <Student brojIndeksa="124" godinaUpisa="2013" prezime="Matic" ime="Ana"/>
</Studenti>
```

Листинг 88 Валидан XML документ

Без обзира на то на који начин је дефинисан кључ, постојећи XML СУБП дозвољавају измену вредности кључа. Да би то било спречено, предложено је проширење XML Schema-е, дато у

одељку 3.1.5, на основу којег генератор кода, описан у поглављу 4, генерише XQuery функције и тригере, који спречавају измену вредности кључа.

Пример 6

У Листинг 89 дата је XML Schema са описом елемента Radnik, а у Листинг 90 дат је XML документ који је валидан у односу на ову шему. Шема је проширена елементом xc:constraint, чији атрибут type има вредност keyCon. Елемент xc:constraint садржи елемент xc:condition. Атрибути овог елемента су from и pk, који, редом, садрже XPath израз којим се селектује елемент Radnik, и вредност атрибута који је кључ.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Radnici">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Radnik" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="jmbg" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:length value="13"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="ime" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:maxLength value="30"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="prezime" use="required">
       <xs:simpleType>
        <xs:restriction base="xs:string">
         <xs:maxLength value="30"/>
        </xs:restriction>
       </xs:simpleType>
      </xs:attribute>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:key name="Radnik_PK">
   <xs:selector xpath="Radnik"/>
   <xs:field xpath="@jmbg"/>
  </xs:key>
  <xc:constraint type="xc:keyCon">
   <xc:condition from="/Radnici/Radnik" pk="jmbg"/>
  </xc:constraint>
 </xs:element>
</xs:schema>
```

Листинг 89 XML Schema елемента Radnici са проширењем за ограничење кључа

Кључ елемента Radnik је атрибут mbr. Ограничење кључа се дефинише као:

KeyConRadnik(Radnik, Key(Radnik, mbr), (mbr, nebitna uloga, {update, NoAction})).

Критична операција је измена вредности атрибута који је кључ, и ту акцију треба забранити.

```
<Radnici>
    <Radnik jmbg="1509010800041" ime="Marko" prezime="Markovic"/>
</Radnici>
```

Листинг 90 XML документ Radnici

5.2.2.1 Контрола ограничења кључа помоћу XQuery функција

Модификација вредности кључа је могућа у постојећим XML СУБП. Ако је потребно забранити модификацију, то се мора урадити ван XML Schema-е, помоћу процедуралних механизама, као што је приказано у Листинг 91.

```
declare function local:canUpdateRadnik($OLD as element(Radnik),
                                        $NEW as element(Radnik))
    as xs:boolean {
        let $r := if ($OLD/@jmbg = $NEW/@jmbg) then true() else false()
        return $r
};
declare function local:doUpdateRadnik($OLD as element(Radnik),
                                      $NEW as element(Radnik))
    as xs:boolean{
        let $doUpdate :=
                    update replace doc('Radnik.xml')/Radnici/
                    Radnik[@jmbg=$OLD/@jmbg] with $NEW
    return true()
};

declare function local:updateRadnik($OLD as element(Radnik),
                                    $NEW as element(Radnik))
    as xs:boolean{
        let $canUpdate := if (local:canUpdateRadnik($OLD, $NEW))
            then local:doUpdateRadnik($OLD, $NEW)
            else false()
    return $canUpdate
};
```

Листинг 91 XQuery функција за забрану модификације кључа

5.2.2.2 Контрола ограничења кључа помоћу тригера

Ако су вредности атрибута jmbg у постојећем и новом елементу једнаке, могућа је замена старог елемента Radnik, новим елементом. Овај тригер је дат у Листинг 92.

```
CREATE TRIGGER "OgranicenjeKljucaRadnikBeforeUpdate"
BEFORE REPLACE
ON collection('OgranicenjeKljuca')/Radnici/Radnik
FOR EACH NODE
DO {
 if ($OLD/@jmbg = $NEW/@jmbg )
 then
  ($NEW)
 else
  error(xs:QName("OgranicenjeKljucaRadnikBeforeUpdate"),"Primarni kljuc se ne moze
menjati");
}
```

Листинг 92 Тригер за забрану модификације кључа

5.2.3 Ограничење референцијалног интегритета

У овом одељку приказано је ограничење референцијалног интегритета. Прво је описано ограничење референцијалног интегритета у релационом моделу података, а затим у XML моделу података. XML СУБП не имплементирају у потпуности ограничење референцијалног интегритета, тако да га је могуће нарушити операцијама ажурирања. Због тога је предложено

проширење XML Schema-е дато у одељку 3.1.7, на основу којег генератор описан у поглављу 4 генерише XQuery функције и тригере, који у потпуности реализују проверу овог ограничења, приликом извршавања операција ажурирања.

5.2.3.1 Ограничење референцијалног интегритета у релационом моделу података

Референцијални интегритет [Mog00] је основно и најчешће коришћено међурелационо ограничење које повезује податке у релацијама $r(N_i)$ и $r(N_j)$. Ово ограничење има задатак да одржава податке у појавама над шемама релација N_i и N_j у конзистентном стању. Референцијални интегритети између шема релација N_i и N_j, који су последица тривијалних зависности садржавања у шеми универзалне релације, називају се и ограничењем страног кључа. Међурелациона зависност садржавања $Ni[X] \subseteq Nj[Y]$ је тривијална, ако важи $X=Y$.

Ограничење страног кључа је основно, ако је између шема релација N_i и N_j задовољен услов

$$X \subseteq R_i \wedge X=Y \wedge Y=K_p(R_j).$$

Нека скуп обележја $Y=\{B_1, ..., B_n\}$ представља кључ шеме релације N_j. Ниједно од обележја B из скупа Y не може имати нула вредности у торкама релације $r(N_j)$. Ако путем ограничења $Null(N_i, A)$ није другачије наведено, обележја из скупа $X=\{A_1,..., A_n\}$ могу имати нула вредности у торкама релација $r(N_i)$.

У стандардима за релациони језик података SQL, посвећена је значајна пажња ограничењу референцијалног интегритета и механизмима за његову проверу у бази података.

5.2.3.2 Ограничење референцијалног интегритета у XML моделу података

Ограничење референцијалног интегритета се у XML моделу података може дефинисати помоћу keyref механизма у XML Schema документу. На овај начин се повезују скупови атрибута који у једном елементу представљају примарни кључ, а у другом елементу страни кључ.

Посматрајмо елементе RadnoMesto, који је приказан у Листинг 93, и Radnik, који је приказан у Листинг 94. Атрибути елемента RadnoMesto су oznakaRM и nazivRM, док су атрибути елемента Radnik mbr, ime, prezime и oznakaRM. Атрибут елемента Radnik је и oznakaRM, јер тај атрибут треба да представља страни кључ. Неопходно је да елемент Radnik садржи атрибут oznakaRM, јер без њега не би било могуће дефинисати ограничење референцијалног интегритета помоћу keyref механизма, а не би се могла дефинисати ни остала ограничења описана у поглављу 3.

Такође, ако је потребно да неки елемент има више страних кључева из различитих елемената, наводе се сви атрибути који представљају стране кључеве, а у елементима keyref наводе се референце ка елементима у којима су ти атрибути примарни кључеви.

```
<xs:element name="RadnoMesto" maxOccurs="unbounded">
  <xs:complexType>
    <xs:attribute name="oznakaRM" type="xs:string" use="required"/>
    <xs:attribute name="nazivRM" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

Листинг 93 Елемент RadnoMesto

```
<xs:element name="Radnik" maxOccurs="unbounded">
  <xs:complexType>
    <xs:attribute name="mbr" type="xs:string" use="required"/>
    <xs:attribute name="oznakaRM" type="xs:string" use="required"/>
    <xs:attribute name="ime" type="xs:string" use="required"/>
    <xs:attribute name="prezime" type="xs:string" use="required"/>
```

```
  </xs:complexType>
</xs:element>
```

Листинг 94 Елемент Radnik

У циљу навођења атрибута који представљају кључеве и стране кључеве, потребно је навести key и keyref елементе. Обележје oznakaRM представља кључ елемента RadnoMesto, као што је приказано у Листинг 95, док обележје mbr представља кључ елемента Radnik, као што је приказано у Листинг 96.

```
<xs:key name="RadnoMestoPK">
  <xs:selector xpath="RadnoMesto"/>
  <xs:field xpath="@oznakaRM"/>
</xs:key>
```

Листинг 95 Кључ елемента RadnoMesto

```
<xs:key name="RadnikPK">
  <xs:selector xpath="Radnik"/>
  <xs:field xpath="@mbr"/>
</xs:key>
```

Листинг 96 Кључ елемента Radnik

Атрибут oznakaRM је страни кључ у елементу Radnik, а односи се на примарни кључ елемента RadnoMesto, то јест на атрибут oznakaRM у елементу RadnoMesto, а то је у шеми приказано на начин као у Листинг 97.

```
<xs:keyref name="RadnikFK" refer="RadnoMestoPK">
  <xs:selector xpath="Radnik"/>
  <xs:field xpath="@oznakaRM"/>
</xs:keyref>
```

Листинг 97 Страни кључ у елементу Radnik

Могуће је за атрибут који представља страни кључ означити да је optional, што значи да се не мора за сваки подређени елемент навести вредност страног кључа из надређеног елемента.

Пример 7

На следећем примеру биће описани неки од проблема који се јављају при коришћењу ограничења референцијалног интегритета. XML документ у Листинг 98 садржи податке о радницима, пројектима, као и о ангажовању радника на пројектима. Како однос између радника и пројекта одговара повезнику са максималним кардиналитетима „више-више“, ангажовање на пројекту садржи примарне кључеве ова два елемента.

```
<?xml version="1.0" encoding="UTF-8"?>
<Posao                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Posao.xsd">
 <Radnik mbr="R1" ime="Petar" prezime="Petric"/>
 <Radnik mbr="R2" ime="Marko" prezime="Maric"/>
 <Radnik mbr="R3" ime="Ana" prezime="Antic"/>
 <Radnik mbr="R4" ime="Nina" prezime="Ninic"/>
 <Projekat spr="P1" naziv="Plate"/>
 <Projekat spr="P2" naziv="PDV"/>
 <Angazovanje mbr="R1" spr="P1" brc="10"/>
 <Angazovanje mbr="R2" spr="P1" brc="20"/>
</Posao>
```

Листинг 98 XML документ Posao

XML Schema документ који описује овај XML документ дат је Листинг 99.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
```

```
 <xs:element name="Posao">
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Radnik" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="mbr" type="xs:string" use="required"/>
      <xs:attribute name="ime" type="xs:string" use="required"/>
      <xs:attribute name="prezime" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Projekat" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="spr" type="xs:string" use="required"/>
      <xs:attribute name="naziv" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Angazovanje" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="mbr" type="xs:string" use="required"/>
      <xs:attribute name="spr" type="xs:string" use="required"/>
      <xs:attribute name="brc" type="xs:int" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:key name="Radnik_PK">
   <xs:selector xpath="Radnik"/>
   <xs:field xpath="@mbr"/>
  </xs:key>
  <xs:key name="Projekat_PK">
   <xs:selector xpath="Projekat"/>
   <xs:field xpath="@spr"/>
  </xs:key>
  <xs:key name="Angazovanje_PK">
   <xs:selector xpath="Angazovanje"/>
   <xs:field xpath="@mbr"/>
   <xs:field xpath="@spr"/>
  </xs:key>
  <xs:keyref name="Angazovanje_FK1" refer="Radnik_PK">
   <xs:selector xpath="Angazovanje"/>
   <xs:field xpath="@mbr"/>
  </xs:keyref>
  <xs:keyref name="Angazovanje_FK2" refer="Projekat_PK">
   <xs:selector xpath="Angazovanje"/>
   <xs:field xpath="@spr"/>
  </xs:keyref>
  <xc:constraint type="xc:refInt">
   <xc:condition from="/Posao/Radnik" fromName="Radnik"
            to="/Posao/Angazovanje" toName="Angazovanje"
            keyref="mbr" onupdate="xc:cascade" ondelete="xc:cascade"/>
   <xc:condition from="/Posao/Projekat" fromName="Projekat"
            to="/Posao/Angazovanje" toName="Angazovanje"
            keyref="spr" onupdate="xc:restrict" ondelete="xc:restrict"/>
  </xc:constraint>
 </xs:element>
</xs:schema>
```

Листинг 99 XML Schema документ који описује XML документ Posao

Органичење референцијалног интегритета између елемената Radnik и Angazovanje дато је на следећи начин:

RefIntAngazovanjeRadnik(RefIntCon,
Angazovanje@mbr⊆Radnik@mbr,
((Angazovanje, referencing, {(insert, noAction), (update, noAction)}),
(Radnik, referenced, {(delete, noAction, Cascade), (update, noAction, Cascade)})))

Кључ елемента Radnik је обележје MBR. Кључ елемента Projekat је обележје SPR, а кључ елемента Angazovanje је унија обележја MBR и SPR. Обележје MBR у елементу Angazovanje је страни кључ и односи се на примарни кључ елемента Radnik, а обележје SPR у елементу Angazovanje је такође страни кључ и односи се на примарни кључ елемента Projekat.

Елемент keyref у XML Schema документу већ повезује два елемента преко њихових примарних и страних кључева. Код атрибута MBR и SPR у елементу Angazovanje, XML атрибут use има вредност required, јер су та два атрибута делови примарног кључа елемента Angazovanje и не смеју имати нула вредности.

У одељцима 5.2.3.3 и 5.2.3.4 реализована је контрола ограничења референцијалног интегритета помоћу XQuery функција и тригера. Када XML СУБП не подржава тригере, једини могући начин за имплементацију ограничења је коришћењем XQuery функција.

5.2.3.3 Контрола ограничења референцијалног интегритета помоћу XQuery функција

У овом одељку биће приказан XQuery код који спречава нарушавање ограничења референцијалног интегритета. Овај код подржава две операције ажурирања, модификацију и брисање надређеног елемента у XML документу. У зависности од одабране реализације контроле ограничења, операција ће бити забрањена (NoAction) или каскадно изведена (Cascade).

XML документ над којим се врше провере ограничења референцијалног интегритета дат је у Листинг 98.

5.2.3.3.1 Модификација елемента Radnik

Модификација елемента Radnik може се реализовати на два начина: са спречавањем измене (NoAction), и са каскадном изменом (Cascade). Ове акције се морају спровести програмски, јер XML базе података за сада не подржавају забрану или каскадну акцију.

Ако вредност атрибута MBR елемента Radnik не постоји у неком од подређених елемената Angazovanje, онда се модификација може извршити, јер се тиме не нарушава ограничење референцијалног интегритета. Функцијом canUpdateRadnik се још проверава да ли постоји такав елемент Radnik, код ког је вредност атрибута mbr једнака новој вредности атрибута mbr, то јест, вредности атрибута mbr елемента радник који се модификује. Ако је један од ових услова задовољен, позивом функције doUpdateRadnik врши се замена целог елемента Radnik са траженом вредношћу матичног броја, новим елементом Radnik, у којем могу бити промењене вредности и осталих атрибута ime и prezime. Ако постоји елемент Angazovanje који се референцира на радника ког треба модификовати, таква промена се неће извршити. Модификација елемента Radnik приказана је у Листинг 100.

```
declare function local:canUpdateRadnik($OLD as element(Radnik), $NEW as
element(Radnik))
    as xs:boolean{
        let $exists :=
            not(exists(doc('Posao.xml')/Posao/Angazovanje[@mbr=$OLD/@mbr]) or
                exists(doc('Posao.xml')/Posao/Radnik[@mbr=$NEW/@mbr]))
   return ($exists)
};

declare function local:doUpdateRadnik($OLD as element(Radnik), $NEW as
element(Radnik))
    as xs:boolean{
```

```
        let $i := update replace doc('Posao.xml')/Posao/Radnik[@mbr=$OLD/@mbr] with
$NEW
        return true()
};

declare function local:updateRadnik($OLD as element(Radnik), $NEW as
element(Radnik))
    as xs:boolean{
        let $updMBR := if(local:canUpdateRadnik($OLD, $NEW))
            then local:doUpdateRadnik($OLD, $NEW)
            else false()
        (: ukljuci zabranu modifikacije :)
    return $updMBR
};
```

Листинг 100 XQuery функција за контролу референцијалног интегритета приликом модификације елемента Radnik са забраном модификације подређеног елемента (NoAction)

Ако се спроводи каскадна модификација, онда се након модификације елемента Radnik, спроводи и модификација елемента Angazovanje. Вредност матичног броја се мења и у сваком елементу Angazovanje где је постојала стара вредност, тако што се цео пронађени елемент Angazovanje мења новим елементом, чија је вредност обележја mbr нова вредност, а вредности обележја spr и brc се преузимају из старог елемента Angazovanje. Ова модификација је приказана у Листинг 101.

```
declare function local:canUpdateRadnik($OLD as element(Radnik), $NEW as
element(Radnik))
    as xs:boolean{
        let $exists := exists(doc('Posao.xml')/Posao/Radnik[@mbr = $OLD/@mbr]) and
not (exists(doc('Posao.xml')/Posao/Radnik[@mbr = $NEW/@mbr]))
   return ($exists)
};

declare function local:doUpdateRadnik($OLD as element(Radnik), $NEW as
element(Radnik))
    as xs:boolean{
        let $i := update replace doc('Posao.xml')//Radnik[@mbr=$OLD/@mbr] with $NEW
        return true()
};

declare function local:doUpdateAngazovanje($OLD as element(Radnik), $NEW as
element(Radnik))
    as xs:boolean {
        let $r :=
        for $i in doc('Posao.xml')/Posao/Angazovanje[@mbr = $OLD/@mbr]
   let $spr:=$i//@spr
   let $brc:=$i//@brc
            let $r := update replace
doc('Posao.xml')/Posao/Angazovanje[@mbr=$OLD/@mbr  and @spr=$spr  and @brc=$brc ]
with <Angazovanje mbr="{$NEW/@mbr}"  spr={$spr}  brc={$brc}  />
            return <res/>
        return true()
};

declare function local:updateRadnik($OLD as element(Radnik), $NEW as
element(Radnik))
    as xs:boolean{
        (: iskljuci zabranu modifikacije :)
        let $updMBR := if(local:canUpdateRadnik($OLD, $NEW))
            then local:doUpdateAngazovanje($OLD, $NEW) and
local:doUpdateRadnik($OLD, $NEW)
            else false()
        (: ukljuci zabranu modifikacije :)
    return $updMBR
};
```

Листинг 101 XQuery функција за контролу референцијалног интегритета приликом модификације елемента Radnik са каскадном променом подређеног елемента Angazovanje (Cascade)

5.2.3.3.2 Брисање елемента Radnik

Брисање елемента радник може се забранити, уколико постоји подређени елемент Angazovanje који се референцира на њега. Брисање радника се у том случају може дозволити само ако он нема подређених елемената Angazovanje. Ова провера се одвија у функцији canDeleteRadnik. XQuery функција за брисање радника дата је у Листинг 102.

```
declare function local:canDeleteRadnik($OLD as element(Radnik))
    as xs:boolean{
        let $r := not(exists(doc('Posao.xml')/Posao/Angazovanje[@mbr=$OLD/@mbr]) or
not (exists(doc('Posao.xml')/Posao/Radnik[@mbr=$OLD/@mbr])))
        return ($r)
};

declare function local:doDeleteRadnik($OLD as element(Radnik))
    as xs:boolean{
        let $i := update delete doc('Posao.xml')/Posao/Radnik[@mbr=$OLD/@mbr]
        return true()
};

declare function local:deleteRadnik($OLD as element(Radnik))
    as xs:boolean{
        let $delMBR := if(local:canDeleteRadnik($OLD))
            then local:doDeleteRadnik($OLD)
            else false()
    return $delMBR
};
```

Листинг 102 XQuery функција за брисање елемента Radnik са забраном брисања подређеног елемента

Ако је у питању каскадно брисање, треба избрисати и све елементе Angazovanje код којих је вредност обележја mbr једнака вредности обележја mbr елемента Radnik који се брише. Функција canDeleteRadnik проверава да ли је могуће брисање радника са датом вредношћу матичног броја, тако што проналази елемент Radnik са задатом вредношћу атрибута. Ако такав елемент постоји, из документа се прво бришу сви елементи Angazovanje чија је вредност обележја mbr једнака датом, а затим се брише и сам нађени елемент Radnik. XQuery функција је дата у Листинг 103.

```
declare function local:canDeleteRadnik($OLD as element(Radnik))
    as xs:boolean{
        let $exists := exists(doc('Posao.xml')/Posao/Radnik[@mbr = $OLD/@mbr])
        return ($exists)
};

declare function local:doDeleteRadnik($OLD as element(Radnik))
    as xs:boolean{
        let $i := update delete doc('Posao.xml')/Posao/Radnik[@mbr=$OLD/@mbr]
        return true()
};

declare function local:doDeleteAngazovanje($OLD as element(Radnik))
    as xs:boolean{
        let $i := update delete doc('Posao.xml')/Posao/Angazovanje[@mbr=$OLD/@mbr]
        return true()
};

declare function local:deleteRadnik($OLD as element(Radnik))
    as xs:boolean{
        (: iskljuci zabranu modifikacije :)
        let $delMBR := if(local:canDeleteRadnik($OLD))
            then local:doDeleteAngazovanjee($OLD) and local:doDeleteRadnik($OLD)
```

```
            else false()
    return $delMBR
};
```

Листинг 103 XQuery функција за брисање елемента Radnik са каскадним брисањем подређеног елемента

На сличан начин се може реализовати и модификација елемента Projekat. Ако се покуша брисање елемента Projekat, ова акција се може забранити уколико у било ком елементу Angazovanje постоји таква вредност обележја spr која одговара вредности обележја spr елемента Projekat који се брише.

5.2.3.4 Контрола ограничења референцијалног интегритета помоћу тригера

У овом одељку биће приказани тригери који спречавају нарушавање ограничења референцијалног интегритета. Тригери се активирају код две операције ажурирања: модификације и брисања надређеног елемента у XML документу. Како тригери у XML СУБП Sedna не могу да реализују каскадно брисање, овде су наведени само тригери који забрањују модификацију и брисање надређеног елемента.

5.2.3.4.1 Модификација елемента Radnik

Модификација елемента Radnik се забрањује ако је он референциран у елементу Angazovanje, или ако се вредност атрибута mbr радника кога треба модификовати, не нађе ни у једном елементу Angazovanje. У супротном, стари елемент Radnik мења се оним елементом Radnik. Тригер за модификацију радика дат је у Листинг 104.

```
CREATE TRIGGER "ReferencijalniIntegritetRadnikBeforeUpdate"
BEFORE REPLACE
ON collection('ReferencijalniIntegritet')/Posao/Radnik
FOR EACH NODE
DO {
 if (exists(fn:doc('Posao', 'ReferencijalniIntegritet')/
              Posao/Angazovanje[@mbr=$OLD/@mbr]) or
     exists(fn:doc('Posao', 'ReferencijalniIntegritet')/
              Posao/Radnik[@mbr=$NEW/@mbr]))
 then
  error(xs:QName("ReferencijalniIntegritetRadnikBeforeUpdate"),"Ne   moze   da   se
obrise zato sto se mbr vec koristi u Angazovanje")
 else
   ($NEW);
}
```

Листинг 104 Тригер за забрану модификације елемента Radnik

5.2.3.4.2 Брисање елемента Radnik

Слично као код модификације, брисање елемента Radnik могуће је само ако тај радник није референциран ни у једном елементу Angazovanje. У супротном, или ако такав радник ни не постоји, брисање се забрањује. Тригер који контролише брисање елемента Radnik, дат је у Листинг 105.

```
CREATE TRIGGER "ReferencijalniIntegritetRadnikBeforeDelete"
BEFORE DELETE
ON collection('ReferencijalniIntegritet')/Posao/Radnik
FOR EACH NODE
DO {
 if(exists(fn:doc('Posao', 'ReferencijalniIntegritet')/
             Posao/Angazovanje[@mbr=$OLD/@mbr]) or
 not (exists(fn:doc('Posao', 'ReferencijalniIntegritet')/
               Posao/Radnik[@mbr=$OLD/@mbr])))
 then
```

```
  error(xs:QName("ReferencijalniIntegritetRadnikBeforeDelete"),"Ne moze da se
promeni zato sto se mbr vec koristi u Angazovanje")
 else
   ($OLD);
}
```

Листинг 105 Тригер за забрану брисања елемента Radnik

5.2.4 Закључак

У одељку 5.2 дати су примери генерисаног кода који реализују три ограничења, која нису у потпуности подржана XML Schema-ом, нити су реализована у потпуности у постојећим XML СУБП. То су ограничење торке, ограничење кључа и ограничење референцијалног интегритета. На основу спецификације дате у одељцима 3.1.3, 3.1.5 и 3.1.7, генерисане су XQuery функције и тригери, који врше проверу ових ограничења приликом извршавања операција ажурирања. Програмери овиме добијају могућност да у практичном раду употребљавају код који је генерисан на овај начин, уместо да ручно имплементирају провере ограничења, што свакако побољшава ефективност њиховог рада.

5.3 Ограничења која нису дефинисана у XML моделу података

У овом одељку дат је предлог реализације ограничења која нису до сада дефинисана у XML моделу података, а у пракси пројектовања база података обрасци ограничења таквих типова често се појављују. То су следећа три ограничења:

- проширено ограничење торке,
- ограничење проширеног референцијалног интегритета и
- ограничење инверзног референцијалног интегритета.

У овом одељку примењени су исти принципи као у одељку 5.2. То значи да су, на основу спецификација, датих у одељцима 3.1.4, 3.1.9 и 3.1.8, генерисане XQuery функције и тригери који реализују проверу наведених ограничења. Примери реализације ових ограничења дати су у наставку.

5.3.1 Проширено ограничење торке

У овом одељку приказано је проширено ограничење торке. Прво је описано ограничење у релационом моделу података, а затим у XML моделу података. У постојећим XML СУБП не постоји имплементација овог ограничења, нити га XML Schema подржава. Проширење шеме описано је у одељку 3.1.4, а на основу њега генератор описан у поглављу 4 генерише XQuery функције и тригере, који у потпуности реализују проверу овог ограничења, приликом извршавања операција ажурирања.

5.3.1.1 Проширено ограничење торке у релационом моделу података

У релационом моделу података проширено ограничење торке је вишерелационо ограничење које дефинише логички услов који мора да буде задовољен између скупова обележја различитих шема релација. У следећем примеру приказан је део шеме базе података у којој је било неопходно дефинисати и проширено ограничење торке, како би била успостављена веза између обележја која се налазе у две шеме релације.

Пример 8

Слика 7 Шеме релација Građanin и Dokument

Дате су шеме релација:

Građanin({JMBG, IME, PREZIME, POL, DATR}, {JMBG}) и

Dokument({TIP, SBROJ, DATIZD, STAT, JMBG}, {SBROJ}),

приказане на Слика 7.

Важи ограничење референцијалног интегритета:

Dokument[JMBG] ⊆ Građanin[JMBG]

У тренутку уписа нове торке у релацију r(Građanin) не сме да постоји одговарајућа торка у релацији r(Dokument), која би имала вредност обележја JMBG исту као вредност обележја JMBG грађанина који се нови уноси.

Проширено ограничење торке, записно следећим правилом:

τ_{ex}(Građanin ⋈ Dokument) = Con(Građanin ⋈Dokument) – DATIZD ≥ DATR

обезбеђује да сви датуми издавања документа неког грађанина морају бити већи од датума рођења тог грађанина.

У релационом моделу података ово ограничење може да се реализује помоћу CHECK ограничења, али и помоћу тригера.

5.3.1.2 Проширено ограничење торке у XML моделу података

У XML моделу података проширено ограничење торке такође повезује атрибуте који се појављују у различитим елементима. Логичким условом задаје се веза између атрибута.

Пример 9

У овом примеру дат је XML документ који садржи податке о грађанима и њиховим личним документима. Сваки грађанин може да има више личних докумената, као што су лична карта, пасош, возачка дозвола. Документ припада тачно једном грађанину, па се у елементу Dokument налази обележје jmbg које се односи на грађанина ком тај документ припада.

```
<?xml version="1.0" encoding="UTF-8"?>
<sup                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Sup.xsd">
 <Gradjanin jmbg="1234" prezime="Peric" ime="Pera" pol="M" datr="1980-02-02"/>
 <Dokument tip="LK" sbroj="4321" datizd="2000-02-02" jmbg="1234"/>
 <Dokument tip="VD" sbroj="5432" datizd="2000-02-02" jmbg="1234"/>
</sup>
```

Листинг 106 XML документ sup.xml

У наставку је дат и XML Schema документ којим се описује претодни XML документ.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="sup">
  <xs:annotation>
     <xs:documentation>Comment describing your root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
     <xs:sequence>
        <xs:element name="Gradjanin" maxOccurs="unbounded">
           <xs:complexType>
              <xs:attribute name="jmbg" type="xs:string" use="required"/>
              <xs:attribute name="prezime" type="xs:string" use="required"/>
              <xs:attribute name="ime" type="xs:string" use="required"/>
              <xs:attribute name="pol" use="required">
                 <xs:simpleType>
                    <xs:restriction base="xs:string">
                     <xs:enumeration value="M"/>
                     <xs:enumeration value="Z"/>
                    </xs:restriction>
                 </xs:simpleType>
              </xs:attribute>
              <xs:attribute name="datr" type="xs:date" use="required"/>
           </xs:complexType>
        </xs:element>
        <xs:element name="Dokument" maxOccurs="unbounded">
           <xs:complexType>
              <xs:attribute name="tip" use="required">
                 <xs:simpleType>
                    <xs:restriction base="xs:string">
                     <xs:enumeration value="VD"/>
                     <xs:enumeration value="LK"/>
                     <xs:enumeration value="P"/>
                    </xs:restriction>
                 </xs:simpleType>
              </xs:attribute>
              <xs:attribute name="sbroj" type="xs:positiveInteger" use="required"/>
              <xs:attribute name="datizd" type="xs:date" use="required"/>
              <xs:attribute name="jmbg" use="required"/>
           </xs:complexType>
        </xs:element>
     </xs:sequence>
  </xs:complexType>
  <xs:key name="GradjaninPK">
     <xs:selector xpath="Gradjanin"/>
     <xs:field xpath="@jmbg"/>
  </xs:key>
  <xs:key name="DokumentPK">
     <xs:selector xpath="Dokument"/>
     <xs:field xpath="@sbroj"/>
  </xs:key>
  <xs:keyref name="DokumentFK" refer="GradjaninPK">
     <xs:selector xpath="Dokument"/>
     <xs:field xpath="@jmbg"/>
  </xs:keyref>
  <xc:constraint type="xc:exTupleCon">
     <xc:condition from="/sup/Gradjanin" to="/sup/Dokument" keyref="jmbg"
                   toPK="sbroj" additional="${from}/@datr ?lt; ${to}/@datizd"/>
  </xc:constraint>
 </xs:element>
</xs:schema>
```

Листинг 107 XML Schema sup.xsd

Назив ограничења у овом примеру је exTupleCon. У елементу xc:constraint појављује се елемент xc:condition у којем се наводи између која два елемента важи проширено ограничење торке, као и услов који треба да буде задовољен. На основу овог услова могу да се формирају XQuery упити, као и тригери, који контролишу проширено ограничење торке.

На основу спецификације дате у одељку 3.1.4, проширено ограничење торке у овом примеру дефинисано је као:

ExTupleConGradjaninDokument(ExTupleCon,
τ_{ex}(Gradjanin⋈Dokument)=Constraint(Gradjanin⋈Dokument): DATIZD≥DATR,
((Gradjanin, nebitna uloga, {(insert, /), (update (DATR), NoAction)}),
(Dokument, nebitna uloga, {(insert, NoAction), (update (DATIZD, JMBG), NoAction)}))

Услов који треба да буде задовољен је да је датум издавања документа већи или једнак од датума рођења грађанина ком тај документ припада. Критичне операције које могу нарушити проширено ограничење торке су унос новог документа, унос новог грађанина, као и модификација датума рођења, и модификација документа.

У наставку овог одељка, контрола проширеног ограничења торке дата је помоћу XQuery функција и тригера.

5.3.1.3 Контрола проширеног ограничења торке помоћу XQuery функција

У овом одељку биће приказан XQuery код који спречава нарушавање проширеног ограничења торке. XML документ над којим се врше провере овог ограничења дат је у Листинг 106. Генерисани код подржава следеће операције ажурирања: унос и модификацију подређеног елемента (Dokument), и унос и модификацију надређеног елемента (Gradjanin).

5.3.1.3.1 Унос новог документа

Приликом уноса новог елемента Dokument потребно је проверити да ли је вредност атрибута datizd већа од вредности атрибута datr у елементу Građanin, а где су вредности атрибута jmbg у та два елемента исте. Када се додаје нови елемент Dokument, прво се проверава ограничење референцијалног интегритета, то јест, да ли вредност обележја jmbg у новом елементу Dokument, постоји у неком елементу Građanin. Ако је датум издавања новог документа већи од датума рођења грађанина ком тај документ припада, што се проверава у XQuery функцији canInsertDokument, у XML документ додаје се нови елемент Dokument, што се ради у XQuery функцији doInsertDokument, која је приказана у Листинг 108. Нови документ се додаје испред свих постојећих елемената Dokument.

```
declare function local:canInsertDokument($NEW as element(Dokument))
    as xs:boolean {
        let $ret := exists(doc('sup.xml')/sup/Gradjanin[@jmbg = $NEW/@jmbg]) and
doc('sup.xml')/sup/Dokument[@jmbg = $NEW/@jmbg]/@datr < $NEW/@datizd)
        return $ret
};

declare function local:doInsertDokument($NEW as element(Dokument))
    as xs:boolean{
        let $i := update insert $NEW preceding doc('sup.xml')/sup/Dokument[1]
        return true()
};

declare function local:insertDokument($NEW as element(Dokument))
    as xs:boolean{
        let $can :=  local:canInsertDokument($NEW)
        let $res := if ($can)
                    then local:doInsertDokument($NEW)
                    else false()
        return $res
};
```

Листинг 108 XQuery функција за унос новог документа

5.3.1.3.2 Унос и модификација грађанина

Када се додаје нови елемент Građanin, за њега још увек не постоје придружени документи, па се ово додавање може обавити, уколико је вредност атрибута jmbg непостојећа.

Пре покушаја промене вредности атрибута datr у елементу Građanin, потребно је пронаћи све документе који припадају том грађанину, тако што се пронађу сви елементи Dokument, чија је вредност атрибута jmbg једнака вредности тог атрибута у елементу Građanin, чији се датум рођења мења. Уколико је у неком од таквих докумената датум издавања мањи од новог датума рођења, промена се забрањује. XQuery функција која проверава проширено ограничење торке приликом модификације елемента грађанин, дата је у Листинг 109.

```
declare function local:canUpdateGradjanin($OLD as element(Gradjanin), $NEW as
element(Gradjanin))
    as xs:boolean {
        let $ret := ($NEW/@jmbg=$OLD/@jmbg) and
                    count(doc('sup.xml')/sup/Dokument[@jmbg = $NEW/@jmbg and
                                                      not($NEW/@datr < @datizd)])
                         = 0
        return $ret
};

declare function local:doUpdateGradjanin($OLD as element(Gradjanin), $NEW as
element(Gradjanin))
    as xs:boolean{
        let $i := update replace doc('sup.xml')/sup/Gradjanin[@jmbg=$OLD/@jmbg]
                  with $NEW
        return true()
};

declare function local:updateGradjanin($OLD as element(Gradjanin), $NEW as
element(Gradjanin))
    as xs:boolean{
        let $can :=  local:canUpdateGradjanin($OLD, $NEW)
        let $res := if ($can)
                    then local:doUpdateGradjanin($OLD, $NEW)
                    else false()
        return $res
};
```

Листинг 109 XQuery функција за модификацију грађанина

5.3.1.3.3 Модификација документа

Приликом покушаја модификације документа, а то се пре свега односи на промену датума издавања, потребно је пронаћи грађанина ком тај документ припада и проверити да ли је датум рођења мањи од новог датума издавања. Функција која врши модификацију елемента Dokument, дата је у Листинг 110.

```
declare function local:canUpdateDokument($NEW as element(Dokument))
    as xs:boolean {
        let $ret := exists(doc('sup.xml')/sup/Gradjanin[@jmbg = $NEW/@jmbg]) and
doc('sup.xml')/sup/Dokument[@jmbg = $NEW/@jmbg]/@datr < $NEW/@datizd)
        return $ret
};

declare function local:doUpdateDokument($NEW as element(Dokument))
    as xs:boolean{
        let $i := update replace doc('sup.xml')/sup/Dokument[@sbroj=$NEW/@sbroj]
with $NEW
        return true()
};

declare function local:updateDokument($NEW as element(Dokument))
```

```
    as xs:boolean{
        let $can :=  local:canUpdateDokument($NEW)
        let $res := if ($can)
                    then local:doUpdateDokument($NEW)
                    else false()
        return $res
};
```

Листинг 110 XQuery функција за модификацију документа

5.3.1.4 Контрола проширеног ограничења торке помоћу тригера

У овом одељку биће приказани тригери који спречавају нарушавање проширеног ограничења торке. XML документ над којим се врше провере овог ограничења дат је у Листинг 106. Генерисани код подржава следеће операције ажурирања: унос и модификацију подређеног елемента (Dokument), и унос и модификацију надређеног елемента (Gradjanin).

5.3.1.4.1 Унос новог документа

Тригер се покреће пре додавања новог елемента Dokument. Услов који се проверава је исти као и код XQuery функције. Грануларност овог тригера је на нивоу чвора, па се за сваки документ који задовољава услов извршава тело DO петље. Овај тригер је дат у Листинг 111.

```
CREATE TRIGGER "ProsOgrTorkeDokumentBeforeInsert"
BEFORE INSERT
ON collection('ProsOgrTorke')/sup/Dokument
FOR EACH NODE
DO {
 if (exists(fn:doc('sup', 'ProsOgrTorke')/sup/Gradjanin[@jmbg = $NEW/@jmbg]) and
     fn:doc('sup', 'ProsOgrTorke')/sup/
     Dokument[@jmbg = $NEW/@jmbg]/@datr < $NEW/@datizd))
 then
  ($NEW)
 else
  error(xs:QName("ProsOgrTorkeDokumentBeforeInsert"),"Dokument ne moze da se doda,
posto ne zadovoljava dodatni uslov");
}
```

Листинг 111 Тригер који се окида пре уноса новог документа

У постојећи XML документ из Листинг 106, могуће је додати нови елемент Dokument, као у Листинг 112, јер је датум издавања тог документа већи од датума рођења грађанина ком тај документ припада.

```
UPDATE INSERT
<Dokument tip="P" sbroj="1223" datizd="2000-02-02" jmbg="1234"/>
INTO fn:doc("sup", "ProsOgrTorke")/sup
```

Листинг 112 Успешно додавање елемента Dokument

Међутим, није могуће додати елемент као у Листинг 113, јер је датум издавања мањи од датума рођења грађанина км документ припада.

```
UPDATE INSERT
<Dokument tip="P" sbroj="1223" datizd="1950-02-02" jmbg="1234"/>
INTO fn:doc("sup", "ProsOgrTorke")/sup>
```

Листинг 113 Неуспешно додавање елемента Dokument

Такође, није могуће додавање елемента Dokument, као у Листинг 114, јер вредност атрибута jmbg не постоји ни у једном елементу Radnik.

```
UPDATE INSERT
<Dokument tip="P" sbroj="1223" datizd="2000-02-02" jmbg="1233"/>
```

```
INTO fn:doc("sup", "ProsOgrTorke")/sup
```

Листинг 114 Неуспешно додавање елемента Dokument

5.3.1.4.2 Унос и модификација грађанина

Тригер се извршава пре покушаја модификације елемента Gradjanin. Модификација је могућа, уколико је задовољен услов да је датум рођења грађанина, а који се мења, мањи од датума издавања свих докумената који припадају том грађанину. Овај тригер је дат у Листинг 115.

```
CREATE TRIGGER "ProsOgrTorkeGradjanin"
BEFORE REPLACE
ON collection('ProsOgrTorke')/sup/Gradjanin
FOR EACH NODE
DO {
 if (($NEW/@jmbg=$OLD/@jmbg) and
   count(fn:doc('sup', 'ProsOgrTorke')/sup/
   Dokument[@jmbg = $NEW/@jmbg and not($NEW/@datr < @datizd)]) = 0)
 then
  ($NEW)
 else
  error(xs:QName("ProsOgrTorkeGradjanin"),"Gradjanin ne moze da se izmeni, posto ne
zadovoljava dodatni uslov");
}
```

Листинг 115 Тригер који се окида пре модификације грађанина

Модификација грађанина из XML документа у Листинг 106 није могућа, јер је датум рођења већи од датума издавања докумената који припадају том грађанину. Ова модификација је дата у Листинг 116.

```
UPDATE
REPLACE $r in fn:doc("sup", "ProsOgrTorke")/sup/Gradjanin[@jmbg="1234"]
WITH
<Gradjanin jmbg="1234" prezime="Peric" ime="Pera" pol="M" datr="2015-02-02"/>
```

Листинг 116 Покушај модификације елемента Gradjanina

5.3.1.4.3 Модификација документа

Датум издавања документа је могуће променити, ако је он већи од датума рођења грађанина ком тај документ припада.

```
CREATE TRIGGER "ProsOgrTorkeDokumentBeforeUpdate"
BEFORE REPLACE
ON collection('ProsOgrTorke')/sup/Dokument
FOR EACH NODE
DO {
 if (exists(fn:doc('sup', 'ProsOgrTorke')/sup/Gradjanin[@jmbg = $NEW/@jmbg])
      and fn:doc('sup', 'ProsOgrTorke')/sup/
      Dokument[@jmbg = $NEW/@jmbg]/@datr < $NEW/@datizd))
 then
  ($NEW)
 else
  error(xs:QName("ProsOgrTorkeDokumentBeforeUpdate"),"Dokument ne moze da se
izmeni, posto ne zadovoljava dodatni uslov");
}
```

Листинг 117 Тригер који се окида пре модификације документа

Покушај модификације документа приказан је у Листинг 118, али није успешан, јер је нови датум издавања документа мањи од датума рођења грађанина ком тај документ припада.

```
(: ne moze, posto je datum izdavanja pre datuma rodjenja :)
UPDATE
REPLACE $r in fn:doc("sup", "ProsOgrTorke")/sup/Dokument[@sbroj="1223"]
```

```
WITH
<Dokument tip="P" sbroj="1223" datizd="1950-02-02" jmbg="1234"/>
```

Листинг 118 Покушај модификације елемента Dokument

5.3.2 Ограничење проширеног референцијалног интегритета

У овом одељку приказано је ограничење проширеног референцијалног интегритета. Прво је описано ограничење у релационом моделу података, а затим у XML моделу података. У постојећим XML СУБП не постоји имплементација овог ограничења, нити га XML Schema подржава. Проширење шеме описано је у одељку 3.1.9, а на основу њега генератор описан у поглављу 4 генерише XQuery функције и тригере, који у потпуности реализују проверу овог ограничења, приликом извршавања операција ажурирања.

5.3.2.1 Ограничење проширеног референцијалног интегритета у релационом моделу података

Постоје скупови шема релација код којих се не могу директно дефинисати референцијални интегритети за све парове шема релација (N_i,N_j) из графа затварања. То су парови за које важи $X=Y=K_p(R_j)$ и $K_p(R_j)\not\subseteq R_i$. У Пример 10 описана је наведена ситуација. Пример је детаљно описан у [MLG, Mog00, Mog00a].

Пример 10

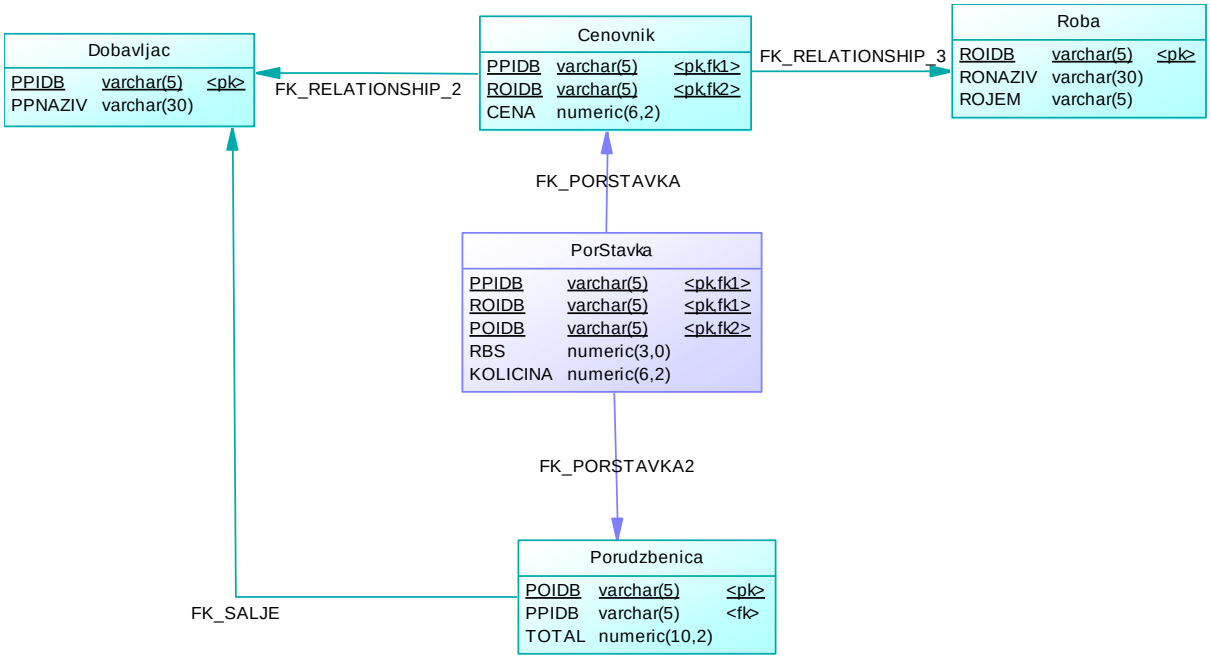
Шема базе података Наручивање представља модел односа између поруџбина, добављача, робе и ценовника робе добављача. Обележја имају следећа значења:

- RBS – редни број ставке поруџбине,
- POIDB – идентификациони број поруџбине,
- TOTAL – укупна цена поручене робе,
- ROIDB – идентификациони број робе,
- KOLICINA – количина робе на ставци,
- PPIDB – идентификациони број довабљача,
- PPNAZIV – назив добављача,
- CENA – цена робе код добављача,
- RINAZIV – назив робе,
- ROJEM – јединица мере.

Дате су шеме релација:

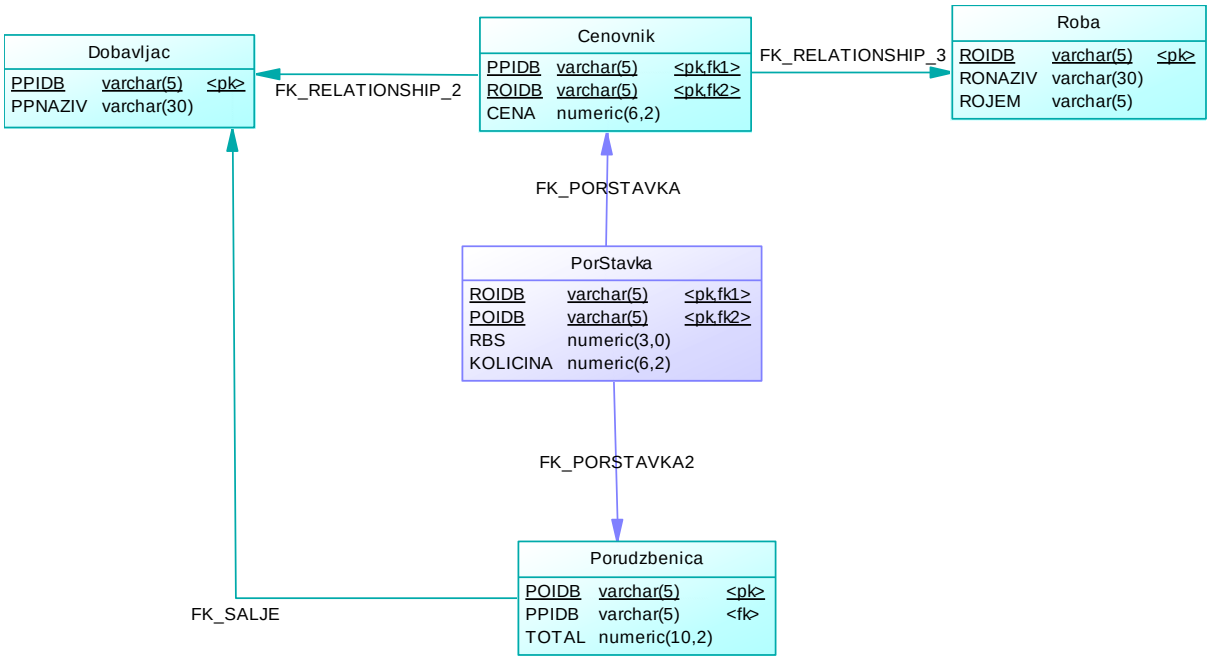
- Dobavljač({PPIDB, PPNAZIV}, {PPIDB})
- Roba({ROIDB, RONAZIV,ROJEM}, {ROIDB})
- Cenovnik({PPIDB, ROIDB, CENA}, {PPIDB+ROIDB})
- Porudžbenica({POIDB, PPIDB, TOTAL}, {POIDB})
- PorStavka({RBS, POIDB, ROIDB, KOLICINA}, {RBS+POIDB, POIDB+ROIDB})

На Слика 8 дат је физички модел ове шеме базе података.



Слика 8 Шема базе података Наручивање

У шеми релације PorStavka не постоји обележје PPIDB, јер је нормализацијом утврђено да је сувишно. Како важе функционална зависност POIDB → PPIDB у шеми релације Porudžbenica, информација о добављачу коме се шаље поруџбеница постоји у шеми релације Porudžbenica. Тај добављач је исти добављач који и добавља робу за коју се прави поруџбеница. На Слика 9 приказан је нормализован модел шеме базе података Наручивање.



Слика 9 Нормализована шема базе података Наручивање

За пар шема релација (PorStavka, Porudžbenica) важи X=Y=(POIDB). Услов $X \subseteq R_i$ је задовољен, па се између шема релација PorStavka и Porudžbenica може дефинисати ограничење основног референцијалног интегритета.

$$PorStavka[POIDB] \subseteq Porudžbenica[POIDB]$$

У тренутку уписа нове торке у релацију Porudžbenica не сме да постоји одговарајућа торка у релацији PorStavka, јер би било нарушено ограничење референцијалног интегритета.

За пар шема релација (PorStavka, Cenovnik) важи X=Y=(PPIDB,ROIDB) и $X \cap R_i$=(ROIDB). Услов $X \subseteq R_i$ није задовољен, јер {PPIDB,ROIDB}⊈{RBS,POIDB,ROIDB,KOLICINA}=R(PorStavka). Међутим, између шема релација PorStavka и Cenovnik требало би да се дефинише референцијални интегритет, како би се обезбедило да ставке садрже робу само оног добављача којем се поруџбина шаље.

Између шема релација PorStavka и Cenovnik дефинише се проширено ограничење страног кључа, или проширени референцијални интегритет:

$$\bowtie(\text{PorStavka, Porudžbenica})[(\text{PPIDB, ROIDB})] \subseteq \text{Cenovnik}[(\text{PPIDB, ROIDB})]$$

Семантика овог ограничења је да ставке поруџбине могу садржати само робу оног добављача, којем се поруџбина упућује. Ако се уместо тог ограничења дефинишу зависности садржавања PorStavka[ROIDB] ⊆ Cenovnik[ROIDB] и Porudžbenica [PPIDB] ⊆ Cenovnik[PPIDB] свака ставка једне поруџбине може садржати робу неког другог добављача, под условом да је његова роба евидентирана у ценовнику.

Ограничење проширеног референцијалног интегритета је вишерелационо међурелационо ограничење. У релационом систему за управљање базом података може се реализовати помоћу SQL наредби CREATE TRIGGER, CREATE/ALTER TABLE, CONSTRAINT CHECK, CREATE ASSERTION, као и помоћу тригера.

5.3.2.2 Ограничење проширеног референцијалног интегритета у XML моделу података

Ово ограничење се може дефинисати и у XML моделу података. Провера важења ограничења проширеног референцијалног интегритета спроводи се приликом уноса или брисања неког елемента, као и приликом промене вредности одређених атрибута.

Пример 11

Посматрајмо XML Schema-у и XML документ који је валидан у односу на њу, који одговара скупу шема релација из Пример 10. У Листинг 119 XML документ Narucivanjedat је XML документ у којем се налазе подаци о произвођачима који добављају различите врсте робе и сваки произвођач за своју робу има одређене цене. Када се роба поручује од неког произвођача, њему се шаље поруџбеница.

```
<Narucivanje                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Narucivanje.xsd">
    <Porudzbenica poidb="P1" ppidb="Dobavljac1"/>
    <Porudzbenica poidb="P2" ppidb="Dobavljac2"/>
    <Porudzbenica poidb="P3" ppidb="Dobavljac2"/>
    <PorStavka poidb="P3" roidb="R3" kolicina="1"/>
    <PorStavka poidb="P1" roidb="R1" kolicina="10"/>
    <PorStavka poidb="P1" roidb="R4" kolicina="100"/>
    <PorStavka poidb="P2" roidb="R1" kolicina="20"/>
    <Cenovnik roidb="R1" ppidb="Dobavljac1" cena="100"/>
    <Cenovnik roidb="R1" ppidb="Dobavljac2" cena="300"/>
    <Cenovnik roidb="R2" ppidb="Dobavljac2" cena="200"/>
    <Cenovnik roidb="R3" ppidb="Dobavljac2" cena="300"/>
    <Cenovnik roidb="R4" ppidb="Dobavljac1" cena="300"/>
    <Cenovnik roidb="R4" ppidb="Dobavljac2" cena="300"/>
    <Cenovnik roidb="R4" ppidb="Dobavljac3" cena="300"/>
    <Roba roidb="R1" ronaziv="papir"/>
    <Roba roidb="R2" ronaziv="spajalice"/>
```

```
    <Roba roidb="R3" ronaziv="olovka"/>
    <Roba roidb="R4" ronaziv="gumica"/>
    <Dobavljac ppidb="Dobavljac1" ppnaziv="Dobavljac1"/>
    <Dobavljac ppidb="Dobavljac2" ppnaziv="Dobavljac2"/>
    <Dobavljac ppidb="Dobavljac3" ppnaziv="Dobavljac3"/>
</Narucivanje>
```

Листинг 119 XML документ Narucivanje

XML Schema документ који описује претходни XML документ дат је у Листинг 120. Сви елементи су наведени испод коренског елемента Narucivanje. Примарни кључеви сваког елемента дефинисани су у xs:key елементу у XML Schema документу. Страни кључеви постоје као атрибути у елементима, а у XML Schema документу су дефинисани помоћу елемента xs:keyref.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema                                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <xs:element name="Narucivanje">
  <xs:annotation>
   <xs:documentation>Comment describing your root element</xs:documentation>
  </xs:annotation>
  <xs:complexType>
   <xs:sequence>
    <xs:element name="Porudzbenica" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="poidb" type="xs:string" use="required"/>
      <xs:attribute name="ppidb" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="PorStavka" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="poidb" type="xs:string" use="required"/>
      <xs:attribute name="roidb" type="xs:string" use="required"/>
      <xs:attribute name="kolicina" type="xs:int" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Cenovnik" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="ppidb" type="xs:string" use="required"/>
      <xs:attribute name="roidb" type="xs:string" use="required"/>
      <xs:attribute name="cena" type="xs:double" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Roba" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="roidb" type="xs:string" use="required"/>
      <xs:attribute name="ronaziv" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
    <xs:element name="Dobavljac" maxOccurs="unbounded">
     <xs:complexType>
      <xs:attribute name="ppidb" type="xs:string" use="required"/>
      <xs:attribute name="ppnaziv" type="xs:string" use="required"/>
     </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
  <xs:key name="Roba_PK">
   <xs:selector xpath="Roba"/>
   <xs:field xpath="@roidb"/>
  </xs:key>
  <xs:key name="Cenovnik_PK">
   <xs:selector xpath="Cenovnik"/>
   <xs:field xpath="@ppidb"/>
   <xs:field xpath="@roidb"/>
  </xs:key>
```

```
  <xs:key name="Porudzbenica_PK">
   <xs:selector xpath="Porudzbenica"/>
   <xs:field xpath="@poidb"/>
  </xs:key>
  <xs:key name="PorStavka">
   <xs:selector xpath="PorStavka_PK"/>
   <xs:field xpath="@poidb"/>
   <xs:field xpath="@roidb"/>
  </xs:key>
  <xs:key name="Dobavljac_PK">
   <xs:selector xpath="Dobavljac"/>
   <xs:field xpath="@ppidb"/>
  </xs:key>
  <xs:keyref name="PorStavka_FK1" refer="Porudzbenica_PK">
   <xs:selector xpath="PorStavka"/>
   <xs:field xpath="@poidb"/>
  </xs:keyref>
  <xs:keyref name="PorStavka_FK2" refer="Roba_PK">
   <xs:selector xpath="PorStavka"/>
   <xs:field xpath="@roidb"/>
  </xs:keyref>
  <xs:keyref name="Cenovnik_FK1" refer="Roba_PK">
   <xs:selector xpath="Cenovnik"/>
   <xs:field xpath="@roidb"/>
  </xs:keyref>
  <xs:keyref name="Porudzbenica_FK" refer="Dobavljac_PK">
   <xs:selector xpath="Porudzbenica"/>
   <xs:field xpath="@ppidb"/>
  </xs:keyref>
  <xs:keyref name="Cenovnik_FK2" refer="Dobavljac_PK">
   <xs:selector xpath="Cenovnik"/>
   <xs:field xpath="@ppidb"/>
  </xs:keyref>
  <xc:constraint type="xc:exRefInt">
     <xc:condition from="/Narucivanje/Cenovnik" fromPK="roidb" linkKeyref="ppidb"
                   to="/Narucivanje/PorStavka" link="/Narucivanje/Porudzbenica"
                   linkPK="poidb"/>
  </xc:constraint>
 </xs:element>
</xs:schema>
```

Листинг 120 XML Schema документ Narucivanje

Шема је проширена елементом xc:constraint, чији атрибут type има вредност exRefInt. Елемент xc:constraint овде наводи услов који повезује два елемента у шеми, преко трећег елемента. У датом примеру, ограничење повезује елементе Cenovnik (from) и PorStavka (to), а за повезивање се користи елемент Porudzbenica (link). Елементи Porudzbenica и PorStavka повезани су референцијалним интегритетом PorStavka@POIDB ⊆ Porudzbenica@POIDB. Да би се знало који добављач испоручује робу која је на поруџбеници, потребно је увести и страни кључ елемента Porudzbenica PPIDB.

Ограничење проширеног референцијалног интегритета, које је специфицирано у одељку 3.1.9, у овом примеру дефинисано је као:

ExRefIntPorStavka_Cenovnik(ExRefIntCon,
(PorStavka⋈Porudzbenica)[(ppidb, roidb)] ⊆ Cenovnik[(ppidb, roidb)],
((Porudzbenica, referencirajuća, {(insert, /), (update (poidb, ppidb), NoAction)}),
(PorStavka, referencirajuća, {(insert, NoAction), (update (poidb, roidb), NoAction)}),
(Cenovnik, referentna, {(delete, NoAction), (update (ppidb, roidb), NoAction)}))

Критичне операције за ограничење проширеног референцијалног интегритета су: унос новог елемента PorStavka, модификација елемента PorStavka, модификација елемента Porudzbenica,

модификација и брисање елемента Cenovnik. Контрола свих наведених операција је урађена и помоћу XQuery функција и помоћу тригера, што је описано у одељцима 5.3.1.3 и 5.3.1.4.

5.3.2.3 Контрола ограничења проширеног референцијалног интегритета помоћу XQuery функција

У овом одељку биће приказан XQuery код који спречава нарушавање ограничења проширеног референцијалног интегритета. XML документ над којим се врше провере овог ограничења дат је у Листинг 119.

5.3.2.3.1 Унос новог елемента PorStavka

Како у елементу PorStavka не постоји атрибут ppidb, који означава ознаку добављача који добавља робу, а ком се шаље поруџбеница, потребно је проверити да ли тај добављач добавља робу са задатим roidb. Приликом уноса новог елемента PorStavka, потребно је прво пронаћи ppidb добављача ком се та поруџбеница упућује. Затим је потребно у елементу Cenovnik проверити да ли тај добаљач добавља баш робу коју треба поручити. Ако је то задовољено, унос новог елемента PorStavka је могуће обавити.

Спајањем елемената PorStavka и Porudžbenica по атрибутима ppidb и roidb, треба да се добије онај ред из ценовника у којем се налазе исте вредности ppidb, односно roidb. То значи да се проверава да ли се поруџбеница шаље баш оном добављачу који ту робу и нуди.

$$PorStavka \bowtie Porudžbenica(@PPIDB+@ROIDB) \subseteq Cenovnik(@PPIDB+@ROIDB)$$

У Листинг 121 дате су XQuery функције у којима се врши унос новог елемента PorStavka. Функција insertPorStavka прво проверава да ли се може унети нови елемент PorStavka у функцији canInsertPorStavka, па ако је то могуће, врши се додавање новог елемента PorStavka у функцији doInsertPorStavka пре свих постојећих елемената PorStavka.

```
declare function local:canInsertPorStavka($NEW as element(PorStavka))
    as xs:boolean{
        let $exists := exists(doc('Narucivanje.xml')/Narucivanje/Cenovnik[@roidb =
$NEW/@roidb and @ppidb = doc('Narucivanje.xml')/Narucivanje/Porudzbenica[@poidb =
$NEW/@poidb]/@ppidb])
   return ($exists)
};

declare function local:doInsertPorStavka($NEW as element(PorStavka))
    as xs:boolean{
        let       $i       :=       update       insert       $NEW       preceding
doc('Narucivanje.xml')/Narucivanje/PorStavka[1]
        return true()
};

declare function local:insertPorStavka($NEW as element(PorStavka))
    as xs:boolean{
        let $res :=  if(local:canInsertPorStavka($NEW))
        then local:doInsertPorStavka($NEW)
        else false()
        return $res
};
```

Листинг 121 XQuery функција за унос елемента PorStavka

5.3.2.3.2 Модификација елемента PorStavka

Модификација елемента PorStavka односи се на промену вредности атрибута poidb или roidb. Елемент PorStavka је везни елемент између елемената Cenovnik и Porudzbenica.

Промена вредности атрибута roidb се дешава уколико треба променити робу на поруџбеници. За истог добављача, чији се ppidb налази у PorStavka, треба проверити да ли добавља и нову робу, чији roidb треба да замени стару вредност. Ако тај добављач добавља и нову робу, промена је могућа.

Промена вредности атрибута poidb се дешава ако треба променити добављача неке робе. Ако исту робу добавља и нови добављач, онда се њему може упутити нова поруџбеница, чија вредност poidb ће бити у елементу PorStavka.

Функција canUpdatePorStavka покушава да у ценовнику пронађе робу чија је вредност roidb једнака новој вредности roidb, а вредност атрибута ppidb једнака оном ppidb који се налази у поруџбеници, чији је poidb једнак новом poidb. Ако је промена могућа, у функцији doUpdatePorStavka врши се замена старог елемента PorStavka новим. Функција updatePorStavka је дата у Листинг 122.

```
declare function local:canUpdatePorStavka($NEW as element(PorStavka))
    as xs:boolean{
        let $exists := exists(doc('Narucivanje.xml')/Narucivanje/Cenovnik[@roidb =
$NEW/@roidb and @ppidb = doc('Narucivanje.xml')/Narucivanje/Porudzbenica[@poidb =
$NEW/@poidb]/@ppidb])
   return ($exists)
};

declare function local:doUpdatePorStavka($OLD as element(PorStavka), $NEW as
element(PorStavka))
    as xs:boolean{
        let $i := update replace
doc('Narucivanje.xml')/Narucivanje/PorStavka[@poidb=$OLD/@poidb and
@roidb=$OLD/@roidb] with $NEW
        return true()
};

declare function local:updatePorStavka($OLD as element(PorStavka), $NEW as
element(PorStavka))
    as xs:boolean{
        let $res :=  if(local:canUpdatePorStavka($NEW))
        then local:doUpdatePorStavka($OLD, $NEW)
        else false()
        return $res
};
```

Листинг 122 XQuery функција за модификацију елемента PorStavka

5.3.2.3.3 Модификација елемента Porudzbenica

Промена вредности кључа poidb се забрањује, јер је то кључ. Могуће је променити вредност атрибута ppidb. У том случају треба проверити да ли нови добављач добавља робу која се налази на тој поруџбеници, то јест у свим ставкама те поруџбенице. У елементима PorStavka постоје ставке са истом вредношћу poidb као у поруџбеници, такве да робу из тих ставки добавља и нови добављач.

У овој функцији праве се помоћни елементи Stavke. Елемент Stavke садржи све оне ставке из елемената PorStavka који се налазе на поруџбеници у којој се врши модификација. Пошто се користи исти код за генерисање функције и тригера, а у XML бази података Sedna у тригеру није могуће дефинисати променљиву, код који креира колекцију елемената Stavke, морао је бити два пута написан. Код другог генерисања елемента Stavke, употребљен је исти код који је филтриран условом да се гледају само оне ставке чија је шифра добављала једнака новој шифри добављача. Уколико је број поделемената Stavka у сваком од овако формираних колекција елемената Stavke једнак, онда је модификација могућа.

XQuery функција updatePorudzbenica, дата је у Листинг 123.

```
declare function local:canUpdatePorudzbenica($OLD as element(Porudzbenica), $NEW as
element(Porudzbenica))
    as xs:boolean{
        let $r :=  $OLD/@poidb = $NEW/@poidb and
        count ( for $ps in doc('Narucivanje.xml')/Narucivanje/PorStavka[@poidb =
$OLD/@poidb]
   let $roidb := $ps//@roidb
   return <Stavke>
      {
       for $c in doc('Narucivanje.xml')/Narucivanje/Cenovnik[@ppidb = $NEW/@ppidb
and @roidb = $roidb]
        return <Stavka ppidb = '{$c//@ppidb}' roidb = '{$c//@roidb}'/>
      }
      </Stavke>
     )
      =
      count((for $ps in doc('Narucivanje.xml')/Narucivanje/PorStavka[@poidb =
$OLD/@poidb]
         let $roidb := $ps//@roidb
         return <Stavke>
           {
            for $c in doc('Narucivanje.xml')/Narucivanje/Cenovnik[@ppidb =
$NEW/@ppidb and @roidb = $roidb]
             return <Stavka ppidb = '{$c//@ppidb}' roidb = '{$c//@roidb}'/>
           }
           </Stavke>
         )//Stavka[@ppidb = $NEW/@ppidb])
        return $r
};

declare function local:doUpdatePorudzbenica($OLD as element(Porudzbenica), $NEW as
element (Porudzbenica))
    as xs:boolean{
        let $i := update replace
doc('Narucivanje.xml')/Narucivanje/Porudzbenica[@poidb=$OLD/@poidb and
@ppidb=$OLD/@ppidb] with $NEW
        return true()
};

declare function local:updatePorudzbenica($OLD as element(Porudzbenica), $NEW as
element (Porudzbenica))
    as xs:boolean{
        let $res :=  if(local:canUpdatePorudzbenica($OLD, $NEW))
        then local:doUpdatePorudzbenica($OLD, $NEW)
        else false()
        return $res
};
```

Листинг 123 XQuery функција за модификацију елемента Porudzbenica

5.3.2.3.4 Модификација елемента Cenovnik

Модификација елемента Cenovnik се забрањује, ако се старе и нове вредности атрибута roidb и ppidb разликују, јер је унија ова два атрибута кључ елемента Cenovnik. Могућа је само модификација атрибута cena, ако се пронађе елемент Cenovnik који има исте вредности атрибута roidb и ppidb, као и нови елемент. XQuery функција за покушај модификације елемента Cenovnik дата је у Листинг 124.

```
declare function local:canUpdateCenovnik($OLD as element(Cenovnik), $NEW as
element(Cenovnik))
    as xs:boolean{
        let $r := $OLD/@roidb=$NEW/@roidb and $OLD/@ppidb=$NEW/@ppidb
   return $r
};
```

```
declare function local:doUpdateCenovnik($OLD as element(Cenovnik), $NEW as
element(Cenovnik))
    as xs:boolean{
        let $i := update replace doc("Narucivanje.xml")/Narucivanje/Cenovnik[@roidb
= $OLD/@roidb and @ppidb = $OLD/@ppidb] with $NEW
        return true()
};

declare function local:updateCenovnik($OLD as element(Cenovnik), $NEW as
element(Cenovnik))
    as xs:boolean{
        let $res := if(local:canUpdateCenovnik($OLD, $NEW))
                        then local:doUpdateCenovnik($OLD, $NEW)
                        else false()
        return $res
};
```

Листинг 124 XQuery функција за модификацију елемента Cenovnik

5.3.2.3.5 Брисање елемента Cenovnik

Приликом брисања елемента Cenovnik, проверава се да ли су вредности атрибута ppidb и roidb из елемента Cenovnik референциране у елементу PorStavka. Уколико јесу, брисање се забрањује. Провера се спроводи тако што се за добављача, чији ppidb се налази у елементу Cenovnik, проналазе поруџбенице. Уколико не постоји елемент PorStavka са тим вредностима атрибута ppidb и roidb, брисање одговарајућег елемента Cenovnik је могуће.

У Листинг 125 дате су XQuery функције у којима се врши брисање елемента Cenovnik. Функција deleteCenovnik прво проверава да ли је могуће брисање помоћу функције canDeleteCenovnik, а ако јесте, онда се извршава функција deDeleteCenovnik, у којој се врши брисање. Функција canDeleteCenovnik проверава да ли је могуће брисање елемента Cenovnik, према правилу које је мало пре описано.

```
declare function local:canDeleteCenovnik($OLD as element(Cenovnik))
    as xs:boolean{
        let $r := not(exists(doc('Narucivanje.xml')/Narucivanje/PorStavka[@roidb =
$OLD/@roidb and @poidb = doc('Narucivanje.xml')/Narucivanje/Porudzbenica[@ppidb =
$OLD/@ppidb]/@poidb]))
   return $r
};

declare function local:doDeleteCenovnik($OLD as element(Cenovnik))
    as xs:boolean{
        let $i := update delete doc("Narucivanje.xml")/Narucivanje/Cenovnik[@roidb
= $OLD/@roidb and @ppidb = $OLD/@ppidb]
        return true()
};

declare function local:deleteCenovnik($OLD as element(Cenovnik))
    as xs:boolean{
        let $res := if(local:canDeleteCenovnik($OLD))
                        then local:doDeleteCenovnik($OLD)
                        else false()
        return $res
};
```

Листинг 125 XQuery функција за брисање елемента Cenovnik

5.3.2.4 Контрола ограничења проширеног референцијалног интегритета помоћу тригера

У овом одељку биће приказани тригери који спречавају нарушавање ограничења проширеног референцијалног интегритета. XML документ над којим се врше провере овог ограничења дат

је у Листинг 119. Код за проверу услова који се користи у тригерима, идентичан је коду који се користи у функцијама, па неће бити посебно објашњаван.

5.3.2.4.1 Унос новог елемента PorStavka

Тригер се извршава пре додавања новог елемента PorStavka. Услов је исти као и код XQuery функције, која је дата у Листинг 121. Описани тригер је дат у Листинг 126.

```
CREATE TRIGGER "ProsireniRIPorStavkaBeforeInsert"
BEFORE INSERT
ON collection('ProsireniRI')/Narucivanje/PorStavka
FOR EACH NODE
DO {
 if (exists(fn:doc('Narucivanje', 'ProsireniRI')/Narucivanje/Cenovnik[@roidb =
$NEW/@roidb and @ppidb = fn:doc('Narucivanje',
'ProsireniRI')/Narucivanje/Porudzbenica[@poidb = $NEW/@poidb]/@ppidb]))
 then
  ($NEW)
 else
  error(xs:QName("ProsireniRIPorStavkaBeforeInsert"),"PorStavka ne moze da se doda,
posto se dodavanjem narusava prosireni referencijalni integritet");
}
```

Листинг 126 Тригер који се окида пре уноса новог елемента PorStavka

Унос елемента PorStavka из није могућа, јер робу са roidb R2 не добавља добављач чија је поруџбеница са poidp P1.

```
UPDATE INSERT
<PorStavka poidb="P1" roidb="R2" kolicina="1"/>
INTO fn:doc("Narucivanje", "ProsireniRI")/Narucivanje
```

Листинг 127 Покушај уноса елемента PorStavka

5.3.2.4.2 Модификација елемента PorStavka

Овај тригер се извршава пре покушаја модификације елемента PorStavka. Код тригера дат је у Листинг 128.

```
CREATE TRIGGER "ProsireniRIPorStavkaBeforeUpdate"
BEFORE REPLACE
ON collection('ProsireniRI')/Narucivanje/PorStavka
FOR EACH NODE
DO {
 if (exists(fn:doc('Narucivanje', 'ProsireniRI')/Narucivanje/
                    Cenovnik[@roidb = $NEW/@roidb and
            @ppidb = fn:doc('Narucivanje', 'ProsireniRI')/Narucivanje/
                          Porudzbenica[@poidb = $NEW/@poidb]/@ppidb]))
 then
  ($NEW)
 else
  error(xs:QName("ProsireniRIPorStavkaBeforeUpdate"),"PorStavka ne moze da se
promeni, posto se promenom narusava prosireni referencijalni integritet");
}
```

Листинг 128 Тригер који се окида пре модификације елемента PorStavka

Пример модификације елемента PorStavka дат је у Листинг 129. Ова модификација не може да се изврши, јер у XML документу из Листинг 119, робу R2 не добавља Dobavljac1, чија је ово поруџбеница.

```
UPDATE REPLACE $b in
      fn:doc("Narucivanje", "ProsireniRI")/
               Narucivanje/PorStavka[@poidb="P1" and @roidb="R1"]
WITH <PorStavka poidb="P1" roidb="R2" kolicina="1"/>
```

Листинг 129 Модификација елемента PorStavka

5.3.2.4.3 Модификација елемента Porudzbenica

Тригер се извршава пре покушаја модификације елемента Porudzbenica. Промена вредности кључа poidb се забрањује, јер је то кључ. Могуће је променити вредност атрибута ppidb. У том случају треба проверити да ли нови добављач добавља робу која се налази на тој поруџбеници, то јест у свим ставкама те поруџбенице. Код овог тригера дат је у Листинг 130.

```
CREATE TRIGGER "ProsireniRIPorudzbenicaBeforeUpdate"
BEFORE REPLACE
ON collection('ProsireniRI')/Narucivanje/Porudzbenica
FOR EACH NODE
DO {
 if ( $OLD/@poidb = $NEW/@poidb and
      count ( for $ps in fn:doc('Narucivanje','ProsireniRI')/Narucivanje/
      PorStavka[@poidb = $OLD/@poidb]
      let $roidb := $ps//@roidb
      return <Stavke>
            {
               for $c in fn:doc('Narucivanje', 'ProsireniRI')/Narucivanje/
               Cenovnik[@ppidb = $NEW/@ppidb and @roidb = $roidb]
               return <Stavka ppidb = '{$c//@ppidb}' roidb = '{$c//@roidb}'/>
            }
            </Stavke>
            )  =  count((for $ps in fn:doc('Narucivanje', 'ProsireniRI')/
                  Narucivanje/PorStavka[@poidb = $OLD/@poidb]
                  let $roidb := $ps//@roidb
                  return <Stavke>
                         {
                         for $c in fn:doc('Narucivanje', 'ProsireniRI')/
                         Narucivanje/Cenovnik[@ppidb = $NEW/@ppidb
                         and @roidb = $roidb]
                         return <Stavka ppidb = '{$c//@ppidb}'
                                        roidb = '{$c//@roidb}'/>
                         }
                         </Stavke>
                        ) //Stavka[@ppidb = $NEW/@ppidb])
    )
 then
  ($NEW)
 else
  error(xs:QName("ProsireniRIPorudzbenicaBeforeUpdate"),"Porudzbenica ne moze da se
izmeni, posto se izmenom narusava prosireni referencijalni integritet");
}
```

Листинг 130 Тригер који се окида пре модификације елемента Porudzbenica

Један пример модификације елемента Porudzbenica дат је у Листинг 131, а неће бити могућ, јер нови добављач Dobavljac3 не нуди све што и стари добављач Dobavljac1.

```
UPDATE REPLACE $b in
   fn:doc("Narucivanje", "ProsireniRI")/
            Narucivanje/Porudzbenica[@poidb="P1" and @ppidb="Dobavljac1"]
WITH <Porudzbenica poidb="P1" ppidb="Dobavljac3"/>
```

Листинг 131 Модификација елемента Porudzbenica

5.3.2.4.4 Модификација елемента Cenovnik

У елементу Cenovnik, могућа је само модификација цене. Забрањује се модификација делова примарног кључа, ppidb или roidb. Тригер је дат у Листинг 132.

```
CREATE TRIGGER "ProsireniRICenovnikBeforeUpdate"
BEFORE REPLACE
ON collection('ProsireniRI')/Narucivanje/Cenovnik
FOR EACH NODE
DO {
 if (exists(fn:doc('Narucivanje', 'ProsireniRI')/Narucivanje/Cenovnik[@roidb =
$OLD/@roidb and @ppidb=$OLD/@ppidb]) and $OLD/@roidb=$NEW/@roidb and
$OLD/@ppidb=$NEW/@ppidb)
 then
  ($NEW)
 else
  error(xs:QName("ProsireniRICenovnikBeforeUpdate"),"PK od Cenovnik se ne moze
menjati");
}
```

Листинг 132 Тригер који се окида пре модификације елемента Cenovnik

Један покушај модификације елемента Cenovnik дат је у Листинг 133, а неуспешан је, јер није дозвољена модификација кључа.

```
UPDATE REPLACE $b in
   fn:doc("Narucivanje", "ProsireniRI")/
           Narucivanje/Cenovnik[@roidb="R1" and @ppidb="Dobavljac1"]
WITH <Cenovnik roidb="R2" ppidb="Dobavljac1"/>
```

Листинг 133 Модификација елемента Cenovnik

5.3.2.4.5 Брисање елемента Cenovnik

Тригер из Листинг 134 се извршава пре покушаја брисања елемента Cenovnik.

```
CREATE TRIGGER "ProsireniRICenovnikBeforeDelete"
BEFORE DELETE
ON collection('ProsireniRI')/Narucivanje/Cenovnik
FOR EACH NODE
DO {
 if (not(exists(fn:doc('Narucivanje', 'ProsireniRI')/Narucivanje/PorStavka[@roidb =
$OLD/@roidb and @poidb = fn:doc('Narucivanje',
'ProsireniRI')/Narucivanje/Porudzbenica[@ppidb = $OLD/@ppidb]/@poidb])))
 then
  ($OLD)
 else
  error(xs:QName("ProsireniRICenovnikBeforeDelete"),"Cenovnik ne moze da se obrise,
posto se brisanjem narusava prosireni referencijalni integritet");
}
```

Листинг 134 Тригер који се окида пре брисања елемента Cenovnik

Брисање елемента Cenovnik дато у Листинг 135 није успешно, јер ценовник са овим кључем има ставке у елементу PorStavka.

```
UPDATE DELETE
fn:doc("Narucivanje", "ProsireniRI")/
        Narucivanje/Cenovnik[@roidb="R3" and @ppidb="Dobavljac2"]
```

Листинг 135 Брисање елемента Cenovnik

5.3.3 Ограничење инверзног референцијалног интегритета

У овом одељку приказано је ограничење инверзног референцијалног ограничења. Прво је описано ограничење у релационом моделу података, а затим у XML моделу података. У постојећим XML СУБП не постоји имплементација овог ограничења, нити га XML Schema подржава. Проширење шеме описано је у одељку 3.1.8, а на основу њега генератор описан у поглављу 4 генерише XQuery функције и тригере, који у потпуности реализују проверу овог ограничења, приликом извршавања операција ажурирања.

5.3.3.1 Ограничење инверзног референцијалног интегритета у релационом моделу података

У релационом моделу података, ограничење инверзног референцијалног интегритета описује егзистенцијалну зависност која се јавља и у ЕР моделу података. Постојање инверзног референцијалног интегритета условљено је постојањем одговарајућег референцијалног интегритета [Mog00].

Посматрајмо шеме релација N_i и N_j, и скупове обележја X и Y, где је Y кључ шеме релације N_j. Између шема релација N_i и N_j могуће је дефинисати следећа ограничења:

- ограничење референцијалног интегритета: $N_i[X] \subseteq N_j[Y]$ и
- ограничење инверзног референцијалног интегритета: $N_j[Y] \subseteq N_i[X]$.

Инверзни референцијални интегритет указује да је свака R_j вредност повезана са бар једном R_i вредношћу. Торке релације $r(N_j)$ егзистенцијално су зависне од торки релације $r(N_i)$. За сваку Y вредност у релацији $r(N_j)$, мора постојати бар једна торка са истом X вредношћу у релацији $r(N_i)$.

Постојање ограничења инверзног референцијалног интегритета изазива блокирање уноса нове торке, јер важи и ограничење референцијалног интегритета. Дефинисањем ограничења инверзног референцијалног интегритета $N_j[Y] \subseteq N_i[X]$ уводи се ограничење $r(N_i)[X \neq \omega] = r(N_j)[Y]$, долази до привидне узајамне блокаде операција над шемама релација N_i и N_j. Блокада се састоји у томе што се у појаву над шемом релације $r(N_i)$ не може уписати нова торка са ненула вредностима за свако обележје A из X, ако у појави $r(N_j)$ не постоји торка са истом Y вредношћу. Такође, не може се уписати нова торка у појаву $r(N_j)$ ако у појави $r(N_i)$ не постоји торка са истом X вредношћу. Према томе, ако је релација $r(N_j)$ празна, појава $r(N_i)$ би требало да увек садржи торке у којима или бар једно или свако обележје из X има нула вредност. Проблем уписа торки у релацију $r(N_j)$ се решава: или уписом једне торке са ненула вредностима за сва обележја из X у $r(N_i)$ и једне торке са истом Y вредношћу у релацију $r(N_j)$, или таквом модификацијом неке постојеће торке из $r(N_i)$ након које ниједно обележје из X неће имати нула вредност и уписом једне торке са истом Y вредношћу у релацију $r(N_j)$. Провера усаглашености базе података са ограничењима референцијалног и инверзног референцијалног интегритета се врши одложено, на крају програма.

Пример 12

Посматрајмо шеме релација:

- Fakultet({FacId, NazivFac}, {FacId}) и
- Departman({DepId, FacId, NazivDep}, {DepId+FacId}).

Шема базе података је приказана на Слика 10.

Слика 10 Шема базе података FakultetDepartman

У овој шеми базе података важе и ограничења:

- ограничење референцијалног интегритета Departman[FacId] ⊆ Fakultet[FacId] и
- ограничење инверзног референцијалног интегритета Fakultet[FacId] ⊆ Deprtman[FacId].

Значење овог ограничења инверзног референцијалног интегритета је да сваки факултет мора да има бар један департман. Задовољење ограничења инверзног референцијалног интегритета у релационом моделу података се проверава приликом операција уноса, брисања и модификације.

Није могуће унети нови департман са ненула вредностима за обележје FacId, док не постоји торка у релацији Fakultet са истом вредности FacId. Такође, није могуће унети нову торку у релацију Fakultet, докле год не постоји бар један департман који припада том факултету. Ограничење инверзног референцијалног интегритета може бити нарушено приликом уноса торке у релацију Fakultet, приликом брисања из релације Departman, или приликом модификације торке у релацији Departman.

Приликом уноса нове торке у релацију Fakultet, треба унети и торке у релацију Departman. Како је провера ограничења референцијалног интегритета већ уграђена у систем за управљање базом података, за страни кључ FacId могуће је унети само постојеће вредности из релације Fakultet. Након тога активира се процедура која ће проверити да ли за сваку торку из релације Fakultet постоји и одговарајућа торка из релације Departman која садржи исту вредност FacId. Окидач треба да се покрене одложено, на крају трансакције, након покретања окидача за управљање уписом нове торкеу релацију Departman. Унос нове торке у релацију Fakultet би био онемогућен, јер у том моменту свакако не постоји торка у релацији Departman која има исту вредност обележја FacId. То је ситуација у којој не би био могућ упис ни у релацију Fakultet ни у релацију Departman, а она се може решити на један од следећих начина, што је описано у раду [Ale13]:

- инверзни референцијални интегритет се реализује само у оквиру апликација информационог система, без имплементирања окидача инверзног референцијалног интегритета на нивоу базе података,
- управљање извршењем окидача базе података,
- изградња посебног корисничког механизма за упис и торке у релацију Департман и торке у релацију Факултет у оквиру једне трансакције.

5.3.3.2 Ограничење инверзног референцијалног интегритета у XML моделу података

Као и у релационом моделу података, инверзни референцијални интегритет се може дефинисати и у XML моделу података [Vid15]. Уколико између два елемента постоји референцијални интегритет, али и егзистенцијална зависност, онда се може говорити о инверзном референцијалном интегритету.

Ако између два елемента важи референцијални интегритет, а надређени елемент егзистенцијално зависи од подређеног елемента, тада се може дефинисати и инверзни референцијални интегритет. Акције које могу нарушити ограничење инверзног референцијалног интегритета су унос надређеног елемента, брисање подређеног елемента и модификација подређеног елемента.

Приликом уноса новог надређеног елемента, мора се проверити да ли постоји подређени елемент чија је вредност страног кључа једнака вредности примарног кључа надређеног елемента. Ово је органичење референцијалног интегритета које може бити дефинисано у самом XML Schema документу, али и помоћу XQuery израза, што је већ описано у одељку 5.2.3. Због постојања инверзног референцијалног интегритета, такође је потребно проверити да ли

за сваки надређени елемент, постоји бар један подређени елемент који се на њега референцира. Како унос нових елемената не би изазвао закључавање, потребно је да се провера обави након уноса.

Приликом брисања подређеног елемента, потребно је проверити да ли је он последњи који се референцира на свог надређеног. Ако јесте, такво брисање треба забранити, или превезати неки други надређени елемент.

Модификација страног кључа у подређеном елементу је могућа ако надређени елемент има још подређених елемената.

Ограничење инверзног референцијалног интегритета може да се реализује XQuery функцијама, као и помоћу тригера . На основу спецификације дате у одељку 3.1.8, генерисане су XQuery функције и тригери, који ће, у зависности од тога да ли је ограничење задовољено или не, дозволити одговарајућу акцију.

Пример 13

У наставку су приказани XML документ (Листинг 136), као и XML Schema документ који га описује (Листинг 137). Коренски елемент садржи елементе Fakultet и Departman. Елемент Fakultet има два атрибута, FacId и FacNaziv, а елемент Departman три атрибута, DepId, FacId и DepNaziv, где је FacId страни кључ и представља ознаку факултета којем припада одређени департман. Елемент Departman мора да има и атрибут који представља страни кључ, да би било могуће дефинисање инверзног референцијалног интегритета. Увођењем страног кључа FacId, кључ елемента Departman је унија атрибута DepId и FacId, па се исте вредности атрибута DepId могу јавити и на различитим факултетима.

```
<Database xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="fakultetDepartman.xsd">
  <Fakultet FacId="FIL" FacNaziv="Filozofski fakultet"/>
  <Fakultet FacId="PMF" FacNaziv="Prirodno matematicki fakultet"/>
  <Fakultet FacId="FTN" FacNaziv="Fakultet tehničkih nauka"/>
  <Fakultet FacId="PF" FacNaziv="Pravni fakultet"/>
  <Departman DepId="DMI" FacId="FIL" DepNaziv="Departman za matematiku i
informatiku"/>
  <Departman DepId="DMI" FacId="PMF" DepNaziv="Departman za matematiku i
informatiku"/>
  <Departman DepId="DF" FacId="PMF" DepNaziv="Departman za fiziku"/>
  <Departman DepId="DB" FacId="PMF" DepNaziv="Departman za biologiju"/>
  <Departman DepId="DA" FacId="FTN" DepNaziv="Departman za arhitekturu i dizajn"/>
  <Departman DepId="DG" FacId="FTN" DepNaziv="Departman za gradjevinarstvo i
geodeziju"/>
  <Departman DepId="MP" FacId="PF" DepNaziv="Medjunarodno pravo"/>
</Database>
```

Листинг 136 XML документ FakultetDepartman

У XML Schema документу је описан претходни XML документ. Дати су типови података за сваки од атрибута, а дефинисани су примарни и страни кључеви. Кључ елемента Fakultet је FacId, а кључ елемента Departman је унија атрибута DepId и FacId.

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="Database">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Fakultet" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="FacId" type="xs:string" use="required"/>
          <xs:attribute name="FacNaziv" type="xs:string" use="required"/>
        </xs:complexType>
```

```
      </xs:element>
      <xs:element name="Departman" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="DepId" type="xs:string" use="required"/>
          <xs:attribute name="DepNaziv" type="xs:string" use="required"/>
          <xs:attribute name="FacId" type="xs:string" use="optional"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:key name="Fakultet_PK">
    <xs:selector xpath="Fakultet"/>
    <xs:field xpath="@FacId"/>
  </xs:key>
  <xs:key name="Departman_PK">
    <xs:selector xpath="Departman"/>
    <xs:field xpath="@DepId"/>
    <xs:field xpath="@FacId"/>
  </xs:key>
  <xs:keyref name="Departman_FK" refer="Fakultet_PK">
    <xs:selector xpath="Departman"/>
    <xs:field xpath="@FacId"/>
  </xs:keyref>
  <xc:constraint type="xc:invRefInt">
    <xc:condition from="/Database/Fakultet" to="/Database/Departman"
                  keyref="FacId"/>
  </xc:constraint>
 </xs:element>
</xs:schema>
```

Листинг 137 XML Schema документ FakultetDepartman

Шема је проширена елементом xc:constraint са вредношћу атрибута type invRefInt. Поделемент елемента xc:constraint је елемент xc:condition. Атрибут from указује на то који је елемент надређени, атрибут to на елемент који је подређени. Атрибут keyref садржи назив атрибута који представља страни кључ у подређеној релацији. Он је посебно наведен, да би страни кључ био лакше издвојен, што би иначе било могуће и проласком кроз шему.

Ограничење инверзног референцијалног интегритета је, на основу спецификације дате у 3.1.8, у овом примеру приказано на следећи начин:

IRICon_Fakultet_Departman(IRICon,
Fakultet@FacId⊆Departman@FacId,
(Fakultet, referencirajuća, {(insert, NoAction), (update (FacId), NoAction)}),
(Departman, referentna, {(delete, NoAction), (update, NoAction)}))

У присуству инверзног референцијалног интегритета проблем се појављује код: уноса новог факултета, који мора да има бар један департман, брисања департмана, уколико је он једини департман на том факултету, промене матичног факултета неком департману, уколико је он једини на том факултету.

5.3.3.3 Контрола ограничења инверзног референцијалног интегритета помоћу XQuery функција

У наставку су дате XQuery функције који за описане акције уноса, брисања и модификације, проверавају да ли оне могу да се спроведу, и ако могу, спроводе те акције. XML документ над којим се врше провере овог ограничења дат је у Листинг 136.

5.3.3.3.1 Унос новог факултета и департмана

Унос новог факултета без департмана није дозвољен. Да би факултет био унет, мора се уз факултет додати и департман, а то је могуће само ако се привремено искључи провера важења инверзног референцијалног интегритета. То се постиже засебном XQuery функцијом, која ће привремено искључити тригер (биће описан у наставку овог поглавља), додати факултет и департман, проверити да ли је инверзни референцијални интегритет задовољен. Ако није задовољен, бришу се и факултет и департман. На крају се тригер поново укључује. Нови факултет се додаје пре свих осталих елемената Fakultet. Функција за унос новог факултета дата је у Листинг 138.

```
declare function local:canInsertFakultet() as xs:boolean {
    let $r := exists(doc('Trigger')/Trigger/Off)
    return $r
};

declare function local:doInsertFakultet($NEW as element(Fakultet)) as xs:boolean {
    let $r :=
     update insert $NEW preceding doc('fakultetDepartman.xml')/Database/Fakultet[1]
    return true()
};

declare function local:insertFakultet($NEW as element(Fakultet)) as xs:boolean
{
    let $canInsert := local:canInsertFakultet()
    let $r := if ($canInsert)
        then
            local:doInsertFakultet($NEW)
        else
            false()
  return $r
};
```

Листинг 138 XQuery функција insertFakultet

Извршавање функције insertFakultet је повезано са извршавањем функције insertDepartman, која је описана у Листинг 139 XQuery функција insertDepartman. Додавање департмана се мора извршити одмах након додавања факултета. Након тога се проверава да ли су сви елементи успешно додати.

Нови департман је могуће унети, ако постоји факултет којем он припада. Ова провера се односи на ограничење обичног референцијалног интегритета. Функција insertDepartman се извршава одмах након извршења функције insertFakultet.

```
declare function local:canInsertDepartman($NEW as element(Departman)) as xs:boolean
{
    let $r := exists(
              doc('fakultetDepartman.xml')/Database/Fakultet[@FacId=$NEW/@FacId])
    return $r
};

declare function local:doInsertDepartman($NEW as element(Departman)) as xs:boolean
    {
    let $r :=
          update $NEW preceding doc('fakultetDepartman.xml')/Database/Departman[1]
    return true()
};

declare function local:insertDepartman($NEW as element(Departman)) as xs:boolean
    {
        let $canInsert := local:canInsertDepartman($NEW/@FacId)
        let $r := if ($canInsert)
        then
```

```
            local:doInsertDepartman($NEW)
        else
            false()
        return $r
};
```

Листинг 139 XQuery функција insertDepartman

Функција која врши укључивање и искључивање тригера, дата је у следећем листингу (Листинг 140). Тригер ће бити искључен пре покушаја уноса новог факултета. Уносе се елементи Fakultet и Departman, а након тога ће тригер поново бити укључен. Провера се извршава тако што се преброје департмани који припадају том факултету, и мора их бити више од нула, да би ограничење инверзног референцијалног интегритета било задовољено.

```
declare function local:triggerOn() as xs:boolean {
    let $r :=  update replace doc('Trigger')/Trigger/Off
    with
    <On />
    let $res :=
    update delete

doc('fakultetDepartman.xml')/Database/Fakultet[@FacId=doc('fakultetDepartman.xml')/
Database/Fakultet/@FacId[not(. =
doc('fakultetDepartman.xml')/Database/Departman/@FacId)]]
    return true()
};

declare function local:triggerOff() as xs:boolean {
  let $r :=  update replace doc('Trigger')/Trigger/On
    with
    <Off />
    return true()
};
```

Листинг 140 XQuery функције triggerOn и triggerOff

5.3.3.3.2 Брисање постојећег департмана

Проблем код брисања постојећег департмана се може јавити у случају да је тај департман једини на свом факултету. Због ограничења инверзног референцијалног интегритета то није могуће, јер би факултет остао без департмана. Због тога се пре брисања департмана мора проверити колико на његовом факултету има департмана. Уколико има више од једног департмана, брисање департмана је могуће. Функција deleteDepartman дата је у Листинг 141.

```
declare function local:canDeleteDepartman($OLD as element(Departman))
    as xs:boolean {
    let $res := not(count(doc('fakultetDepartman.xml')/Database/
                          Departman[@FacId=$OLD/@FacId]) <= 1)
    return $res
};

declare function local:deleteDepartman($OLD as element(Departman))
    as xs:boolean {
        let $CanDelete := local:canDeleteDepartman($OLD)
        let $res := if($CanDelete)
                    then update
                         delete doc('fakultetDepartman.xml')/Database/
                         Departman[@DepId=$OLD/@DepId and @FacId=$OLD/@FacId]
                    else true()
        return $CanDelete
};
```

Листинг 141 XQuery функција *deleteDepartman*

5.3.3.3.3 Измена факултета којем припада одређени департман

Уколико је потребно променити матични факултет неком департману, то значи да се у елементу Departman мења вредност атрибута FacId. Ову промену је могуће урадити у присуству инверзног референцијалног интегритета, уколико тај департман није једини на основном факултету којем припада. У функцији canUpdateDepartman проверава се колико факултет има департмана. Ако је број департмана већи од један, могуће је један од тих департмана превезати на нови факултет, што значи да се вредност атрибута FacId у том департману може заменити вредношћу атрибута FacId неког другог факултета. Функција у којој се врши модификација елемента Departman дата је у Листинг 142.

```
declare function local:canUpdateDepartman($OLD as element(Departman))
    as xs:boolean {
    let $res :=
             not(count(doc('fakultetDepartman.xml')/Database/
             Departman[@FacId=$OLD/@FacId]) <= 1)
    return $res
};

declare function local:updateDepartman($OLD as element(Departman), $NEW as
element(Departman))
    as xs:boolean {
        let $iric := local:canUpdateDepartman($OLD)
        let $r := if ($iric)
                then update replace
                 doc('fakultetDepartman.xml')/Database/Departman[@DepId=$OLD/@DepId
                 and @FacId=$OLD/@FacId] with $NEW
                else false()
        return $iric
};
```

Листинг 142 XQuery функција *updateDepartman*

5.3.3.4 Контрола инверзног референцијалног интегритета помоћу тригера

У наставку су дату тригери који за описане акције уноса, брисања и модификације, проверавају да ли оне могу да се спроведу. XML документ над којим се врше провере овог ограничења дат је у Листинг 136.

5.3.3.4.1 Унос новог факултета и департмана

Унос новог факултета није могућ, ако се за њега не унесе и бар један припадајући департман. Због тога се, пре покушаја додавања, тригер за проверу инверзног референцијалног интегритета мора искључити.

Тригер IRICInsertFakultet се извршава пре покушаја додавања новог елемента Fakultet. Ако је укључен тригер за проверу инверзног референцијалног интегритета, унос новог факултета се забрањује. Код тригера је дат у Листинг 143.

```
CREATE TRIGGER "IRICInsertFakultet"
BEFORE INSERT
ON collection('IRIC')/Database/Fakultet
FOR EACH NODE
DO {
 if (exists(fn:doc('Trigger', 'triggers')/Trigger/On))
 then
  error(xs:QName("IRICInsertFakultet"),"Fakultet ne moze da se ubaci bez
Departmana")
 else
   ($NEW);}
```

Листинг 143 Тригер IRICInsertFakultet

Тригер IRICInsertDepartmanBefore (Листинг 144) извршава се пре додавања новог елемента Departman. Ако не постоји факултет којем тај департман треба да припада, додавање се забрањује.

```
CREATE TRIGGER "IRICInsertDepartmanBefore"
BEFORE INSERT
ON collection('IRIC')/Database/Departman
FOR EACH NODE
DO {
 if (not (exists(fn:doc('FakultetDepartman', 'IRIC')/
                Database/Fakultet[@FacId=$NEW/@FacId])))
 then
  error(xs:QName("IRICInsertDepartmanBefore"),"Departman ne moze da se ubaci, posto
ne postoji Fakultet na koji se referencira")
 else
   ($NEW);
}
```

Листинг 144 Тригер IRICInsertDepartmanBefore

Након уноса факултета и департмана, проверава се да ли сваки факултет има бар један департман, а ако постоји неки од факултета без департмана, такви факултети се одмах бришу из базе. Тригер IRICUpdateTrigger се активира када год се мења тригер – било да се укључује или искључује. Бришу се они факултети чија вредност атрибута FacId не постоји ни у једном елементу Departman. Код тригера IRICUpdateTrigger дат је у Листинг 145.

```
CREATE TRIGGER "IRICUpdateTrigger"
AFTER REPLACE
ON collection('triggers')/Trigger
FOR EACH STATEMENT
DO {
  UPDATE DELETE
         if (exists(fn:doc('Trigger', 'triggers')/Trigger/On))
         then fn:doc('FakultetDepartman', 'IRIC')/
           Database/Fakultet[@FacId=fn:doc('FakultetDepartman', 'IRIC')/
           Database/Fakultet/
           @FacId[not(. =
                       fn:doc('FakultetDepartman', 'IRIC')/
                               Database/Departman/@FacId)]]
         else
           ();
}
```

Листинг 145 Тригер IRICUpdateTrigger

У Листинг 146 приказано је како се врши унос новог факултета и департмана. Као што је претходно описано, прво се искључује тригер, тако што се у XML документу Trigger.xml постојећи тригер замени новим, који је искључен. Затим се додају нови факултет и департман. На крају се поново укључује тригер, тако што се постојећи тригер замењује новим који је укључен.

```
fn:doc("FakultetDepartman", "IRIC")

(: iskljuci triger :)
UPDATE
REPLACE $r in fn:doc("Trigger", "triggers")/Trigger
WITH
<Trigger>
 <Off />
</Trigger>

(: dodaj fakultet :)
UPDATE INSERT
<Fakultet FacId="FIL2" FacNaziv="Filozofski fakultet2"/>
INTO fn:doc("FakultetDepartman", "IRIC")/Database
```

```
(: dodaj departman u fakultet :)
UPDATE INSERT
<Departman DepId="KP" FacId="FIL2" DepNaziv="Krivicno pravo"/>
INTO fn:doc("FakultetDepartman", "IRIC")/Database

(: ukljuci triger :)
UPDATE
REPLACE $r in fn:doc("Trigger", "triggers")/Trigger
WITH
<Trigger>
 <On />
</Trigger>
```

Листинг 146 Додавање новог факултета и департмана који му припада

5.3.3.4.2 Брисање постојећег департмана

Департман је могуће избрисати ако није последњи на факултету којем припада. Тригер IRICDeleteDepartmanBefore (Листинг 147) окида се пре брисања елемента Departman.

```
CREATE TRIGGER "IRICDeleteDepartmanBefore"
BEFORE DELETE
ON collection('IRIC')/Database/Departman
FOR EACH NODE
DO {
 if (count(fn:doc('FakultetDepartman', 'IRIC')/Database/
             Departman[@FacId=$OLD/@FacId]) <= 1)
 then
  error(xs:QName("IRICDeleteDepartmanBefore"),"Departman ne moze da se obrise,
posto je poslednji za svoj Fakultet")
 else
   ($OLD);
}
```

Листинг 147 Тригер IRICDeleteDepartmanBefore

Извршавање наредбе из Листинг 148 изазива окидање тригера IRICDeleteDepartmanBefore, који проверава да ли је могуће избрисати задати департман. Ако тригер не изазове грешку, брисање је могуће.

```
UPDATE DELETE fn:doc("FakultetDepartman", "IRIC")/
                     Database/Departman[@DepId="KP"]
```

Листинг 148 Брисање департмана са DepId KP

5.3.3.4.3 Измена факултета којем припада одређени департман

Промена вредности атрибута FacId у елементу Departman је могућа, ако тај департман није последњи на свом факултету. Ако јесте једини, онда се брисање забрањује, јер би факултет остао без департмана.

```
(: ne dozvoljava prevezivanje poslednjeg departmana iz nekog fakulteta na neki novi
fakultet :)
CREATE TRIGGER "IRICUpdateDepartmanBefore"
BEFORE REPLACE
ON collection('IRIC')/Database/Departman
FOR EACH NODE
DO {
 if (count(fn:doc('FakultetDepartman','IRIC')/Database/
             Departman[@FacId=$OLD/@FacId]) <= 1)
 then
  error(xs:QName("IRICUpdateDepartmanBefore"),"Departman ne moze da se preveze na
drugi Fakultet, posto je poslednji za svoj Fakultet")
 else
   ($NEW);
```

```
}
```

Листинг 149 Тригер IRICUpdateDepartmanBefore

Када се у бази података изврши наредба из Листинг 150, изазива се окидање тригера IRICUpdateDepartmanBefore, који проверава да ли је могуће департман повезати са другим факултетом.

```
UPDATE REPLACE $b in fn:doc("FakultetDepartman", "IRIC")/
                              Database/Departman[@DepId="MP"]
with
<Departman DepId="MP" FacId="FTN" DepNaziv="Medjunarodno pravo"/>
```

Листинг 150 Премештај департмана на факултет са FacId FTN

5.3.4 Закључак

У одељку 5.3 дати су примери генерисаног кода који реализују три врсте ограничења, које нису уопште подржане XML Schema-ом, нити су реализоване у постојећим XML СУБП. То су: проширено ограничење торке, ограничење проширеног референцијалног интегритета и ограничење инверзног референцијалног интегритета. На основу спецификација датих у одељцима 3.1.4, 3.1.9 и 3.1.8, генерисане су XQuery функције и тригери, који врше проверу ових ограничења приликом извршавања операција ажурирања. Програмери добијају могућност да у пракси употребљавају код који је генерисан на овај начин, уместо да ручно имплементирају провере ограничења, чиме се побољшава ефективност њиховог рада.

5.4 Закључак

Основни циљ поглавља 5 је да се на основу спецификација типова ограничења дефинисаних у поглављу 3 генерише код, који може да врши проверу конкретних ограничења у одабраним XML СУБП. Тај циљ је у потпуности остварен. Као што је то уобичајено у релационом моделу, и у XML моделу података све почиње спецификацијом. Пројектанти могу да запишу све типове ограничења користећи нотацију описану у поглављу 3. Сви типови ограничења описани у овој дисертацији обогаћују XML модел података, модификацијом постојећих типова ограничења и додавањем нових. На основу описа ових типова ограничења, генератор кода производи XQuery функције и тригере, који реализују проверу конкретних ограничења. Директну корист од овог генерисаног кода имају програмери, који више не морају ручно да спроводе провере ограничења, већ је довољно да укључе генерисани код у своје пројекте и тиме битно утичу на побољшање ефективности свог рада. Потребно је само једним елементом у XML Schema-и описати ограничење, а добиће се аутоматски генерисан код који валидира то ограничење. На примеру ограничења инверзног референцијалног интегритета један ред у XML Schema-и производи тридесетак линија кода који валидира то ограничење. Програмер не мора да пише овај код, већ је он аутоматски генерисан на основу описа ограничења у XML Schema-и.

6 Закључак и правци даљег истраживања

Ова докторска дисертација бави се истраживањем начина спецификације и имплементације ограничења података у XML базама података. Ограничења над подацима у бази података било ког информационог система омогућују поштовање великог броја правила пословања из реалног система. Ограничења такође обезбеђују конзистентност података у информационом систему. Процедура која обезбеђује аутоматску проверу ограничења убрзава развој, смањује број грешака и олакшава унос података. Због тога, веома је битно у изабраним моделима података дефинисати оквир за презентовање и имплементацију ограничења у системима база података.

Иако су релациони системи за управљање базама података и даље највише заступљени у комерцијалној примени, последњих година интензивно се развијају XML системи за управљање базама података, или се врши надоградња постојећих релационих или објектно-релационих система, тако да обезбеђују све значајнију подршку концепата XML модела података. У XML системима база података подаци се чувају у XML датотекама и омогућено је њихово ажурирање и претрага. Ограничења над подацима у XML моделу података нису у довољној мери истражена, као што је то случај са релационим моделом података. На основу прегледа литературе и постојећих стандарда, закључено је да актуелни XML системи за управљање базама података подржавају само основне типове ограничења, док је подршка осталих, сложенијих типова ограничења запостављена. Стога, предмет истраживања у овој дисертацији јесте дефинисање приступа спецификацији и имплементацији и основних и сложених типова ограничења у XML моделу података.

Главни резултати ове дисертације су следећи:

- дата је типизација ограничења у XML базама података,
- по угледу на ограничења у релационом моделу података, идентификоване су три групе ограничења: а) ограничења које постоје у XML моделу података и која су дефинисана на задовољавајући начин, б) ограничења које постоје у XML моделу података, а чију дефиницију треба проширити и в) ограничења која не постоје у XML моделу података, а XML модел података треба њима допунити,
- дат је формални модел свих типова ограничења; овај модел обухвата област дефинисаности, формализам за записивање, област интерпретације и правило за валидацију,
- реализован је генератор кода који на основу формалног записа ограничења генерише XQuery функције и тригере који имплементирају ограничења и
- обављена је верификација предложеног приступа, тако што је извршена његова провера путем имплементације ограничења у две репрезентативне XML базе података, eXist и Sedna.

Главни допринос ове дисертације је предложена типизација ограничења у XML базама података, која у таквој форми није пронађена у доступној литератури. Осим детаљне анализе постојећих ограничења и прегледа типова ограничења у литератури, у овој дисертацији дат је предлог нових типова ограничења, који су дефинисани по угледу на типове ограничења у релационом моделу података. На основу анализе ограничења у релационим базама података, идентификоване су три групе типова ограничења: она које постоје у XML моделу података, она која треба проширити, и она која треба додати. На овај начин оправдана је хипотеза 1, формулисана у Уводу ове дисертације.

Прву групу типова ограничења чине: ограничење домена, ограничење вредности атрибута и ограничење јединствене вредности. С обзиром на то да постојећи XML модел података не подржава недостајућу (*null*) вредност атрибута, дат је предлог реализације јединствене вредности атрибута, која може бити *null*.

Другу групу типова ограничења чине: ограничење торке, ограничење кључа и ограничење референцијалног интегритета. Ови типови ограничења постоје у XML моделу података, али је било потребно дефинисати додатне механизме за њихову контролу. Код ограничења торке додат је механизам позива произвољне XQuery функције, која обавља додатну проверу вредности атрибута у торци. За ограничење кључа предложен је механизам за забрану промене вредности примарног кључа. У ограничењу референцијалног интегритета додат је механизам који реализује акције забране извршења операције (*NoAction*, или *Restrict*) и каскадног извршења операције (*Cascade*), над елементима који учествују у ограничењу.

Трећу групу типова ограничења чине проширено ограничење торке, ограничење проширеног референцијалног интегритета и ограничење инверзног референцијалног интегритета. За ове типове ограничења нису постојали одговарајући концепти у XML моделу података, па су они морали бити комплетно моделовани, што такође представља један од важних доприноса ове докторске дисертације.

За све наведене типове ограничења дат је формални модел који обухвата област дефинисаности, формализам за записивање, област интерпретације и правило за валидацију. Правила за валидацију дефинисана су псеудокодом, који се, у поступку имплементације, трансформише у XQuery код.

Нотација за запис ограничења дата је у облику проширеног XML Schema језика, чиме је обезбеђено једноставно парсирање специфицираних ограничења, и генерисање кода. На основу записа ограничења, генератор кода производи XQuery функције и тригере. Спецификација генератора кода дата је UML-ом, а имплементација је извршена у програмском језику Java. Код је генерисан употребом *FreeMarker Template Engine* библиотеке. Хипотеза 2 дата у овој дисертацији доказана је реализацијом овог генератора.

У дисертацији је приказана студија случаја која обухвата ограничења свих наведених типова и која је реализована путем два репрезентативна XML система за управљање базама података, eXist и Sedna. Постоје два облика генерисаног кода за реализацију ограничења, а то су XQuery функције и тригери. XQuery функције су коришћене у eXist бази података, јер се показује да је у њој подршка за тригере недовољна. Хипотеза 3 доказана је имплементацијом ограничења у оквиру конкретних XML система за управаљање базама података.

Даља истраживања у овој области могу се одвијати у више праваца, од којих су овде издвојени следећи:

- идентификација нових типова ограничења који до сада нису обухваћени, како из релационог модела података, тако и из других модела података,
- анализа образаца употребе појединих типова ограничења у пракси са становишта учесталости; проверити да ли је решење применљиво на шири скуп проблема или се примењује ретко или једнократно,
- креирање визуелно оријентисаног алата који би служио за дефинисање ограничења,
- интеграција генератора кода у репрезентативни алат за развој информационих система заснован на моделима, као што је, на пример, IIS*Case [Luk10],

- анализа и реализација добијених резултата у другим, актуелним моделима података; обратити посебну пажњу на NoSQL системе за управљање подацима, који већ данас постају све чешће употребљавани у различитим доменима примене, и
- могућност обезбеђења трансформација спецификација ограничења из једног модела података у други модел података; потребно је развити доменски оријентисан језик за дефиницију мета-модела, као и спецификацију одговарајућих трансформација за превођење шема дефинисаних у једном моделу података у шеме дефинисане у другом моделу података; самим тим би била трансформисана и ограничења која су у те моделе уграђена.

У овој дисертацији анализирано је девет типова ограничења из релационог модела података и дефинисани су одговарајући типови ограничења у XML моделу података. Овај скуп типова ограничења, која се могу јавити у теорији и пракси, још увек није комплетан. У пракси постоје и други типови ограничења, као што је, на пример, селективно ограничење референцијалног интегритета, која се у потпуности могу реализовати методологијом реализованом у овој дисертацији. За реализацију сваког новог ограничења у XML моделу података могу се користити принципи приказани у овој тези.

Постоје разни визуелни алати, који подржавају израду XML Schema спецификација, као што су XML Spy [XMLSpy] и одговарајући *plugin*-ови за Eclipse [Eclipse] окружење. Ограничења приказана у овој дисертацији у постојећим едиторима немају одговарајућу репрезентацију. Ако би се кренуло од формалне спецификације ограничења, приказаних у овој дисертацији, могле би бити специфициране апстрактна и конкретна синтакса доменски оријентисаног језика за задавање ограничења разних типова. Из тога би могла бити извршена спецификација визуелне синтаксе таквог језика. Визуелна синтакса би тада омогућила визуелну дефиницију ограничења. Визуелни едитор заснован на таквој синтакси, који би поред постојећих ограничења, могао да обрађује и ограничења из ове дисертације, значајно би олакшао рад пројектанту, јер би уместо уношења дефиниција ограничења у текстуалном облику, та ограничења представљао графичким симболима.

Тренутни сценарио употребе ограничења у овој дисертацији захтева покретање генератора кода, као самосталне апликације. Резултат рада генератора су датотеке које садрже XQuery функције или тригере. Интеграција генератора кода у програмске алате за рад са XML базама података би ову процедуру аутоматизовала. То значи да би на основу текстуалне или визуелне спецификације, аутоматски био генерисан код за проверу ограничења. Ако би типови ограничења, описани у овој тези били уграђени у XML СУБП, односно постали део XML Schema-е, тада би било довољно само дефинисати ограничења у развојним алатима, а њихову проверу би у потпуности преузео XML СУБП, што је увек један важан циљ у развоју модела података и система за управљање базама података који их подржавају.

7 Литература

[Ale13] Aleksić, S., Ristić, S., Luković, I., Čeliković, M., A Design Specification and a Server Implementation of the Inverse Referential Integrity Constraint, ComSIS, Vol. 10, No. 1, January 2013. pp. 283-320

[Are02] Arenas, M., et al., On Verifying Consistency of XML Specifications, ACM PODS, USA, 2002, pp. 259-270

[Are02a] Arenas, M., Fan, W., Libkin, L. What's Hard about XML Schema Constraints?, Database and Expert Systems Applications, LNCS, Volume 2453, 2002, pp. 269-278

[Are04] Arenas, M., Libkin, L., A Normal Form for XML Documents, ACM Transactions on Database Systems, Vol. 29, No. 1, March 2004, Pages 195-232

[Are06] Arenas, M., Normalization Theory for XML, Sigmod Record,2006, Vol. 35, No.4, pp.57-64.

[Ata07] Atay, M., Chebotko, A., Liu, D., Lu, S., and Fotouhi, F., "Efficient schema-based XML-to-Relational data mapping," Information Systems, vol. 32, 2007, pp. 458-476.

[Bai02] Bailey, J., Papamarkos, G., Poulavassilis, A., Wood, P., An Event-Condition-Action Language for XML, In Proc. of the WWW2002, 2002. pp. 486-495

[Bar99] Baru, C., "XViews: XML Views of Relational Schemas," Proceedings of the 10th International Workshop on Database & Expert Systems Applications, IEEE Computer Society, 1999, pp. 700-705.

[Basex] BaseX, http://basex.org/ (септембар 2014.)

[Ben03] Benedikt, M., et al., Capturing Both Types and Constraints in Data Integration, ACM SIGMOD 2003, USA, pp. 277-288, ACM Press, New York, 2003.

[Bon01] Bonifati, A., Braga, D., Campi, A., Ceri, S., Active XQuery, ACM SIGMOD 2001 May 21-24, Santa Barbara, California, USA

[Bon01a] Bonifati, A., Ceri, S., Paraboschi S., Active Rules for XML: A new paradigm for e-services, VLDB J 10(1):39-47, 2001.

[Bon01b] Bonifati, A., Ceri, S., Paraboschi, S., Pushing reactive services to XML repositories using active rules, WWW, ACM, New York, pp. 633-641, 2001.

[Bon02] Bonifati, A., Braga, D., Campi, A., Ceri, S., Active XQuery, 18th International Conference on Data Engineering, San Jose, CA, USA, 2002., ISDE 2002, pp. 403-412

[Bou10] Bourret, R., XML Database Products, http://www.rpbourret.com/xml/XMLDatabaseProds.htm (септембар 2014.)

[Bun01] Buneman, P., Fan, W., Simeon, J., Weinstein, S., Constraints for Semistructured Data and XML, SIGMOD Record, 2001, pp. 47-54.

[Bun02] Buneman, P., Davidson, S., Fan, W., Hara, C., Tan, W., Keys for XML. Computer Networks, 2002, 39(5), pp. 473-487

[Bun03] Buneman, P., Davidson, S., Fan, W., Hara, C., Tan, W., Reasoning about Keys for XML, Information Systems, 28(8):1037-1063, 2003.

[Cha06] Chaudhuri, N., Glace, J., Wilson, G., HIERARCHICAL VS. RELATIONAL XML SCHEMA DESIGNS, A STUDY FOR THE ENVIRONMENTAL COUNCIL OF STATES, REPORT ECO41T1, http://www.exchangenetwork.net/dev_schema/schemadesigntype.pdf, June 2006.

[Coc99] Cochrane, R., Kulkarni, K. G., Mendonca Mattos, N., Active Database Features in SQL3, Active Rules in Database Systems, Springer-Verlag, 1999. pp. 197-219

[Dat03] Date, C. J., An Introduction to Database Systems, 8th edn. Addison-Wesley, London, 2003.

[Dav07] Davidson, S., Fan, W., Hara, C., Propagating XML Constraints to Relations, J. Comput. System Sci., 73(2007), pp. 316-361.

[Deu99] Deutsch, M. F. Fernandez, D. Florescu, A. Levy, D. Suciu, A Query language for XML, In Eighth International World Wide Web Conference, 1999.

[DOM] Document Object Model (DOM), http://www.w3.org/DOM/ (септембар 2014.)

[DTD] Document Type Definition (DTD), http://www.w3schools.com/dtd/ (септембар 2014.)

[Elm03] Elmasri, R., Navathe, S. B., Fundamentals of Databases Systems, 4th edn. Addison-Wesley, London, 2003.

[Eclipse] Eclipse, http://www.eclipse.org/ (септембар 2014.)

[eXcelon] Updating XML data, eXcelon 1.1, User Guide, Chapter 7, 1998. (септембар 2014.)

[eXist] eXist, http://exist-db.org/exist/apps/homepage/index.html (септембар 2014.)

[Fan00] Fan, W., Simeon, J., Integrity constraints for XML, PODS, 2000, pp.23-34.

[Fan01] Fan, W., Libkin, L., On XML Integrity Constraints in the Presence of DTDs, PODS ’01, Santa Barbara, California, USA, ACM 1-58113-361-8/01/05, pp. 114-125

[Fan03] Fan, W., Simeon, J., Integrity Constraints for XML, Journal of Computer and

System Sciences 66(1), 2003, pp. 254-291

[Fan05] Fan, W., XML Constraints: Specification, Analysis, and Applications, DEXA, 2005, pp.805-809.

[Fer02] Fernández, M., Kadiyska, Y., Suciu, D., Morishima, A., and Tan, W., "SilkRoute: A framework for publishing relational data in XML," ACM Trans. Database Syst., vol. 27, 2002, pp. 438-493.

[Fon05] Fong, J., Cheung, S.K., "Translating relational schema into XML schema definition with data semantic preservation and XSD graph," Information and Software Technology, vol. 47, 2005, pp. 437-462.

[FreeMarker] FreeMarker Java Template Engine, http://freemarker.org/ (септембар 2014.)

[Gri05] Grinev, M., Rekouts, M, Introducing Trigger Support fo XML Database Systems, Proceedings of the Spring Young Researcher's Colloquium on Database and Inforation Systems SYRCoDIS, St. Petersburg, Russia, 2005

http://panda.ispras.ru/~grinev/mypapers/triggers_syrcodis2005.pdf

[Gri06] Grinev, M., Fomichev, A., Kuznetsov, S., Sedna: A Native XML DBMS, SOFSEM 2006, Theory and Practice of Computer Science, Lecture Notes in Computer Science Volume 3831, 2006, pp 272-281

[Hu04] Hu, J., Tao, L., An Extensible Constraint Markup Language: Specification, Modeling, and Processing, XML Conference and Exhibition, USA, 2004.

[Hu05] Hu, J., Tao, L., Extensible constraint markup language, US 20050262115 A1, http://www.google.com/patents/US20050262115 (септембар 2014.)

[Hu06] Hu, J., Tao, L., Visual Modeling of XML Constraints Based on a New Extensible Constraint Markup Language, Engineering Letters, Vol. 13, Issue 4, 2006. p248

[Jac02] Jacinto, M. H., et al., XCSL: Constraint Specification Language, Latin American Conference on Informatics, 2002.

[Java] Java Software, http://www.oracle.com/java/index.html (септембар 2014.)

[Jum08] Jumaa, H., Fayn, J, Rubel, P., XML based mediation for automating the storage of SCP-ECG data into relational databases, IEEE Computers in Cardiology, 2008, pp. 445-448.

[Jum10] Jumaa, H., Fayn, J., Rubel, P., An Automatic Approach to Generate XML Schemas from relational Models, 12th International Conference on Computer Modelling and Simulations, 2010, DOI 10.1109/UKSIM.2010.99, pp. 509-514

[Kri03] Krishnamurthy, R., Kaushik, R., Naughton, J., "XML-to-SQL Query Translation Literature: The State of the Art and Open Problems," Database and XML Technologies, 2003, pp. 1-18.

[Lan07] Anders H. Landberg, J. Wenny Rahayu, Eric Pardede, Extending XML triggers

with path-granularity, WISE, Springer, Berlin, pp. 410-422, 2007.

[Lan14] Anders H. Landberg, J. Wenny Rahayu, Eric Pardede, XTrigger: XML database triger, Computer Science - Research and Development

[Laz05] Lazzaretti, A. T., Mello, R. S., A Domain Integrity Constraint Control for XML Documents, Brazilian Symposium on Databases, Brazil, pp 115-129, 2005.

[Lee01] Lee, D., Mani, M., Chiu, F., Chu, W.W., "Nesting-based Relationalto-XML Schema Translation," Int. Workshop on the Web and Databases (WebDB), 2001, pp. 61-66.

[Lee02] Lee, D., Mani, M., Chiu, F., Chu, W.W., "NeT & CoT: Translating relational schemas to XML schemas using semantic constraints," ACM International Conference on Information and Knowledge Management, 2002, pp. 282-291.

[Lee06] Lee, D., Mani, M., Chu, W.W., "Schema conversion methods between XML and relational models," Information and Software Technology, vol. 48, 2006, 245–252.

[Liu03] Liu, J., Vincent, M., Liu, C., Local XML Functional Dependencies, Proceedings of the 5th ACM international workshop on Web information and data management, New Orleans, LU, USA, pages 23-28, 2003.

[Luk03] Luković I, Ristić S, Mogin P, "On The Formal Specification of Database Schema Constraints", I Serbian – Hungarian Joint Symposium on Intelligent Systems, September 19-20, 2003, Subotica, Serbia and Montenegro, Proceedings, Polytechnical Engineering College, Subotica, Serbia and Montenegro, and Budapest Polytechnic, Budapest, Hungary, pp. 125-136.

[Luk07] Luković I, Mogin P, Pavicević J, Ristić S, "An Approach to Developing Complex Database Schemas Using Form Types", Software: Practice and Experience, John Wiley & Sons Inc, Hoboken, USA, ISSN: 0038-0644, DOI: 10.1002/spe.820, Vol. 37, No. 15, 2007, pp. 1621-1656.

[Luk07a] Luković, I., Realizacija ograničenja šeme relacione baze podataka putem sistema za upravljanje bazom podataka, Slajdovi za nastavu iz predmeta Tehnologije razvoja softvera, Fakultet tehničkih nauka, Novi Sad, 2007.

[Luk10] Luković I, Popović A, Mostić J, Ristić S, A Tool for Modeling Form Type Check Constraints and Complex Functionalities of Business Applications, Computer Science and Information Systems (ComSIS), Consortium of Faculties of Serbia and Montenegro, Belgrade, Serbia and Montenegro, ISSN: 1820-0214, Vol. 7, No. 2, pp. 359-385, 2010.

[Lv07] T. Lv and P. Yan, Schema Conversion from Relation to XML with Semantic Constraints, Proceedings of the 4th Int. Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), IEEE Computer Society, 2007, pp. 619-623.

[Mog96] Mogin, P., Luković, I., Principi baza podataka, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, 1996.

[Mog00] Mogin, P., Luković, I., Govedarica, M., Principi projektovanja baza podataka, Univerzitet u Novom Sadu, Fakultet tehničkih nauka, 2000.

[Mog00a] Mogin, P., Luković, I., Govedarica, M., Extended referential integrity, Novi Sad Journal of Mathematics, Vol. 30, No. 3, 2000, pp. 111-122

[Ogb00] Ogbuji, C., Validating XML with Schematron, http://www.xml.com/lpt/a/2000/11/22/schematron.html, 2000. (септембар 2014.)

[Pav00] Pavlova, E., et al., Constraints for Semistructured Data, Russian Conference on Digital Libraries, Russia, 2000.

[Pro02] Provost, W., Beyond W3C XML Schema, http://www.xml.com/pub/a/2002/04/10/beyondwxs.html, 2002. (септембар 2014.)

[Rek05] Rekouts, M., Incorporating active rules processing into update execution in XML database systems, DEXA workshops, IEEE, New York, pp. 831-836, 2005.

[Rob03] Robertsson, E, An Introduction to Schematron, http://www.xml.com/pub/a/2003/11/12/schematron.html, 2003. (септембар 2014.)

[Rod07] Rodrigues, K. R., dos Santos Mello, R., A Faceted Taxonomy of Semantic Integrity Constraints for the XML Data Model, DEXA 2007, LNCS 4653, 2007, pp. 65-74

[Sch05] Schewe, K-D., Redundancy, dependencies and Normal Forms for XML Databases, Proceedings of the 16th Australian database conference – Volume 39, ACM International Conference Proceedings Series, Vol. 103, 2005, pages 7-16

[Schem] Schematron, http://www.schematron.com/overview.html (септембар 2014.)

[Sedna] Sedna, Native XML Database System, www.sedna.org (септембар 2014.)

[Sha01] Shanmugasundaram, J., Kiernan, J., Shekita, E., Fan, C., Funderburk, J., Querying XML Views of Relational Data, Proceedings of the 27th VLDB Conference, Roma Italy, 2001.

[Sha01a] Shanmugasundaram, J., Shekita, E., Barr, R., Carey, M., Lindsay, B., Pirahesh, H., Reinwald, B., "Efficiently publishing relational data as XML documents," The VLDB Journal, vol. 10, 2001, pp. 133-154

[Sha05] Shao, F., Novak, A., Jayavel Shanmugasundaram, Triggers over XML Views of Relational Data, ICDE 2005. Proceedings, 21st International Conference on Data Engineering, 2005, DOI 10.1109/ICDE.2005.147

[Sha08] Shahriar, M.S., Liu, J., On Defining Referential Integrity for XML, International Symposium on Computer Science and its Applications, 13-15 Oct 2008. ISBN

978-0-7695-3428-2, DOI 10.1109/CSA.2008.36, pp. 86-91

[Sha08a] Shahriar, M.S., Liu, J., Preserving Functional Dependency in XML Data Transformation,, Advances in Databases and Information Systems, 12th East European Conference, ADBIS 2008, Pori, Finland, September 5-9, 2008. Proceedings, pp. 262-278

[Sha09] Shahriar, M.S., Liu, J., Towards a Definition of Referential Integrity Constraints for XML, International Journal of Software Engineering and Its Applications, Vol. 3, No. 1, January 2009. pp. 69-82

[Sil05] Silberschatz, A., et al., Database System Concepts, 5th edn. McGraw-Hill, New York, 2005.

[Tat01] Tatarinov, I., Ives, Z., Halevy, A., Weld, D., Updating XML, Proccedings of the ACM SIGMOD International Conference on Managenent of Data 2001, pp. 413-424

[Vid06] Vidaković, J., Racković, M., Generating content and display of library catalogue cards using XML technology, Software: Practice and Experience Volume 36, Issue 5, April 2006, pp. 513–524

[Vid15] Vidaković, J., Luković, I., Kordić, S., Specification and Implementation of the Inverse Referential Integrity Constraint in XML databases, 7th Balkan Conference in Informatics, Craiova, Romania, September 2-4, 2015. (accepted for presentation)

[Vin04] Vincent, M., Liu, J, Liu, C., Strong Functional Dependencies and their Application to Normal Forms in XML, Journal of ACM Transactions on Database Systems (TODS), Volume 29 Issue 3, September 2004, pp. 445-462

[Vin07] Vincent, M., Liu, J., Mohania, M., On the Equivalence between FDs in XML and FDs in Relations, Journal Acta Informatica, Volume 44 Issue 3, June 2007, pp. 207 - 247

[Wan03] Wang, Q., Wu, H., Xiao, J., Zhou, A., Zhaou, J., Deriving Relation Keys from XML Keys, Proceeding ADC '03 Proceedings of the 14th Australasian database conference - Volume 17, pp.227-235

[XDM] XQuery 1.0 and XPath 2.0 Data Model (XDM) , http://www.w3.org/TR/xpath-datamodel/ W3C Recommendation, 14 December 2010 (септембар 2014.)

[XFront] XML_Dev List Group, Extending XML Schemas, http://www.xfront.com/ExtendingSchemas.html

[XML] Extensible Markup Language (XML), http://www.w3.org/XML/ (септембар 2014.)

[XMLD] XML Data, http://www.w3.org/TR/1998/NOTE-XML-data-0105/ (септембар 2014.)

[XMLSch] W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes, http://www.w3.org/TR/xmlschema11-2/ (септембар 2014.)

[XMLSpy] XMLSpy XML Editor, http://www.altova.com/xmlspy.html (септембар 2014.)

[XPath] http://www.w3.org/TR/xpath/ (септембар 2014.)

[XQuery] http://www.w3.org/TR/2010/REC-xquery-20101214/ (септембар 2014.)

[Yan08] Yang, C., Sun, J., “The exchange from relational schemas to XML schemas based on semantic constraints,” Wuhan University Journal of Natural Sciences, vol. 13, 2008, pp. 485-489.

8 Додаци

8.1 Списак листинга

8.2 Списак слика

8.3 Списак коришћених скраћеница

ACID	Atomicity, Consistency, Isolation, Durability
API	Application Programing Interface
CASE	Computer Aided Software Engineering
CLOB	Character Large Object

DTD	Document Type Definition
ECA	Event-Condition-Action
IIS*Case	Integrated Information Systems CASE Tool
IMS	Inforation Management System
OCL	Object Constraint Language
SQL	Structured Query Language
SUBP	Систем за управљање базама података
UML	Unified Modeling Language
URI	Uniform Resource Identifier
XCML	Extensible Constraint Markup Language
XDC	XML Domain Contraint
XDCL	XML Domain Contraint Language
XFK	XML Foreign Key
XID	XML Inclusion Dependency
XML	eXtensible Markup Language
W3C	World Wide Web Consortium