Факултет инжењерских наука Универзитета у Крагујевцу

Назив студијског програма: Машинско инжењерство

Ниво студија: Основне академске студије

Модул: Информатика у инжењерству

Предмет: Базе података

Број индекса: 110/2014

Милош В. Бошковић

XML (и) базе података

завршни рад

Комисија за преглед и одбрану:

1. **Др Милан Ерић - ментор**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

У оквиру завршног рада са темом "*XML (и) базе података*" кандидат треба да:

Проучи литературу и презентира: концепт примене XML-а у пројектовању база података, везу XML докумената и база података и XML базе података. Рад треба да обухвати уводна разматрања везана за развој и примену XML-а, опис синтаксе и семантике XML-а и опис докумената које подржава XML. На реалном примеру приказати примену XML-а. На крају дати закључна разматрања и списак коришћене литературе.

Препоручена литература:

1. Полишчук Ј.: **Пројектовање информационих система**, Електротехнички факултет, Подгорица, 2007.
2. Rainer K., Turban E.: **Introduction to Information Systems**, John Wiley & Sons, Inc., 2009
3. Лазаревић Б., ……: **Базе података**, ФОН Београд, Београд 2003.
4. Павловић-Лажетић Г.: **Основе релационих база података**, Математички факултет, Београд, 2000.
5. Staken K. :**Introduction to Native XML Databases**, http://www.xml.com/pub/a/2001/10/31/nativexmldb.html
6. Ronald Bourret: **XML and Databases**, http://www.rpbourret.com/xml/XMLAndDatabases.htm

Крагујевац, јун 2017.

Ментор:

Prof. Dr Milan D. Erić

Изјава о самосталној изради рада

Изјављујем да сам овај завршни рад урадио самостално, служећи се стеченим знањем и искуством као и наведеном литературом.

110/2014 Милош В. Бошковић

_____

Резиме рада

Концизно је обрађена задата тема, описани су основни појмови везани за XML и базе података као и њихов историјски развој. Разрађена је сама срж XML-а као језика за означавање докумената, начин функционисања XML-а, синтакса XML-а и семантика језика. У раду је описан концепт база података, модели база података и нормализација модела база података. XML се својом функцијом означавања докумената може применити и на моделирање базе података и преношење садржаја унутар документа у табеле. Примена стечених знања о XML-у и базама података израдом пројектног задатка.

Кључне речи: *XML, базе података, XML базе података, DTD, модели база података.*

Abstract

Theme was described concised, basic terms of XML and database were described and their hitorical development. Core of XML was elaborated as markup language for that defines a set of rules for encoding documents, way of functioning, syntax of XML, language semantics. At labor was described concept of database, models of database and normalization of database. XML as markup language can be applied for modeling database and tranfer of document’s content to tables. Appling of acquired knowledge about XML and database for production of project.

Key words: *XML, database, XML database, DTD, models of database.*

Садржај

Увод

У савременом свету се налази велики број информација и често се спомиње преоптерећење информацијама (eng. information overload) са којима се савремени човек сусреће сваки дан. Велики број података свакодневно настају и нестају, а људи се због прекомерних информација губе у свему томе. Све већи број истраживања указује да квалитет живота зависи од тога како се носимо са прекомерним информацијама са којима се сваки дан сусрећемо. Добра организованост података је од круцијалног значаја, зато је потребно да се подаци ефикасно складиште и да се обезбеди брз и ефикасан приступ подацима. Подаци се могу разликовати према својој структури и у зависности од тога могу се складиштити на различите начине.

Чување велике количине података није могуће без управљања подацима на прави начин. Један од начина складиштења и ефикасног управљања подацима је у оквиру база података, које су добро развијене. Оне су погодне за чување структурних података. Пример података би био телефонски именик код кога је свако поље дефинисано тако да има јасну структуру, на пример име, презиме, адреса и број телефона. Међутим, све чешће се јављају подаци који немају јасно дефинисану структуру или такозвани полуструктурни подаци. Такви подаци садрже поља која нису позната у моменту пројектовања базе, подаци код којих се исте врсте података могу представити на различите начине. Пример полуструктурних података би био телефонски именик, али са значајном количином могућих варијација унитар те структуре. На пример, осим имена, адресе и броја телефона, нека поља могу имати више адреса, већи број бројева телефона или додатни текст. У овом примеру имамо редундансу података и сами подаци унутар базе нису једнозначни, база као таква је лоше пројектована.

XML је на самом почетку најављиван као технологија будућносит али што се саме примене тиче једва да је било трага примени. Међутим XML је изненада нашао неко своје место и примену у технологији. XML је релатвно брзо постао једно од средстава без којих се данас сам веб не може замислити. XML је постао најпожељнија синтакса за нове формате докумената у готово свим рачунарским апликацијама. Употребљава се на Linuxu, Windowsu, Macintoshu и многим другим платформама. Како време пролази веб све више расте, тако и његов садржај постаје већи и потребе за информација постају веће (и сама потреба за бржим добијањем информација). Интернет се све мање посматрамо као свет реклама и података а све више се посматра као одређени простор за озбиљно пословање. Информација на интернету више није статичан садржај већ је углавном жива информација – последица упита над базом, а интернет је постао једна глобална мрежа где су сви рачинари повезани и подсећа на оне локалног типа LAN. Оно што је некада HTTP представљао за развој интернета, данас је XML за модерно пословање путем интернета нарочито у сегменту B2B(Buisness to buisness – типично велики информациони системи).

1.0 Уводна разматрања

1.1 Појам XML

XML је акроним од eXtensible Markup Language, а W3C (World Wide Web Consortium) је прихватио да XML буде стандардан језик за означавање (engl.*markup*) докумената. XML је пре свега технологија. Након тога се дефинише као језик мада је XML мање језик а више конвенција за кодирање.

XML је метајезик за означавање текстуалних докумената. Његова намена је дефинисање опште синтаксе за означавање података, свима разумљивим ознакама (eng. tags). Подаци се смештају у XML докумете у облику знаковних низова (engl. *strings*), између текстуалних ознака које их описују. Основне јединице података и ознака у XML-у називају се елементи.

XML спецификација прецизно дефинише синтаксу које се мора придржавати при писању ознака: како су елементи разграничени ознакама, како ознака изгледа, каква имена елементи могу имати, где се стављају атрибути итд. Ознаке XML докумета се на први поглед чине сличним ознакама HTML докуметима, али се суштински разликују.

XML није ту као земена за HTML. Како се ВЕБ у будућности буде још развијао, настоји се у томе да се HTML користи за форматирање и приказивање а XML за описивање података.

Ознаке у XML документу описују његову структуру. Помоћу њих можете видети који елементи су придружени којим другим елементима. У добро пројектованим XML документу, ознаке описују и семантику документа. Примера ради, ознака може указати на то да је елемент датум или име особе или бар-код. У добро пројектованим XML апликацијама, ознаке ништа не казују о томе како докумет треба приказати. Другим речима, не казују да ли је одређени елемент исписан полуцрно или курзивом или је ставка набрајања на листи. XML је језик за означавање структуре и семантике, а не за означавање начина приказивања.[2]

XML обезбеђује стандардан формат за рачунарске документе, флескибилан је и може да се прилагоди најразличитијим областима као што су ВЕБ локације, електронска размена података, векторска графика, итд.

XML је скуп више сродних технологија, он сам као језик не представља ништа посебно али тек са њему сроднм технологијама даје праве резултате. Сродне технологије су:

- DTD(Document type definition)
- CSS(Cascading Style Sheets)
- XLS(Microsoft Excel file)
- ADO(ActiveX Data Objects)
- Xlinks(XML Linking Language)

- XFragments
- Xpointer
- ...itd.[3]

У неким од поменутих технологија XML се понаша као клијент док је за неке сервер а може да буде и једно и друго истовремено.[3]

Један од највећих проблема када је у питању трансфер информација је његов садржај у логичком смислу те речи. XML служи као конектор за трансфер јер у себи поред информације има и њену позицију у односу на друге информације.[3]

XML има много предности као формат за описивање података у односу на остале формате:

- Синтакса етикета није фиксна. Аутор има слободу да креира етикете за потребе одређене апликације. Семантика етикета није ограничена али је зависна од контекста у ком апликација користи документ;
- Флексибилан је и проширив, односно даје могућност додавања нових информација и допушта постојање типова података у оквиру једног документа;
- Описује податке стављањем акцената на то шта подаци јесу а не како они изгледају;
- Има формат који је читљив за човека па је лакше лоцирати и исправљати грешке. Има дрволику структуру коју је лако пратити;
- Има могућност интернационалог коришћења захваљујући чињеници да користи Unicode кодну шему;
- Независан је од платформе односно од софтвера и хардвера који се користи;
- Постоји велики број готових апликација за процесирање XML-а које се могу користити.[1]

Осим предности XML има и своје слабости као на пример то да манипулација подацима је често спорија у односу на традиционалне формате, а оптимизација је комплекснија захваљујући богатству и великој изражајној моћи упитних језика које користи. XML документ мора бити добро форматиран, односно мора да задовољи веома прецизна и строга правила наметнута XML стандардом.

Синтакса XML-а се сматра једноставном и веома строгом. Правила су лака за учење и коришћење па је само креирање софтвера који чита и манипулише XML-ом врло једноставно. Нека од синтаксних правила XML-а су да сви XML елементи морају имати затворене етикете, етикете су осетљиве на велика слова (eng. case sensitive), сви XML документи морају имати јединствен пар етикета за дефинисање кореног елемента, сви други XML елементи морају бити унутар кореног елемента, сви елементи морају имати под-елементе (уметнуте елементе) који морају бити угнеждени унутар родитељског елемента.[1]

1.2 Развој XML-а

XML је настао као потомак SGL-а(Standardized Generalized Markup Language). Језик од ког је настао SGML седамдесетих година изумили су Charles F. Goldfarb, Ed Mosher и Ray Lorie iz IBM-а, а развијало га је више стотина људи широм света, све док га 1986. ISO није прихватио као стандард 8879. SGML је решавао многе проблеме које решава XML, и то на исти начин на који их решава XML. То је семантички и структуриран језик за означавање текстуалних докумената. SGML је изузетно моћан, па је био прихваћен у америчкој војсци, државним службама, авионској, космичкој индустрији и другим областима којима је требао ефикасан начин руковања техничким документима дугачким десетине хиљада страница.[2]

Године 1996, John Bosak, Tim Bray, C.M. Sperberg-McQueen, James Clark и неколицина других почели су рад на „лаганој" верзији SGML-а, која је задржала највећи део његових функционалности, али изоставила оне које су се у претходних 20 година искуства показале редундантним, тешко остваривим, збуњујућим за крајње кориснике или – просто – некорисним. Резултат је био XML 1.0 објављен фебруара 1998.; он је одмах постигао успех. Пуним срцем су га прихватили многи програмери којима је требао структурирани језик за означавање, али се нису могли натерати да савладају SGML-ову сложеност. XML је почео да се употребљава у најразличитијим областима – од судских докумената до узгајања стоке. [2]

Међутим, XML 1.0 је био тек почетак. Следећи стандард био је Namespaces in XML(Простори имена у XML-у), што је био покушај да сеознаке различитих XML апликација употребљавају у истом документу – и то без сукоба. Следећи је био Extensible Stylesheet Language (XSL), што је XML апликација за примену стилова на XML документе, да би се добио облик који могу приказати читачи веб-а. XSL није једина могућност предтављања XML докумената. Када је изумљен XML, у HTML-у су се већ употребљавали каскадни описи стилова (engl.*cascading style sheets,* CSS), и показало се да они прилично добро одговарају у XML-у.

XLink (Extensible Linking Language) почео је дефинисањем моћних структура за повезивање XML докумената у хипертекстуалну мрежу; у поређењу с њим, HTML-ова ознака A (која служи истој сврси) подсећа нас на реч „анемична". I XLink се поделио на два засебна стандарда: XLink за описивање веза између докумената и XPointer за адресирање појединачних делова XML документа. Тада је примећено да XPointer i XSLT развијају префињене, али некомпатибилне синтаксе за исти посао: идентификовање одреоеених елемената XML докумената. Зато су адресни делови обе спецификације одвојени и обједињени у трећој спецификацији званој XPath. Мало касније издвојио се још један део XLinka, XInclude, као синтакса за прављење сложених докумената обједињавањем појединачних докумената и њихових одломака. [2]

Следећи део слагалице био је једнообразни интерфејс за приступање садржају XML документа из Java, JavaScript или C++ програма. Најједноставнији интерфејс за програмирање апликација (API) само је третирао документ као објекат који садржи друге објекте. Штавише, већ се и у W3C и изван њега, радило на дефинисању таквог објектног

модела докумената (Document Object Model, DOM) за HTML. Није било тешко проширити га тако да обухвати и XML. [2]

Једно од најбећих изненађења током развја XML-а је било то што су га програмери више користили за структуре сличне записима, као што су серијализовани објекти и табеле база података, него за наративне структуре за које се традиционално употребљавао SGML. DTD-ови су веома лепо радили за наративне структуре, али су имали нека ограничења у погледу докумената сличнх записима, а управо су њих програмери правили. Главни проблем је био непостојање типова података и сама чињеница да DTD није XML документ. Више компаније је почело са решавањем свих проблема тако што су почели да раде на језицима за описивање. Језик који је објављен 2001. W3C XML верзија 1.0 се показао као неусех јер је био превише сложен и тежак.

После свих догађаја постало је јасно да XML 1.0, XPath, SAX, DOM i W3CXML Schema Language имају сличне, али различите концептуалне моделе структуре XML документа. На пример XPath i SAX сматрају CDATA одељке обичном синтаксом, а DOM их третира различито од чворова (engl.*nodes*) обичног текста. Зато је формирана група која ће да припрема скуп информација о XML-у (XML Information Set) који је заједнички за све наведене стандарде.

Како XML документа преносе информације преко интернета које су све веће вредности изражена је потреба да се трансакције обезбеде и да се проверавају идентитети његових учесника. Поред постојећих механизама као што су SSL и HTTP, који су уграђени у уграђени у софтвере за пренос, развијени су формати за обезбеђивање самих XML документатата током целог века коришћења а не само током преноса. Тако је настао XML Encryption – стандардна XML синтакса за шифровање садржаја укључујући и поверљиве делове докумената. За проверавање идентитета приступа документима задужен је XML Signature(XML потпис), заједнички IETF i W3C стандард за дигитално потписивање садржаја и сама уградња тих потписа у XML документе. Пошто дигитално потписивање и алгоритми за шифровање раде са низовима бајтова, а не са моделима података, XML Signature и XML Encryption су засновани на стандардном формату за серијализацију, Canonical XML, чија је сврха да уклања небитне разлике између докумената као што су размаци унутар ознака и употреба наводника или полунаводника за раздвајање вредности атрибура.

Током свих промена кроз године, основна спецификација XML 1.0 је остала непреомењена. Додавањем нових функција није мењала старе него је само надограђивала претходну. XML 1.0 је заснован на стандарду Unicode 2.0. Како се сам Unicode развијао и почео да обухвата много писама XML је почео да заостаје. Првенствено зато што је 2004. објављен XML 1.1 који нуди мало тога новог програмерима.

Без обзира на све XML ће имати још много проширења. Чак се велика колекција спецификација баве само технологијама које леже у основи XML-а. Направљено је много и још брже се ради на програмирању XML апликација, међу којима су SOAP, SVG, XHTML, MathML, Atom, XForms, WordprocessingML и хиљаде других. XML је и даље остао као чврст темељ за много других технологија.

1.3 Појам безе података

Под појмом база података подразумева тачно организован скуп података који се односи на сличне појмове или предмете и скуп програмских модула који омогућава приступ тим подацима. Како је већ речено ови модули се зову системи за управљање базом података, скраћено СУБП.

Класични начин организовања података где су се подаци организовали у датотеке, док се организација података у интегрисаној форми назива база података.

Сама организација података се може остварити на више различитих начина по којима се касније применом одговарајућих механизама базе могуће вршити претраживања и налажења потребних података.

Систем за управљање базом података(Data Base Management System - DBMS) је сервер базе података. Он обликује физички приказ базе података у складу са траженом логичком стуктуром. Такође обавља у име клијената све операције над подацима. Он је у стању да садржи разне базе има њихову логичку структуру. Подаци су се у бази логички организовали по неком моделу података. Модел података представља како ће изгледати логичка структура података. Модел чини основу за конципирање, пројектовање и имплементацију базе. Досадашњи DBMS-ови обично су подржавали неки од следећих модела:

Релациони модел се заснива на математичком појму релације. Подаци и везе међу њима се приказују „правоугаоним табелама“.

Мрежни модел база је предочена усменим графом. Чворови су типови записа, а лукови дефинишу везе међу типовима података.

Хијерархијски модел је специјални случај мрежног. База је предочена једним стаблом или скупом стабала. Чворови су типови записа а, хијерахијски однос „надређени подређени“ изражава везе међу типовима записа.

Објектни модел је инспирисан објектно-оријентисаним програмским језицима. База је скуп трајно похрањених објеката који се састоје од својих интерних података и метода за руковање тим подацима.

Рад са базом података се не састоји само од дефинисања структуре базе и писања упита за извештавање и ажурирање, већ постоје и неки послови који се морају обављати периодично или по потреби:

- Потребно је додељивати права приступа појединим корисницима базе, вршити контролу додељених права, а по потреби укидати права корисницима;
- У редовним временским интервалима треба проверавати интегритет података у бази;
- Периодично се морају вршити радње потребне за евентуални опоравак у случају отказа: архивирање базе на одређене меморијске медијуме и активирање дневника успешно завршених трансакција;

- Повремено се морају испитивати перформансе базе података и по потреби вршити подешавања физичких параметара, како би се време одзива смањило на најмању могућу вредност.[4]

Да би се што ефикасније и успешније обављали ови и слични послови, база мора да поседује већи број функција за управљање и администрирање базом података.[4]

Управо за извршење тих функција, по правилу, је потребна посебна особа, обзиром да се ради о осетљивим пословима, које је потребно добро познавати, а то је делом и питање организације после, а не само система за управљање базом података. [4]

Надаље, веома често је потребно добро разрадити концепт трансакције као логичке јединице приступа бази, закључивањем и другим механизмима, који омогућавају коректан и ефикасан вишекорисnички рад. [4]

Под термином податак се подразумева као чињеница о неком предмету или догађају која се може забележити и сачувати на рачунару. Под податком се подразумева чињеница као таква тј. Каква јесте. Податак сам по себи нема значење, тек када се интерпретира неком врстом система за обраду података оприма значење и постаје информација.[5]

Термини информација и податак су уско повезани и често се користе као синоними. Међутим, корисно је разликовати термине податак и информација. Информацију дефинише као податак који је био обрађен на такав начин да се знање особе која користи податак повећало. Други начин да се из података добију информације је да се подаци сумирају или на неки други начин обраде и презентују.[5]

Основне карактеристике информације као ресурса:

- Неисцрпна као ресурс;
- Трошењем се не уништава њен садржај;
- Подноси мноштво конзумената истовремено;
- Трошењем (употребом) повећава своју вредност;
- Ограничења (људска способност)[6]

Разликовање података, информација, знања и мудрости кључно је за успешно управљање знањем. Флеминг (1996) наводи следеће тезе:

- Скуп података није информација;
- Скуп информација није знање;
- Скуп знања није мудрост;
- Скуп мудрости није истина.[6]

1.4 Развој база података

Базе података су се развиле као логичан наставак датотека – file – податакакоје су веома дуго коришћене у рачунарству, а и дан данас се користе као основне јединице у које се смештају подаци разних типова – текстуални, табеле, слике итд. Прве датотеке података су биле секвенцијалне – подаци су смештани у низу један за другим, а и претраживање података се одвијало на исти начин.[4]

У самим почецима примене рачунара подаци су били смештени на картице – бушене картице, а касније на магнетне касете и магнетне траке. Као што је већ познато оба медија су секвенцијалног типа и да би нашли податак који нам је потребан, а који се налазио на другом крају – страни траке или касете, требало је премотати целу траку односно касету.

Након тога је дошло до појаве уређаја као што је дискета – floppi disc и дискова – hard disc, CD који су омогућавали директан приступ подацима. Са појавом нових могућностима коришћене су тзв. индекс-секвенцијалне и индексне датотеке код којих је променом индекса и одговарајућих кључева било могуће брже претраживање датотека и налажење потребних података.

На сам развој база података је пресудан утицај имао развој рачунара, рачунарских мрежа, као и клијент/сервер обраде. Истраживање, пројектовање и употреба база података су показали низ својих добрих страна као што су смањени трошкови одржавања; смањена потреба за мрежним ресурсима; побољшан интегритет података; доношење исправних одлука на основу објективних инфрмација, бржа реакција на тржишту, итд.[5]

Традиционални приступ развоју програмских система за чување и обраду података, до појаве база података крајем шездесетих година, био је у суштини ad hoc: полазећи од дефиниције апликације, креирала би се датотека или скуп датотека података потребних за дефинисану апликацију, и писали би се програми који обрађују податке тих датотека у складу са апликацијом. Сва проширења дефинисане апликације реализовала би се додавањем нових датотека и писањем додатних програма. Овакав приступ је брзо показао све своје недостатке:[7]

- Понављањем истих података уз различите апликације. На пример, у једном библиотечком окружењу, апликације за обраду књига, реверса и читалаца могле би да користе одговарајуће датотеке – КЊИГЕ, РЕВЕРСИ, ЧИТАОЦИ, редом. при том би, с обзиром на потребе ових апликација, датотека РЕВЕРСИ садржала све податке о књизи и читаоцу који су већ садржани и у датотеци КЊИГЕ односно ЧИТАОЦИ.

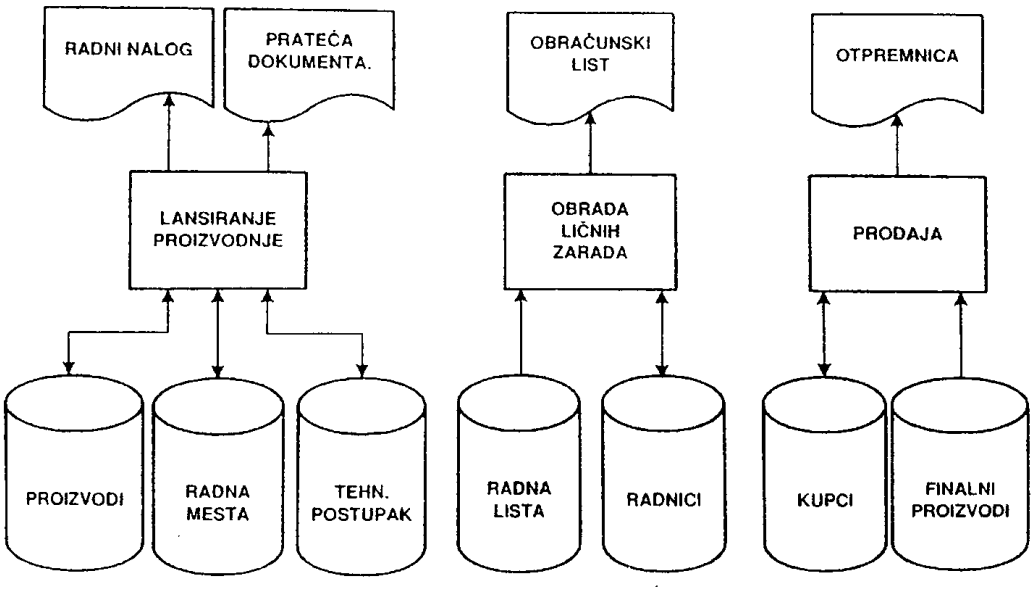
- Неконзистентност података. Због понављања истих података у разним датотекама, може се догодити да се промена вредности примерцима истог податка не изведе доследно, тј. да се једном примерку истог податка вредност промени (намерно или ненамерно), а да се то не учини са осталим примерцима тог податка. На пример, адреса истог читаоца може да буде различито записана у датотекама ЧИТАОЦИ и РЕВЕРСИ, што ће произвести неконзистентност (некоректност) података.

- Програми за обраду података зависе од начина структуирања података. Тако ће ти програми бити различити у случајевима различите организације датотека (секвенцијална, индекс-секвенцијална, директна, индексирана, итд). Зато промена структуре података захтева промену програма.

- Обрада података је скупа, с обзиром на неконзистентност и зависност програма од организације података.

- Коришћење истих података од стране већег броја корисника је отежано. На пример, истовремени покушај два или више корисника да промене садржај датотеке РЕВЕРСИ завршиће се, у многим случајевима, памћењем промена само оног корисника који је последњи завршио рад са датотеком. Остали корисници ни на који начин неће бити обавештени о томе да њихове промене нису упамћене. У другим случајевима вишекорисничког приступа подацима смештеним у систем датотека, када се приступ врши истој копији датотеке, могући су и тежи примери нарушавања коректности података.

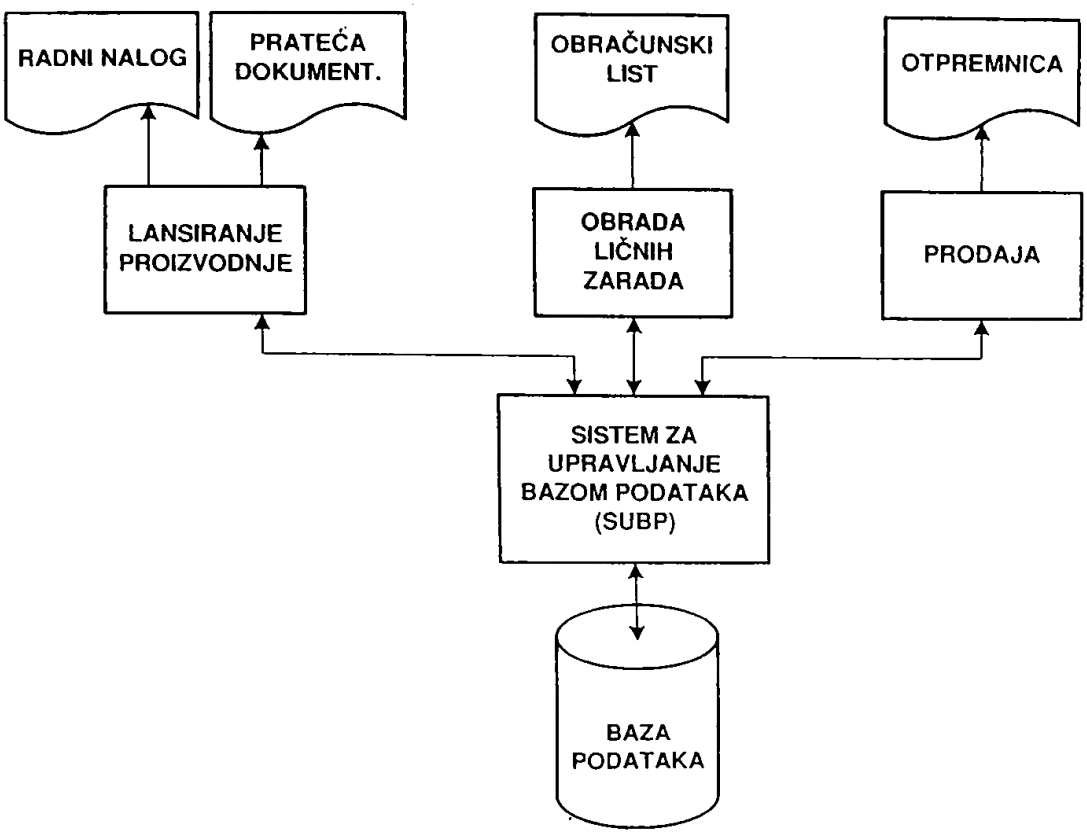
- Неадекватна реализација опоравка од пада система. У случају пада система (нпр. нестанка електричног напајања), активни послови немају могућност поништавања својих делимичних извршења (ако су она део јединствене логичке целине), а често, по успостављању система, ни евиденцију о свом делимичном извршењу. На пример, ако би свим читаоцима требало продужити (на реверсу) рок задужења за по 10 дана, и ако би усред извршавања тог посла дошло до пада система, по успостављању рада не би постојала информација о томе који су реверси обрађени а који још нису.[7](слика:1.1)



Слика 1.1 – Класична обрада података

Систем за управљање базом података данас представља сложен софтверски систем који треба да омогући:

- Складиштење података са минимумо редудансе;
- Поузданост података и при могућим хардверским и софтверским отказима;
- Поуздано паралелно коришћење заједничких података од стране више овлашћених корисника
- Логичку ифизичку независност програма од података. Без обзира што се подаци физички памте, по правилу, само једном, у јединственој физичкој организацији, сваки корисник добија своју сопствену логичку слику података каква њему највише одговара;
- Једноставно комуницирање са базом података преко језика блиских кориснику, тзв. „упитних језика“, како би се се непрофесионални корисници непосредно укључили у развој информационог система, а професионалним програмерима значајно повећала продуктивност.(слика:1.2)



Слика 1.2 – Систем за управљање базом података

2.0 XML

2.1 Како ради XML

За функционалност коју пружа XML неопходан је парсер (рашчлањивање, форматирање). У Internet Exploreru 5.0 и каснијим верзијама постоји уграђен парсер за форматирање из нпр. Visual Basica потребно је поставити референцу на XML формат. Када је XML у питању форматирање значи рашчишћавање текстуалног фајла и прављење структура која рекурзивно пуни елементима XML стабла. То значи да парсер изводи следеће операције:

- Чита предпроцесорски део документа (део на почетку документа између знакова ?) да би дошао од информације које се односе на документ а нису део самог XML стабла. На пример: `<?xml version="1.0" encoding="windows-1252"?>`
- Затим ишчитава први таг у XML структури и записује његово име – ово је top level или startni tag;
- Затим се записује име елемента;
- Затим се ишчитавају остали елементи редом да би се одредило која својства има дати елемент структуре и затим се та својства уписују – ако је у питању елемент уписује се његова вредност или у форми уређених парова атрибути = вредност ако је у питању атрибут;
- Ако следећи таг након првог наредног није ознака за затварање ишчитава се следећи таг и он се дефинише као дете тренутног елемента. Онда се парсер враћа на претходни корак. Ако је нађени таг ознака за затварање онда је елемент дефинисан;
- Овај процес се понавља док се не обради читав документ.[3]

Приликом именовања елемената XML докумената поштују се следећа правила:

- Имена смеју да садрже слова, бројеве и друге карактере,
- Имена не смеју да почињу бројем или интерпукцијским знаком,
- Имена не смеју почињати словима xml или XML или Xml,
- Имена не могу имати празан простор у себи. [3]

Нека општа препорука би била:

Имена треба да буду описујућа и што је могуће краћа и јасна да олакшавају руковање, пример би био:

`<prezime>, <adresa_stanovanja>`

XML атрибути су ту у документима да описују елементе у отварајућем тагу као и код HTML- а. Користе се за додатне информације о елементу. Пример:

```
<IMG SRC="slika.gif">
```

Вредности атрибута се увек стављају унутар знакова навода. Да ли ће се користити једноструки или двоструки знаци навода потпуно је свеједно:

```
<ime="Stevo">
```

или

```
<ime='Stevo'>
```

Али је некада потребно користити једноструке знаке навода као у:

```
<ime='Slobodan "Stevo" Đorđević'>
```

Подаци се могу складиштити или у елементима или у атрибутима. Елементи имају следећу форму:

```
<ime>Stevo</ime>
```

а атрибути у форми:

```
<nesto ime="Stevo">
```

Два начина овога могу бити:

```
<komintent tip="nabavljač">
<ime>Pera</ime>
<prezime>Perić</prezime>
</komintent>
```

или

```
<komintent>nabavljač</komintent>
<ime>Pera</ime>
<prezime>Perić</prezime>
```

У првом примеру тип је атрибут а у другом примеру тип је елемент. Оба типа дају исте информације. Не постоје правила када треба користити атрибуте а када елементе. Поједина начела дају препоруке да се елементи користе када је у питању нешто што је само по себи целокупна информација а не неки њен део.[3]

2.2 Синтакса XML-а

Синтаксна правила XML-а су веома једноставна и стриктна. Лако се уче и примењују. Због тога је креирање апликација које читају и манипулишу XML-ом релативно једноставно.[3] Код је:

```
<?xml version="1.0" ?>
<note>
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</ subject>
<body>Puno pozdrava iz Beograda</body>
</note>
```

Прва линија XML документа – XML декларација – одређује XML верзију документа. У овом реду се дефинише да претраживач форматира документ XML парсером односно да документ третира као XML фајл а не као HTML фајл. Без ове линије добили бисмо поруку о грешци. Ова линија нема затварајући еквивалент јер није део XML документа него је његова декларација. Затим следи основни таг који документ форматира као поруку (<note>). Могућ је само један основни таг иначе опет добијамо поруку о грешци. Следеће четири линије описују четири подчлана основног члана (to, from, subject, i body). Последња линија затвара основни таг (</note>).

- Сви XML елементи морају бити затворени

  У XML-у, изостављање завршног тага води у грешку. Док код HTML-а не би било проблема. Исправно затварање завршног тага у XML-у би изгледало: [3]

  ```
  <p>ovo je paragraf</p><p>ovo je drugi paragraf</p>
  ```

- XML тагови разликују мала и велика слова

  За разлику од HTML-а, XML тагови су case sensitive.

  У XML-у, таг <Poruka> није исто као и таг <poruka>. Зато се мора водити рачуна да се отварајући и затварајући тагови буду идентични, како по називу тако и по употребљивим карактерима: [3]

  ```
  <Poruka>Ovo je neispravno</poruka>
  <poruka>Ovo je ispravno</poruka>
  ```

- Сви XML елементи морају бити прописно угнеждени.

  Неисправно угнеждени елементи немају смисла у XML-у. Док се у HTML-у елементи могу преклапати у XML-у то никако није случај.[3]

HTML исправно

```
<b><i>Ovo je tekst</b></i>
```

XML исправно

```
<b><i>Ovo je tekst</i></b>
```

- Сви XML документи морају имати основни (top level) или стартни таг.

  Први таг у XML документу је основни таг. Сви XML документи морају да имају један пар тагова који дефинише основни таг. Сви остали елементи су угнеждени у основни таг. Гнеждање у дубини је неограничено. Значи елемент може имати неограничен број елемената-деце. Однос који влада је такозвани родитељ дете однос. [3]

```
<note>
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</ subject>
<body>Puno pozdrava iz Beograda</body>
</note>
```

Овде је случај да су осовни тагови <note> и </note> док су његови подчланови парови:

```
<to> i </to>
<from> i </from>
<subject> i </ subject>
<body> i </body>
```

- Вредности атрибута морају бити под знацима навода

  У XML-у се вредности атрибута морају уоквирити знацима навода. XML елементи могу имати атрибуте и форме и форме име=вредност парова (као и у HTML-у). [3]

```
<?xml version="1.0"?>
<note date="10/06/2001">
<note>
<to>Pera</to>
<from>Mika</from>
<subject>pozdrav</ subject>
<body>Puno pozdrava iz Beograda</body>
</note>
```

- У XML-у је сачуван празан простор

  Коришћењем XML-а празан простор је приказан у парсираном документу. [3]

```
<body>Puno            pozdrava iz Dubaia</body>
```

Па ће парсиран бити:

Puno pozdrava iz Dubaia

док са HTML-ом ово није случај

- У XML-у, CR / LF карактери се претварају у LF карактер.
  У XML-у, нов ред у тексту је увек сачуван као LF (line feed). У Windows апликацијама нов ред је пар CR (carriage return) и LF карактера. Код UNIX система карактер за нов ред је LF мада неке апликације користе и само CR. Ова резлика међу оперативним системима често за последицу има да се подаци враћају у облику стрима (engl. toka) а не у жељеном формату. [3]

- XML није нешто специјално али има своје мале тајне
  XML је заправо само текст дизајниран тако да га чита машина односно софтвер а не човек. Софтвер који подржава чисти текст може да обрађује XML. На пример у Notepadu се може обрађивати XML документ. XML може да садржи не-енглеске карактере (č,ć,ž,đ...) међутим тада је потребно документ сачувати у Unicode формату што није могуће на старијим верзијама Windows-a. Стога је у саму декларацију XML фајла уведен и атрибут дешифровање (eng. encoding) што заправо говори browseru коју кодну страну да користи. [3]

  ```
  <?xml version="1.0" encoding="windows-1252"?>
  ```

  Међути овде је потребно обратити пажњу. Фајлови сачувани као Unicode не могу имати и encoding атрибут иначе се појављује грешка у browseru. [3]

2.3 Дефинисање типа документа

Дефинисање типа података (eng. Document Type Definitions — DTD) представља оригиналан начин за спецификовање структуре XML докумената. Он има различиту структуру од XML-а и користи се за спецификацију допуштених градивних блокова и редослед елемената у XML документу. Једноставни градивни блокови XML докумената са тачке DTD-а су:

- Елементи;
- Етикете - главни градивни елементи XML документа(тагови)
- Атрибути – наводе се у оквиру етикета;
- Ентитети – промењиве које се користе да дефинишу општи текст. Предефинисани ентитети у XML-у су на пример: <(<), > (>), & (&), " (”);
- PCDATA (Parsed Character DATA) – текст између отворене и затворене етикете XML елемента који ће се анализирати помоћу парсера;
- CDATA (Character DATA) – текст који се неће анализирати помоћу парсера и етикете унутар текста не означавају обележавање. [1]

DTD може да буде дефинисан унутар структуре XML документа а може да буде декларисан и као спољашња референца. На пример, нека је дат следећи фрагмент XML докумена:

```
<student studbr="st0001">
    <ime>Igor Jovanovic</ime>
    <starost>21</starost>
</student>
```

DTD који описује структуру овог дела XML документа је:

```
<!ELEMENT student (ime, starost)>
<!ATTLIST student studbr CDATA>
<!ELEMENT ime (#PCDATA)>
<!ELEMENT starost (#PCDATA)>
```

Овим DTD-ом је дефинисано да елемент студент има два елемента који су његова деца, име и старост, садржи карактерске податке и атрибут студбр који такође садржи податке представљене карактерима.[1]

3.0 Базе података

3.1 ЕР модел

Како би се обликовала шема за базу података која је усклађена са правилима релационог модела потребно је прво направити ЕР шему. Директно прављење релационе шеме је тежи начин, прављењем ЕР шеме се знатно олакшава посао пројектовања базе података. Обликује се мање прецизан концепт шеме, која представља апстракцију реалног света. ЕР шема се аутоматски преводи у релациону шему.

Основни концепт ЕР модела је узет као општа дефиниција система. У њему се цео систем посматра као скуп три категорије:

- Ентитета: објекти или догађаји;
- Веза: односи међу ентитетима;
- Атрибута: особине ентитета и веза.

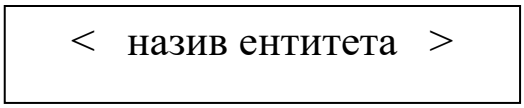
ЕР модел је једноставан начин моделирања базе података где су његове предности:

- Једноставан,
- Прегледан,
- Лак за примену и комуникације,
- Има добру семантику,
- Не зависи ни од једног конкретног система за управљање базом података,
- Може се релативно лако превести у било који модел базе података (хијерархијски, мрежни и релациони)

3.1.1 Ентитети и атрибути

Ентитет је објекат модела који је једнозначно одређен. Ентитет је нешто у шта желимо спремити податке, нешто што је у стању постојати или непостојати, те се може идентификовати. Ентитет може да буде објекат или биће(на пример кућа, дрво, ауто) или нека појава.

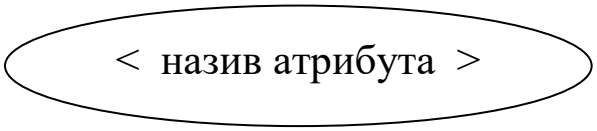
Ентитети се графички приказују у правоугаоном оквиру у који се уписује назив ентитета у једнини.(слика:3.1)



Слика 3.1 – Графички приказ ентитета

Ентитети поседују особине (свиојства, обележја). Опис једне особине се састоји од атрибута (пример: висина, тежина, старост...) која је једнозначно одређена врста особине и вредност атрибута(пример: 196см, 90кг, 22 године...).[6]

Атрибути се графички приказују у елипсастом оквиру у који се уписује назив атрибута.(слика:3.2)



Слика 3.2 – Графички приказ атрибута

Кључ је атрибут или комбинација атрибута који једнозначно одређују ентитет из скупа. Обично је то један(прост елементаран кључ) или комбинација два или више атрибута (сложен кључ). [6]

У једном ентитету може постојати више комбинација атрибута који задовољава дефиницију кључа (пример: за једно возило могућиатрибути-кандидати су: регистарски број, број саобраћајне, број мотора, број шасије.). Један од кандидата који се изабере да практично служи за једнозначну идентификацију ентитета се назива примарни кључ(primary key), а остали, неизабрани кандидати се тада називају алтернативни кључеви.[6]

Примарни кључ се разликује од осталих атрибута по томе што је подвучен.(слика:3.3)

Слика 3.3 – Графички приказ примарног кључа

3.1.2 Везе

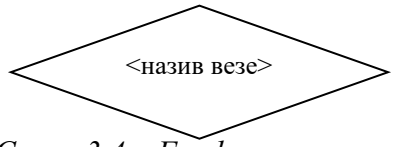
Везе се успостављају између два или више типова ентитета. Заправо је реч о именованој бинарној или к-нарној релацији између примерака ентитета заданих типова. За сада ћемо се ограничити на везе између тачно два типа ентитета. Функционалност везе може бити:

- Један-напрема-један (1 : 1). Један примерак првог типа ентитета може бити у вези с највише једним примерком другог типа ентитета, те такође један примерак другог типа може бити у вези с највише једним примерком првог типа. На пример веза СТУДИРА између типова ентитета СТУДЕНТ и ФАКУЛТЕТ.

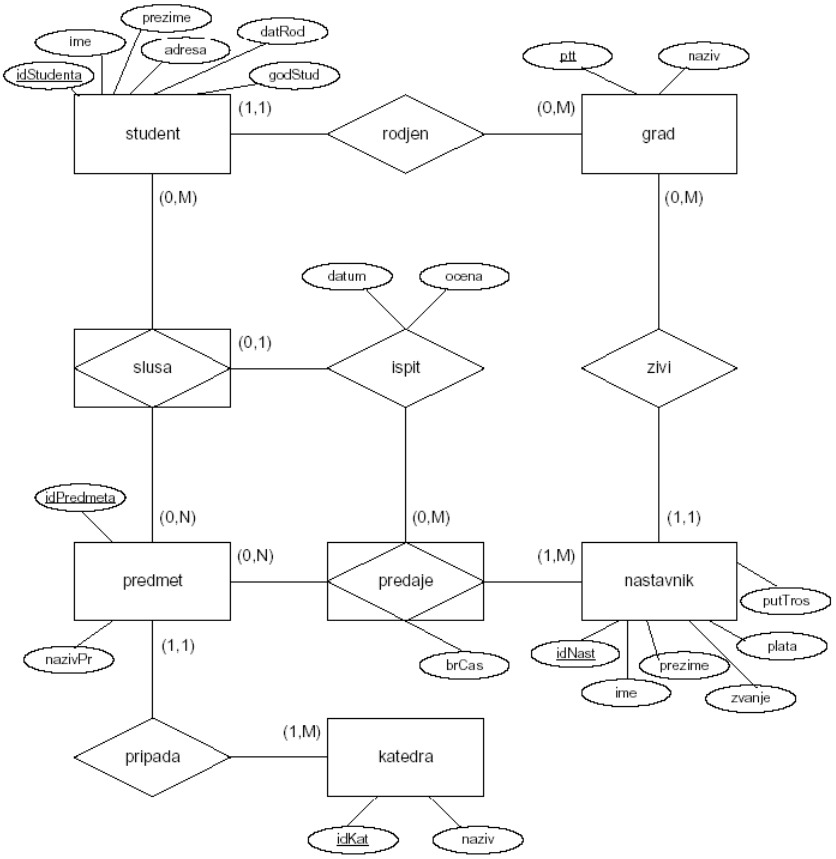
- Један-напрама-много (1 : Н). Један примерак првог типа ентитета може бити у вези с 0, 1 или више примерака другог типа ентитета, но један примерак другог типа може бити у вези с највише једним примерког првог типа. На примјер веза ПРЕДАЈЕ између типова ентитета НАСТАВНИК и СТУДЕНТИ .

- Много-напрама-много (М : Н). Један примерак првог типа ентитета може бити у вези с 0, 1 или више примерака другог типа ентитета, те такође један примерак другог типа може бити у вези с 0, 1 или више примерака првог типа. На пример веза УПИСАО између типова ентитета СТУДЕНТ и КОЛЕГЕ .

Везе се графички приказују ромбом у коме се уписује назив везе(слика:3.4).



Слика 3.4 – Графички приказ везе



Слика 3.5 – Пример ЕР модела

3.2 Релациони модел

Релациони модел се појављивао у академским оквирима. Прве релације које су се појављивале биле су споре и неефикасне. Развојем релационог модела дошло се до тога да је постао ефикаснији и бржи у сваком погледу. У релационом моделу база података систем се представља преко скупа релација. Релације се могу представљати као табеле, где су колоне атрибути релације, а врсте н-торке релације.

3.2.1 Релација, атрибут, кључ

Релациони модел базе података се састоји од скупа табела – тзв. релација. Свака релација има име по којем се разликује од осталих релација у истој бази. Једна колона релације обично садржи вредност једног атрибута (за ентитете или везу) – зато што колону поистовећујемо са атрибутом и обрнуто. Сваки атрибут има своје име од кога се разликује од осталих атрибута у релацији. Вредност једног атрибута су подаци истог типа. Дефинисан је скуп дозвољених вредности за атрибут, који се зове домен атрибута.

Вредност атрибута мора бити једнострука и једноставна. Само у одређеним условима се толерише да недостаје вредност атрибута. Један ред релације обично представља један пример ентитета, или бележи везу између два или више премера. Ред се назива н-торка. У једној релацији не смеју да постоје две једнаке н-торке. Број атрибута је степен релације, а број н-торки је кардиналност релације.

Пример у табеларном приказу за релацију АУТО са подацима о аутомобилима. Табела је(слика:3.6):

REG_BROJ	*PROIZVOĐAČ*	*MODEL*	*GODINA*
ZID654	Ford	Fiesta	1997
BXI930	Volkswagen	Golf	1996
COI453	Nissan	Primera	1997
ZXI675	Ford	Escort	1995
RST786	Fiat	Punto	1993
TXI521	Ford	Orion	1995
HCY675	Volkswagen	Jetta	1997

Слика 3.6 – Релација са подацима о аутомобилима

Кључ К релације Р је подскуп атрибута у релацији који има следећа својства:

- Вредност атрибута К који је кључ једнозначно одређује н-торку у релацији. Значи да у релацији не постоје две н-торке које имају исту вредност кључа.
- Ако се избаци неки атрибут из кључа, тада се нарушава прво својсто.

Постоје случајеви да у релацији постоји више кандидата за кључ. Тада се само један од њих може прогласити примарним кључем. Атрибути који су делови примарног кључа називају се примарни атрибути. Вредност примарног кључа не сме остати не уписан ни у једној н-торки.

Грађу релације се описује шемом релације, која се састоји од имена релације и пописа имена атрибута у заградама. Примарни кључ је подвучен и тако се разликује од осталих атрибута. Пример шеме за релацију о аутомобилима би изгледала:

АУТО(РЕГ_БРОЈ, ПРОИЗВОЂАЧ, МОДЕЛ, ГОДИНА)

3.3 Нормализација релационе шеме

Нормализацоја релација је поступак развоја концептуалног модела у коме се свака релација неког релационог модела своди на одговарајућу „нормалну форму“. [6]

Сви упитни језици засновани на релационој алгебри и релационом рачуну захтевају нормализоване релације, односно релације у Првој нормалној форми (1НФ).

Релација Р је у Првој нормалној форми (1НФ) ако су све вредности њених атрибута атомске. Основе операције у одржавању базе података у релационом моделу су додавање нове н-торке у релацију, избацивање н-торки, измена вредноси неког атрибута у релацији итд.[6]

Пример релације која се налази у 1НФ:

Student(BI,ŠPred,Ime,Sem,ŠSmer,NazSmer,ImeRuk,NazPred,Ocena)

Дефинисање Друге, Треће и Boyce-Codd-ове нормалне форме заснивају се на концепту функционалне зависностиатрибута релације, тј:

Дата је релација Р са атрибутом X и Y. Атрибут Y је функционално завистан од атрибута X (или X функционално одређује Y),

R.X -> R.Y,

ако и само ако свакој вредности X одговара једна и само једна вредност Y. [6]

На пример у релацији Студент постоје следеће функционалне зависности:

BI -> Ime

BI -> Sem

BI -> ŠSmer

BI -> ImeRuk

Атрибут Y релације P је потпуно функционално завистан од атрибута X релације P ако је функционално завистан од атрибута X, а није функционално завистан ни од једног правог подскупа атрибута X. [6]

Пример можемо показати на релацији студент:

BI, ŠPred ---> Ocena

BI -/-> Ocena

ŠPred -/-> Ocena

атрибут оцена је потпуно функционално завистан од сложеног атрибута BI,ŠPRED.

Релација P је у Другој нормалној форми (2НФ) ако и само ако је у 1НФ и сви њени некључни атрибути потпуно и функционално зависе од примарног кључа. Свођење на 2НФ врши се декомпозицијом ,тј. у једној пројекцији је примарни кључ и атрибути који зависе од њега,а у другим пројекцијама се реализују оне функционалне зависности које су проузроковале непотпуне функционалне зависности. [6]

Релација STUDENT:

(BI,Ime,Sem,ŠSmer,ŠPred,NazPred,NazSmer,Ocena,ImeRuk)

Своди се на 2НФ следећом декомпозицијом:

Student1(BI,Ime,Sem,ŠSmer,NazSmer,ImeRuk)

Prijava(BI,ŠPred,Ocena)

Predmet(ŠPred,NazPred)

Релација P је у Трећој нормалној форми (3НФ) ако и само ако је у 2НФ и ако сви њени некључни атрибути нетранзитивно функционално зависе од примарног кључа.

Због постојања транзитивне зависности атрибута NazSmer и ImeRuk од примарног кључа BI релација Student није у 3NF.

Свођење на 3НФ се врши декомпозицијом релације у њене пројекције. За наведени пример декомпозиција је:

STUDENT(BI, Ime, Sem, ŠSmer)

SMER(ŠSmer, NazSmer, ImeRuk)

Дефиниције 2НФ и 3НФ нису довољно прецизне, посебно у случајевима када релација има тзв "преклапајуће" кандидате за кључ (два или више сложених кандидата за кључ који имају барем један заједнички атрибут).

Детерминанта релације P је било који атрибут, прост или сложен, одкога неки други атрибут у релацији потпуно функционално зависи.

Релација P је у Boyce-Codd-овој нормалној форми (BCNF) ако исамо ако су све детерминанте у релацији и кандидати за кључ.

Релација P је у четвртој нормалној форми (4НФ) ако и само ако кад год постоји вишезначна функционална зависност, на пример А ->-> Б, тада сви атрибути релације морају такође бити функционално зависни од А.

Дефиниција у основи каже да у релацији у 4НФ све функционалне и вишезначне зависности морају бити функционалне зависности атрибута од кључа. Или, другим речима, релација је у 4НФ ако је у BCNF и ако су све вишезначне зависности функционалне зависност од примарног кључа.

Специфична врста зависности која постоји у новој верзији релације ПРОГРАМ се назива зависност спајања. Кажемо да су у релацији ПРОГРАМ атрибути ПРЕДМЕТ, НАСТАВНИК, КЊИГА везани преко зависности спајања.

У релацији P(X, Y, ..., Z) постоји зависност спајања ако и само ако релација P резултује из природног спајања њених пројекција по X,Y, ..., З, где су X,Y, ..., Z подскупови атрибута релације P.

Релација P је у Петој нормалној форми ако и само ако се свака зависност спајања може приписати кандидату за кључ.

4.0 XML базе података

Када се помиње XML често се поставља питање: „Да ли се XML документ може сматрати базом података?“

Када кажемо база података подразумева се нека колекција података, онда се у том смислу XML може сматрати базом података. Гледајући на овај начин онда се свака датотека може сматрати базом података, јер у себи садржи неку врсту података. Међутим XML има особине које му дају предност у односу на остале документе. Само-описив је (етикете описују структуру и тип података), преносив је (Unicode) и може представљати податке у дрволикој структури или у облику графа.

Велика предност XML-а је у томе што он може да обезбеди много ствари које су карактеристичне за базе података: складиштење докумената (XML документи), шеме (DTD, XML шеме).упитне језике (XQuery, XPath, XQL, XML-QL, QUILT), интерфејс (SAX, DOM, JDOM), итд. Оно што XML-у не иде у прилог је што му недостаје много заједничких ствари са базама а то је: ефикасно складиштење, индески, безбедност, трансакције и интегритет података, приступ од стране више корисника, тригери, упити над више докумената у исто време итд.

Када имамо ситуацију да радимо са малом количином података, кад имамо свега неколико корисника и кад су перформансе не толико захтевне, XML документи се могу сматрати базом података. Пример су INI датотеке (датотеке које се користе за конфигурисање инцијалних подешавања програма). То је датотека која се линеарно чита и пише али само кад се апликација стартује или кад се завршава. Бољи пример би био већ поменути телефонски именик (име, презиме, број телефона, адреса, итд.). Постоје базе података које су јефтине и лаке за коришћење (Access), нема пуно разлога да се за такве сврхе користи XML. Једнина права предност је преносивос података али у већини случајева она није довољна.

У другим ситуацијама и оним бројнијим, када је база података многоструко већа, постоји много корисника, када се захтва интегритет података и захтеви за врхунским перформансама, XML документи не задовољавају критеријуме који су потребни да би се користили као базе података.

4.1 Врсте XML докумената

Сви XML документи, према степену ригидности своје структуре, упадају у две широке категорије: податак-центрични (eng. data-centric) и документ-центрични (eng. document-centric).[1]

4.1.1 Податак-центрични документи

Податак-центрични документи су документи који XML користи за чување и размени података. Они су пројектовани да се користе за обраду од стране рачунара, пре него за људску употребу. Чињеница да се подаци неко време памте и размењују у XML формату, за апликацију која користи те податке није од значаја. Примери таквих података су јеловник у неком ресторану, редослед продаје производа у некој продавници, план летења авиона на аеродрому итд. Ови подаци се карактеришу прилично регуларном структуром, фино-зрнастим подацима (најмање независне јединице података су на нивоу PCDATA, елемената или атрибута) и јасно одређеним редоследом са мало или без мешовитог садржаја. Овакви подаци су најчешће смештени у некој релационој бази података и јавља се потреба за трансфером података из релационе базе у XML документ, из XML документа у базу података или у оба смера.[1] Податак-центрични документ је:

```
<meni datum='5.10.2007'>
      <jelo>
            <ime>Domaca pileca supa</ime>
            <cena>100.00</cena>
            <kalorije>650</kalorije>
      </jelo>
      <jelo>
            <ime>Francuski tost</ime>
            <cena>30.50 din</cena>
            <kalorije>300</kalorije>
      </jelo>
      <jelo>
            <ime>Bakin dorucak</ime>
            <cena>200.00 din</cena>
            <kalorije>350</kalorije>
      </jelo>
</meni>
```

У општем случају, сваки веб сајт који динамички конструише HTML документ попуњавајући шаблон подацима из базе података може бити замењен низом податак-центричних XML докумената и једним или више XSLT (Extensible Stylesheet Language Transformations) документом. XSLT је језик који омогућава трансформацију XML документа у неку другу форму (на пример у HTML или XHTML документ). [1]

4.1.2 Документ-центрични документи

Документ-центрични документи су пројектовани тако да се углавном користе за људску употребу. Примери су књиге, електронска пошта, рекламе и готово сваки ручно направљени XHTML документ. Они се карактеришу мање регуларном или нерегуларном структуром, крупно зрнастим подацима (најмања независна јединица може бити на нивоу елемента са мешовитим садржајем, а може бити и на нивоу целог документа) и доста мешаним садржајем. Редослед елемената врло често није од значаја. [1]

Овакви документи су најчешће ручно писани у XML-у или неком другом формату (RTF, PDF, SGML) а онда конвертовани у XML.[1] Документ-центрични документ је:

```
<Proizvod>
      <Ime>KIRKOLINA caj za mrsavljenje</Ime>
      <Proizvodjac>Kirka-Pharma</ Proizvodjac>
      <Opis>
            <Paragraf>
            Predstavlja mesavinu lekovitog bilja koje kombinovanim
            dejstvom regulisu promet materija u organizmu,
            ubrzavaju sagorevanje masnih naslaga i uticu na
            smanjenje telesne tezine. <i>Krusina, sena, zova</i>
            stimulisu metabolizam, podsticu probavu, smanjuju nadutost.
            <i>Breza, pirevina, rastavic</i> eliminisu nakupljene
            toksicne materije, poboljsavaju cirkulaciju.
            <i>Maticnjak</i> oslobada od stresa koji je cesto uzrok
            nekontrolisanog konzumiranja hrane. <i>alfija</i>
            kao izuzetni antiseptik stiti od mogucih infekcija
            i utice na jacanje organizma.
      </Paragraf>
      <Paragraf>
        <b>Moete:</b>
      </Paragraf>
      <List>
            <Item>
              <Link URL="Naruci.html">Naruci caj</Link>
            </Item>
            <Item>
              <Link URL="Kirkolina.htm">Vise o ovom proizvodu</Link>
            </Item>
            <Item>
              <Link URL="Katalog.zip">Katalog nasih proizvoda</Link>
            </Item>
            </List>
            <Paragraf>
              Ovaj caj kosta <b>samo 500 dinara.</b>
            </Paragraf>
            </Opis>
</Proizvod>
```

У пракси није увек јасна разлика између ова два типа докумената. Тако на пример, податак-центрични документ као на пример фактура, може садржати крупно-зрнасте податке, нерегуларно структуиране. С друге стране, документ-центрични документ као на пример корисничка упутства могу садржати фино-зрнасте, регуларно структуиране податке као на пример име аутора и датум ревизије. Строга подела између податак-центричних и документ-центричних докумената није увек могућа, тако да је понекад тешко одредити модел коришћења XML-а. У специфичним случајевима када је потребно користити документа оба типа потребно је направити хибридни модел какав је предложио Dare Obasanjo. [1]

4.2 Пренос података између XML документа и базе

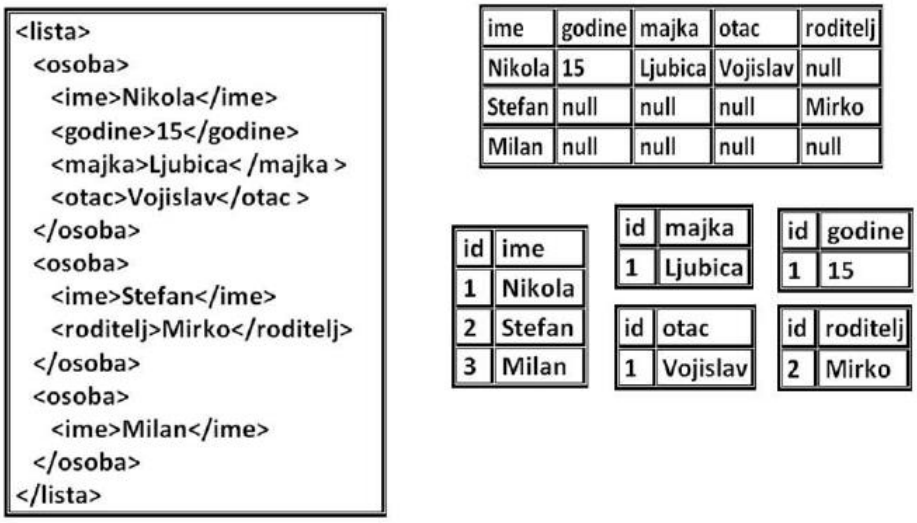
Један од популарнијих начина управљања XML подацима је његово пресликавање у већ постојеће (релационе) базе података. Потребно је извршити пресликавање XML шеме (DTD, XML шема) у шему базе података. За овај пренос података пројектује се посебан софтвер или се користи софтвер који део само релационе базе података када кажемо да је база података XML-проширена.

Приликом преноса података из XML документа у релациону базу података прихватљиво је да се одбаце неке информације о документу (име и DTD) као и његова физичка структура (дефиниција и коришћење ентитета, CDATA (Character data) секције, начин коришћења бинарних података и друго). У неким ситуацијама је чак прихватљиво да се одбаци и део логичке структуре документа (инструкције за процесирање и редослед појављивања елемената са истог нивоа). При преносу у супротном смеру, из релационе базе у XML документ, добијени документ највероватније неће садржати CDATA секције а редослед елемената на истом нивоу зависиће од редоследа у коме их база података даје на излазу. Као последица игнорисања информација о документу и његовој физичкој структури долази до тога да када се XML документ трансформише у релационе податке, а затим се на основу добијених података покуша реконструкција полазног документа, добија документ који се знатно разликује од полазног XML документа. У том случају се каже да систем не омогућава кружно путовање (eng. round trip) XML документа. [1]

Софтвер за пренос података свој рад заснива на пресликавању између докумената и базе односно између елемената XML документа и табела и атрибута релационе базе. Међутим, због флексибилне природе XML -а, овакво пресликавање врло често има као резултат или табеле са великим бројем недостајућих вредности или веома велики број табела (слика 4.1). Тешко је уклопити богату структуру XML -а у круте релационе табеле без дељења документа у веома мале стандардне јединице које се могу приказати као ћелије у табели. Због тога чак и једноставна XML шема производи велики број табела. Структурне информације у шеми базираној на стаблу добијају се спајањем табела у релационој шеми. XML упити се конвертују у SQL упите над релационим табелама и тако се чак и једноставни XML упити транслирају у скупу серију спајања табела одговарајуће релационе базе. [1]

Ronald Bourret је дефинисао две врсте пресликавања шема XML докумената у релационе базе података. То је пресликавање базирано на табелама и објектно-релационо пресликавање.

Пресликавање базирано на табелама се користи код многих производа средњег апликативног слоја за пресликавање XML докумената у релациону базу података. XML документи се моделирају као једна табела или скуп табела.[1]



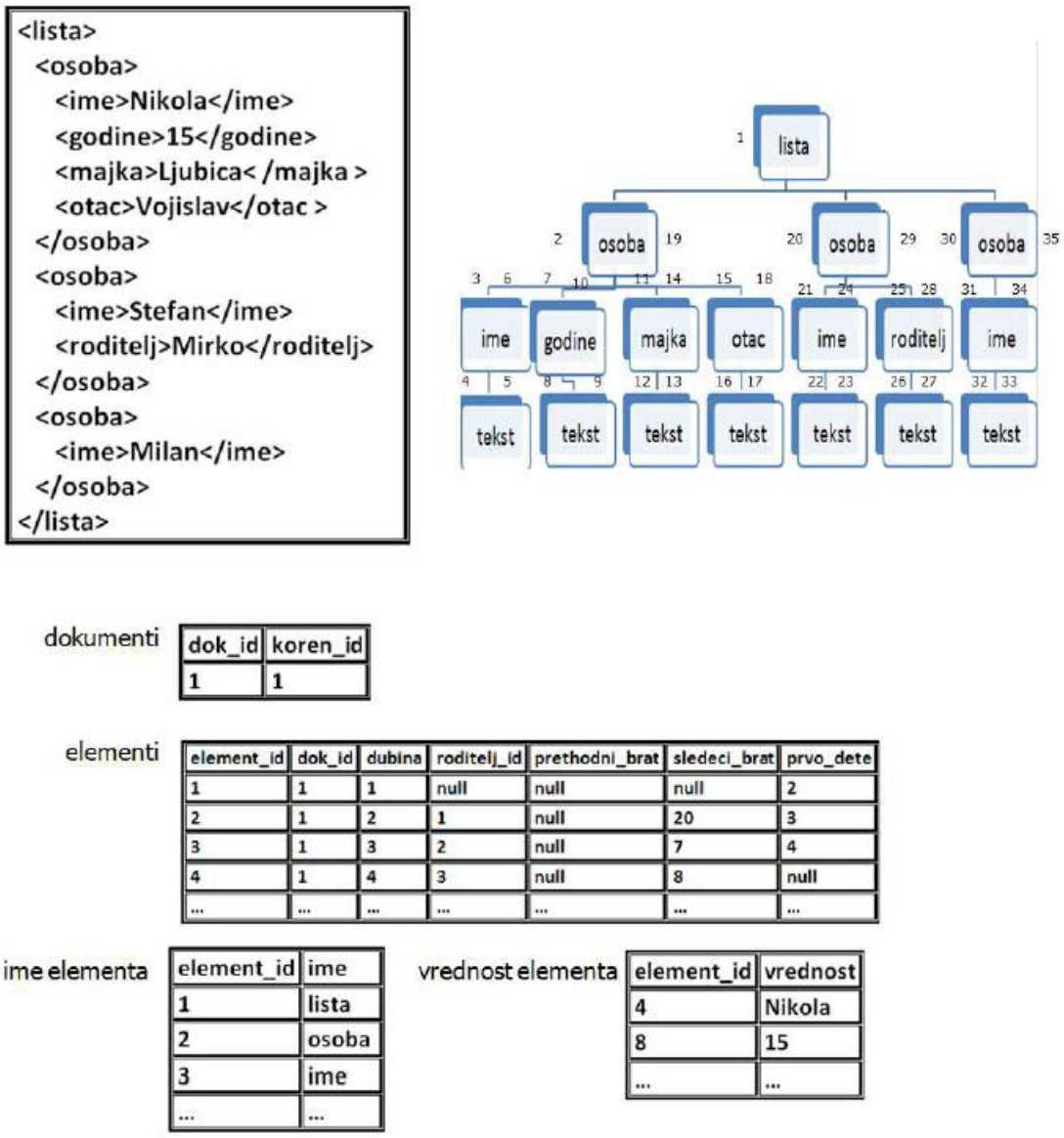
Слика 4.1 -Релациона презентација дела XML документа

Објектно-релационо пресликавање се користи од стране свих XML -проширених релационих база података и неких производа средњег апликативног слоја. Овом методом, XML документ се моделира као стабло објеката који су специфични за податке у том документу. Код овог модела се типови елемената са атрибутима, садржајем или комбинованим садржајем у општем случају моделују као класе а прости елементи, атрибути и сами подаци се моделују као скаларни подаци класа. Модел се затим пресликава на релациону базу података коришћењем стандардних објектно-релационих техника. Пример је дат на слици 4.2. Ову врсту пресликавања треба разликовати од ДОМ (Document Object Model) који моделује сам документ и исти је за све XML документе. Објектно-релационо пресликавање моделује податке у документу и разликује се за скупове XML докумената који се уклапају у задату XML шему.[1]

Применом ове врсте пресликавања, структура документа мора тачно да се уклапа у структуру коју пресликавање очекује. Како ово често није случај, коришћењем XSLT-а (Extensible Stylesheet Language Transformations) документ се прво трансформише у одговарајућу структуру коју пресликавање очекује па се тек онда преноси у базу. Слично, документ који се добије из базе података, пре употребе прво се трансформише у структуру коју очекује апликација.[1]

Приликом преноса података између XML документа и базе података јављају се проблеми, а проблеми који се најчешће јављају су:

- Типови података. XML нема уграђену подршку за типове података и са изузетком CDATA ентитета, XML документ представља обичан текст. То значи да софтвер за пренос података мора да конвертује текст у друге типове података а број формата које софтвер препознаје је ограничен. Једно од начина решавања овог проблема је додавање типова података преко резервисаних атрибута.



Слика 4.2 – Архитектура базирана на моделу

- Бинарни подаци. Бинарни подаци у XML документу се могу чувати на два начина: као CDATA елементи или коришћењем BASE64 кодирања. Оба начина су проблематични по софтвер за пренос података. XML-у не постоји стандард за индикацију да ли неки елемент садржи BASE64 кодиране податке, па се може десити да софтвер не може да препозна да се ради о бинарно кодираним подацима. Мана је и то што нотација везана за BASE64 кодирање и за CDATA елементе најчешће се занемарује приликом преноса података.

- NULL вредност. NULL вредност је подржана од стране XML-а захваљујући концепту опционих атрибута и елемената. При пресликавању саме структуре документа треба водити рачуна да се опциони елементи и атрибути пресликају само на оне колоне у табелама које дозвољавају NULL вредност. Исто правило важи и приликом пресликавању у супротном смеру.

- UNICODE подршка. Са изузимањем појединих контролних карактера, XML може да садржи само UNICODE карактере. Велики број релационих база података нуди веома слабу чак и никакву подршку за UNICODE и обично се захтева да се унапред изврши посебна конфигурација.

- Процесирајуће инструкције и коментари. Они не спадају у податке унутар документа и најчешће се занемарују током процесуирања. Могу да се појаве било где унутар документа и зато се не уклапају ни у једно пресликавање.

- Памћење маркера. У неким случајевима је потребно да се елементи са комплексним садржајем запамте баш онако какви јесу без даље обраде. У XML документима се маркери памте коришћењем референци ентитета. Ово се може користити и код база података је проблем са SQL-ом који не може да распознаје референце ентитета.

4.3 Релационе базе података

Релациона решења предлажу на који начин треба нормализовати XML податке на табеле и колоне при чему свака ћелија садржи атомски текстуални податак. Уопштено гледано сваки XML документ се може нормализовати за смештање у релациону базу података. Овај начин има смисла само код добро структуираних XML докумената. Што је неки документ мање структуиран, има више комплексних елемената и елемената које имају комплексан садржај, тако ће број табела у бази расти тако да ће на крају релациони модел XML документа изгубити практичан смисао.

Пост-релациона решења су настала у покушају да се релационе базе података прилагоде захтевима које је пред њих поставио објектно оријентисани приступ. Та нова решења у великој мери олакшавају управљање XML подацима. Већина релационих DBMS је увела LOB (Large OBjects) типове података који омогућавају смештање великих количина података у појединачним ћелијама. Осим тога произвођачи DMBS-а (а и сам SQL стандард) су додали подршку за ћелије које могу да садрже податке који се понављају као

и могућност претраге целог текста (енг. full-text) односно претраге свих речи садржаних у тексту. Поред овога развијен је велики број додатних алата који омогућавају лакше манипулисање XML подацима. Када се уз помоћ претраге целог текста пронађе податак који је од интереса, уз помоћ ових алата се податак може представити у XML облику и може се манипулисати њиме коришћењем DOM -а (објектни модел документа) или XPath -а (XML Path Language језик за одабир чворова из XML документа). Међутим ни један од ових система не омогућава комплетно кружно путовање (енг. round trip) произвољног документа из базе и у базу.[1]

Изворне XML базе података (енг. Native XML Database — NXD), за разлику од претходних решења која покушавају само да модификују постојеће релационе базе, су базе података пројектоване специјално за управљање XML документима. Оне одржавају природну дрволику структуру ових докумената и карактеришу се великом флексибилношћу. [1]

Изворне XML базе података по дефиницији подразумевају:

- Дефинисање (логичког) модела за XML документе и смештање и добијање докумената у складу са тим моделом. Минимално, модел мора укључити елементе, атрибуте, PCDATA и редослед документа. Примери таквих модела су XPath модел података, XML Infoset (XML Information Set), и модели имплицирани са DOM (Document Object Model) и догађајима у SAX (Simplified API for XML);
- XML документ је основна јединица (логичког) складиштења, као што је то врста у табели код релационих база;
- Нема захтева за постојањем специфичног физичког модела складиштења. На пример, база може бити изграђена на релационој, хијерархијској или објектно оријентисаној бази, или може користити одговарајући формат за смештање као што су индексоване, компресоване датотеке.[1]

Закључак који се може извести јесте да су изворне XML базе специјализоване за смештање XML докумената, смештајући све компоненте XML модела нетакнутим. XML базе података не морају бити самосталне базе и не морају смештати XML податке усвојој изворној форми. Формат у коме се врши складиштење података или физички модел нису од значаја за категоризацију база.

Треба ипак знати да изворне XML базе података немају тенденцију да замене постојеће базе података већ да помогну у самој ефикасности управљања XML документима.Табела 4.3 је приказано поређање ралационих и XML база података:

Relaciona baza	XML baza
Osnovu čini tabela	Osnovu čini kolekcija
Relaciona tabela sadrži slogove (koji predstavljaju vrste u tabeli) definisane po istoj shemi	Kolekcija sadrži XML dokumente istog modela
Relacioni slog predstavlja listu nesortiranih vrednosti	XML dokument ima strukturu drveta sa međusobno povezanim čvorovima
SQL upit vraća nesortiran skup slogova	XQuery upit vraća sortiranu sekvencu čvorova

Табела 4.3 – Поређење релационих и XML база података

Архитектура изворних XML база података се може поделити у две категорије: базиране на тексту и базирана на моделу.

Изворне XML базе базиране на тексту чувају XML податке као текст. Оно што је карактеристично за ове XML базе је изузетно добро индексирање које омогућава да се веома брзо референцира било који XML документ или неки његов део. Базе базиране на тексту због тога постижу изузетне перформансе када се подацима приступа у складу са предефинисаном хијерархијом.[1]

Изворне XML базе базиране на моделу формирају интерни објектни модел на основу XML документа и онда складиште тај модел. Најчешће су изграђене тако да користе неку релациону базу као средство за физичко складиштење података. У том случају перформансе ових база у великој мери зависе од релационих база које раде у позадини. Ако користе неки други систем за складиштење података онда се обично користе физички показивачи између XML докумената чиме се обезбеђују перформансе које су веома сличне перформансама база заснованих на тексту.[1]

Особине изворних XML база података карактеришу различите врсте XML база података, неке особина су: подршка колекцији докумената, трансакција, сигурност, вишекорисnички приступ, прорамски интерфејс, упитни језици итд.

Велики број изворних XML база података има подршку за логгички модел груписања докумената названим „колекције“. Колекција има улогу која је слична табели у релационим базама. На овај начин се обезбеђује да упити које постављамо буду огграничени на одређени скуп података. Суштинска разлика између колекције и табеле је то што не захтевају све изворне XML базе да шема буде придружена колекцији. То значи да се код тих база може сместити било који XML документ у колекцију, без обзира на шему. Упити се могу слати за сва документа у колекцији. Изворне XML базе које подржавају ову особину зову се независне шеме.

Скоро све изворне XML базе података подржавају један или више упитних језика. Најпопуларнији је XPath (са проширењем за упите кроз више докумената) и XQuery, декларативни упитни језик препоручен од стране W3C. [1]

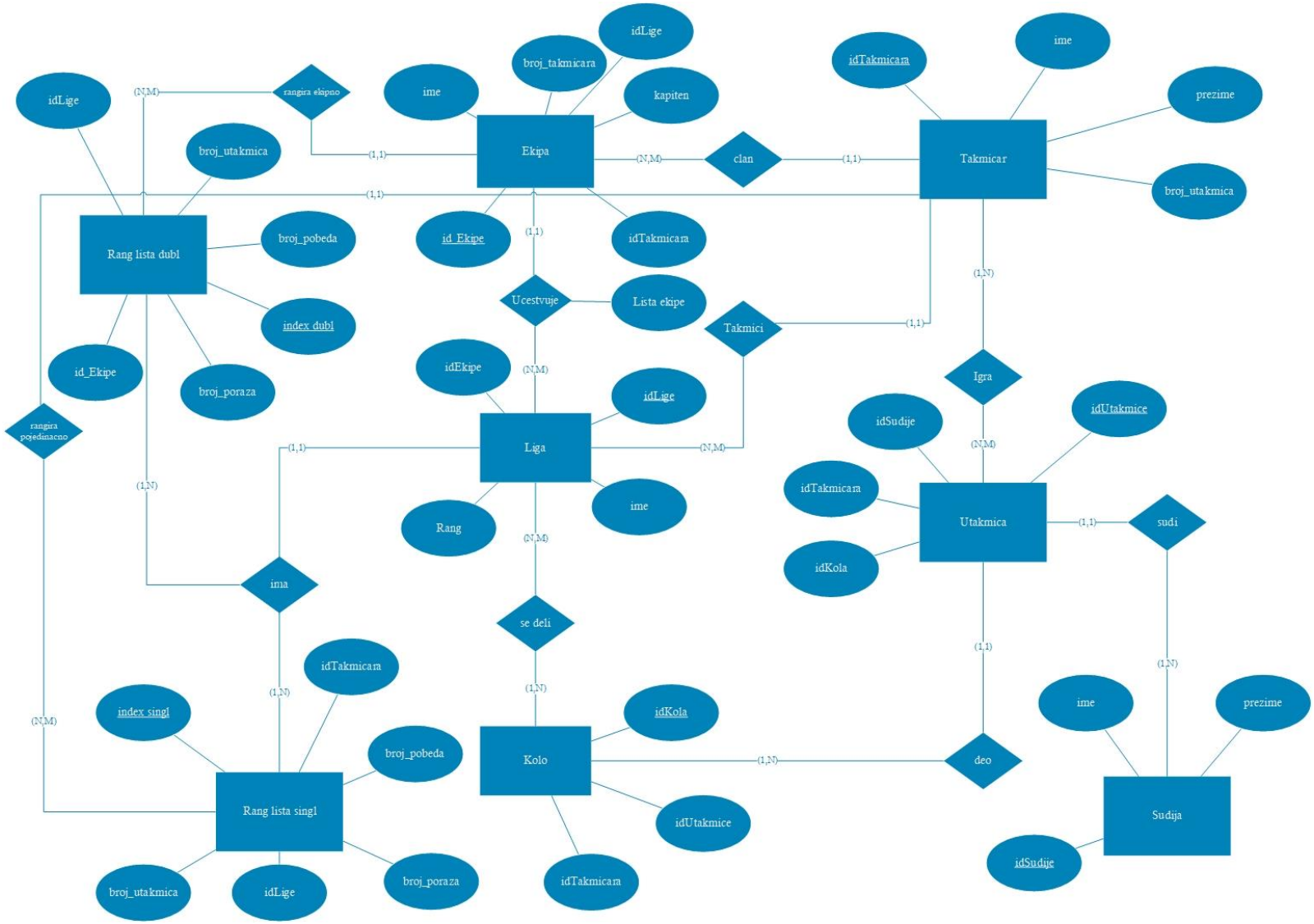
Ажурирање представља велику слабост постојећих изворних XML база података. Постоји велики број стратегија: од једноставног брисања и замене, преко коришћења DOM стабла па до дефинисања посебних језика за модификацију. Многи производи захтевају добијање документа из базе, његово ажурирање коришћењем неког од XML API (XML Application Programming Interface) и онда његово враћање у базу. Неколико производа имају развијене одговарајуће језике који омогућавају ажурирање документа унутар базе. У те сврхе, неке изворне XML базе података подржавају XML: DB XUpdate, језик који је XML -заснован. У оквиру њега се користи XPath за идентификовање неког скупа чворова. Затим се врши спецификација да ли се ти чворови желе убацити у базу, избрисати или се жели убацивање нових чворова пре или после тог идентификованог скупа чворова. [1]

5.0 Пројектни задатак – студијски приказ

У овом пројектном задатку смо представили обраду података стонотениске лиге Сторек. Обрада се заснива на статистичком вођењу појединачних ранг листи, дубл и екипној ранг листи. Сторек се у текућој сезони састоји из три лиге. Свака лига броји одређени број екипа (минимално 10, а максимално 14). Свака екипа се састоји од играча (минимално 3, а максимално 5). Играче је могуће додати у састав све до почетка 4 . кола у првенству. Свака екипа има капитена који је задужен за заказивање утакмица. Капитен је описан именом и презименом и бројем телефона. Сваке седмице се одиграва по једно коло. Распоред одигравања утакмица у првенству се одређује применом Бергерових таблица (користе се и у шаху). Како би се водила ранг листа на правилан начин сваки меч има записник у коме се води евиденција резултата у мечу. Меч се састоји из десет партија, девет појединачних и једне дубл партије. Уколико једна екипа освоји шест или више партија, онда добија 3 бода, а противник 0, док нерешен исход доноси по 1 бод свакој од екипа. Појединична ранг листа се састоји тако сто је сваки играч описан именом и презименом, називом екипе за коју игра и скором победа и пораза, освојених и изгубљених сетова. Уколико два играца имају исти број победа, боље пласиран играч је онај који има више одиграних партија, а у колико је и то једнако онда одлучује међусобни сусрет та два играча. Уколико се нису састали онда се гледа ко има више освојених сетова, уколико је и то једнако онда се гледа ко има мање изгубјених сетова. На крају сезоне се три првопласирана играча у свакој од лига награђују пехарима и медаљама, као и првопласирани дублови у свакој лиги, и три првопласиране екипе у свакој од лига.

Ер модел

За лигу је познато: идентификациони број лиге, име, ранг лиге, идентификациони број екипе. У лиги учествује више такмичара а такмичар може да се такмичи у једној и само једној лиги. Такмичар се карактерише идентификационим бројем такмичара, именом, презименом, бројем утакмица. У свакој лиги учествује више екипа да би се одржала сама лига. Екипа се карактерише са идентифакионим бројем екипе, именм, капитеном, бројем утакмица, идентификационим бројем лиге и идентификационим бројем такмичара, екипа може да учествује у једној и само једној лиги. Такмичар је члан једне и само једне екипе, а екипа има више такмичара. Такмичар игра једну или више утакмица. Утакмца се карактерише са идентификационим бројем утакмице, идентификационим бројем тамичара, идентификационим бројем судије и идентификационим бројем кола, утакмицу игра више такмичара. Утакмицу суди један и само један судија а судије може да суди једну или више утакмица. Судија се карактерише са идентификационим бројем судије, именом и презименом. Свака утакмица је део једног и само једног кола а коло може да има једну или више утакмица. Коло се карактерише са идентификационим бројем кола, идентификационим бројем утакмице и идентификационим бројем такмичара. Лига се дели на више кола а коло може имати једну или више лига. Свака лига има једну и само једну ранг листу сингла. Ранг листа синг има једну или више лига и карактерише се са индеском синга, бројем утакмица, бројем победа, бројем пораза, идентификационим бројем лиге, идентификационим бројем такмичара. Свака лига има једну и само једну ранг листу дубла. Ранг листа дубл има једну или више лига и карактерише се са индеском дубла, бројем утакмица, бројем победа, бројем пораза, идентификационим бројем лиге, идентификационим бројем екипе. Такмичар се рангира појединачно на једној и само једној сингл ранг листи. На сингл ранг лисити се рангира више такмичара. Екипа се рангира екипно на једној и само једној дубл ранг листи. На дубл ранг листи се рангира више екипа. *Слика (5.1)*



Слика 5.1 – ЕР модел лиге стоног тениса

Р модел

Применом правила за превођење ЕР модела у Р модел на следећи начин:

Ентитет лига постаје релациона шема са истим именом, примарни атрибут идентификациони број лиге постаје примарни кључ релације, атрибути ентитета посатју атрибути релације а идентификациони број екипе постаје секундарни кључ.

Ентитет екипа постаје релациона шема са истим именом, примарни атрибут идентификациони број екипе постаје примарни кључ релације, атрибути ентитета посатју атрибути релације а идентификациони број такмичара постаје секундарни кључ.

Ентитет такмичар постаје релациона шема са истим именом, примарни атрибут идентификациони број такмичара постаје примарни кључ релације, атрибути ентитета посатју атрибути релације.

Ентитет коло постаје релациона шема са истим именом, примарни атрибут идентификациони број кола постаје примарни кључ релације, идентификациони број утакмице, идентификациони број такмичара постају секундарни кључеви.

Ентитет утакмица постаје релациона шема са истим именом, примарни атрибут идентификациони број утакмице постаје примарни кључ релације, идентификациони број судије, идентификациони број такмичара, идентификациони број кола постају секундарни кључеви.

Ентитет судија постаје релациона шема са истим именом, примарни атрибут идентификациони број судије постаје примарни кључ релације, атрибути ентитета посатју атрибути релације.

Ентитет ранг листа дубл постаје релациона шема са истим именом, примарни атрибут индекс дубл постаје примарни кључ релације, атрибути ентитета посатју атрибути релације а идентификациони број екипе постаје секундарни кључ.

Ентитет ранг листа сингл постаје релациона шема са истим именом, примарни атрибут индекс сингл постаје примарни кључ релације, атрибути ентитета посатју атрибути релације а идентификациони број такмичара постаје секундарни кључ.

Liga(idLige, ime, rang, idTakmicara, idEkipe)

Ekipa(idEkipe, ime, broj_utakmica, kapiten, idTakmicara)

Takmicar(idTakmicara, ime_takmicara, prezime_takmicara, broj_utakmica)

Kolo(idKola, idUtakmice, idTakmicara)

Utakmica(idUtakmice, idSudije, idTakmicara, idKola)

Sudija(idSudije, ime_sudije, prezime_sudije)

Rang lista dubl(index dubl, broj_utakmica, broj_pobeda, broj_poraza, id_ekipe)

Rang lista singl(index singl, broj_utakmica, broj_pobeda, broj_poraza, idTakmicara)

Међурелациона ограничења:

Liga[idEkipe]⊆Ekipa[idEkipe]

Liga[idTakmicara]⊆Takmicar[idTakmicara]

Utakmica[idKola]⊆Kolo[idKola]

Utakmica[idTakmicara]⊆Takmicar[idTakmicara]

Utakmica[idSudije]⊆Sudija[idSudije]

Kolo[idUtakmice]⊆Utakmice[idUtakmice]

Kolo[idTakmicara]⊆Utakmice[idTakmicara]

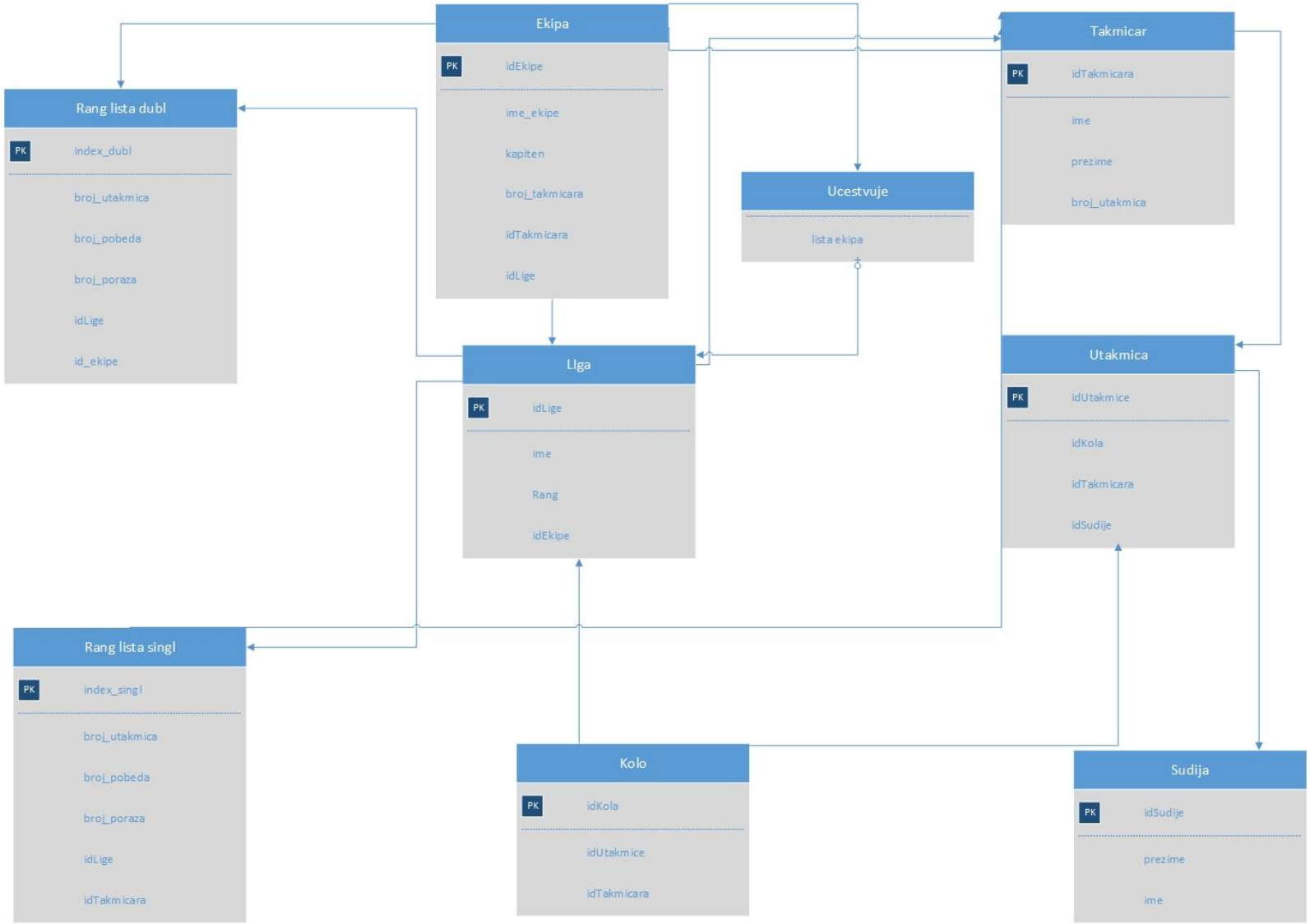
Ekipa[idTakmicara]⊆Takmicar[idTakmicara]

Rang lista dubl[idLige]⊆Liga[idLige]

Rang lista dubl[idEkipe]⊆Ekipa[idEkipe]

Rang lista singl[idLige]⊆Liga[idLige]

Rang lista singl[idTakmicara]⊆Takmicar[ime_Takmicara]



Слика 5.2 – P модел лиге стоног тениса

Креирање базе података применом XML-а

idLige	Ime	Rang	idEkipe
5	Super liga Kragujevca	1	2
6	Prva liga Kragujevca	2	5
7	Druga liga Kragujevca	3	7

```
<lista_liga>
	<liga>
		<idLige>5</idLige>
		<ime>super liga kragujevca</ime>
		<rang>1</rang>
		<idEkipe>2</idEkipe>
	</liga>
	<liga>
		<idLige>6</idLige>
		<ime>prva liga kragujevca</ime>
		<rang>2</rang>
		<idEkipe>5</idEkipe>
	</liga>
	<liga>
		<idLige>7</idLige>
		<ime>druga liga kragujevca</ime>
		<rang>3</rang>
		<idEkipe>7</idEkipe>
	</liga>
</lista_liga>
```

idEkipe	ime_Ekipe	kapiten	broj_utakmica	idTakmicara
2	Falcons	Milos	12	5
5	Cowboys	Nemanja	10	8
7	Hawks	Dejan	9	1

```
<lista_ekipa>
      <ekipa>
            <idEkipe>2</idEkipe>
            <ime_ekipe>Falcons</ime_ekipe>
            <kapiten>Milos</kapiten>
            <broj_utakmica>12</broj_utakmica>
            <idTakmicara>5</idTakmicara>
      </ekipa>
      <ekipa>
            <idEkipe>5</idEkipe>
            <ime_ekipe>Cowboys</ime_ekipe>
            <kapiten>Nemanja</kapiten>
            <broj_utakmica>10</broj_utakmica>
            <idTakmicara>8</idTakmicara>
      </ekipa>
      <ekipa>
            <idEkipe>7</idEkipe>
            <ime_ekipe>Hawks</ime_ekipe>
            <kapiten>Dejan</kapiten>
            <broj_utakmica>9</broj_utakmica>
            <idTakmicara>1</idTakmicara>
      </ekipa>
</lista_ekipa>
```

idTakmicara	ime_takmicara	prezime_takmicara	broj_utakmica
1	Dejan	Tomasevic	11
5	Milos	Teodosic	12
8	Nemanja	Bjelica	10

```
<lista_takmicara>
      <takmicar>
            <idTakmicara>1</idTakmicara>
            <ime_takmicara>dejan</ime_takmicara>
            <prezime_takmicara>tomasevic</prezime_takmicara>
            <broj_utakmica>11</broj_utakmica>
      </takmicar>
      <takmicar>
            <idTakmicara>5</idTakmicara>
            <ime_takmicara>milos</ime_takmicara>
            <prezime_takmicara>teodosic</prezime_takmicara>
            <broj_utakmica>12</broj_utakmica>
      </takmicar>
      <takmicar>
            <idTakmicara>8</idTakmicara>
            <ime_takmicara>nemanja</ime_takmicara>
            <prezime_takmicara>bjelica</prezime_takmicara>
            <broj_utakmica>10</broj_utakmica>
      </takmicar>
</lista_takmicara>
```

Закључак

XML је од самог настанка најављиван као као технологија будућности. Током времена је постао неизоставан елемент функционисања данашње технологије. XML је технологија за означавање докумената. С обзиром да је заснован на UNICODE јер користи његову кодну шему и подржан је од стране више језика. XML је од самог настанка имао широку област примене а пораст његове примене ће се наставити у будућности, јер је XML постао као неизоставан део како самих докумената тако и преноса докумената.

Базе података као супериорни начин за складиштење података настао као бољи начин за складиштење од класичног начина складиштења података. Класичан начин складиштења података у датотеке био је неефикасан и сам процес тражења података је одузимао доста времена. Базе података представљају скуп добро организованих података, које су организоване у табеле ради ефикаснијег складиштења и брже претраге података. Базе података се користе у свим областима IT-а као најбољи начин чувања и додавања података.

XML као технологија за означавање докумената осим података које носи унутрар сваког документа, може да врши и функцију базе података. XML база података има своје предности и мане, XML базе раде са малом количином података, са малим бројем корисника, XML се може сматрати базом података. XML базе података су нашле своју примену у транферу података, што значи да су подаци изведени из базе у XML документ и сам документ постаје база података. Овај начин се показао као ефикаснији из перспективе трошкова конверзије и као лакши начин за складиштење података у XML формат.

Литература

[1] – J. Томашевић: XML базе података у управљању лексичким ресурсима, Магистарски рад, Математички факултет, Београд 2008.

[2] - XML за програмере – Новембар,2017.

[3] – Школа XML-а – Новембар,2017.

[4] – Базе података – Новембар,2017.

[5] – М. Веиновић; Г. Шимић: Увод у базе података, Универзитет Сингидунум, Београд 2010.

[6] – М. Ерић: Презентације базе података, 2017.

[7] – Г. Павловић-Лажетић: Увод у релационе базе података, Математички факултет, Београд 1999.