Претраживање библиографских записа

Retrieving Bibliographic Records

Едиција
Рачунарске науке

Уредници
проф. др Душан Сурла
проф. др Зора Коњовић

Издавач
Група за информационе технологије
Нови Сад, Бул. Краља Петра I 13
gint@eunet.yu

CIP – Каталогизација у публикацији
Библиотека Матице српске, Нови Сад

025.31:004
004.42 BISIS

МИЛОСАВЉЕВИЋ, Бранко
Претраживање библиографских записа / Бранко Милосављевић. – Нови Сад : Група за информационе технологије, 2004 (Нови Сад : АБМ Економик). – X, 85 стр. : илустр. ; 24 cm. – (Едиција Рачунарске науке)

Предговор упоредо на срп. и енгл. језику. – Тираж 300. – Библиографија.

ISBN 86-7444-007-X

а) Библиотечка каталогизација – Јунимарк (Unimarc) – претраживање б) Апликативни програм ″БИСИС″

Бранко Милосављевић

Претраживање библиографских записа

Нови Сад 2004.

Едиција
Рачунарске науке

Назив публикације
Претраживање библиографских записа

Аутор
др Бранко Милосављевић, доцент,
Факултет техничких наука, Нови Сад

Рецензенти
др Душан Сурла, ред. проф. Природно-математички факултет, Нови Сад
др Зора Коњовић, ред. проф. Факултет техничких наука, Нови Сад

Корице
Мирјана Исаков

Штампа
АБМ Економик

Тираж
300

Трошкове штампања ове монографије обезбедила је Европска Комисија преко TEMPUS пројекта JEP 16114.

Printing costs for this monograph were covered by the European Commission through the TEMPUS project JEP 16114.

Foreword

One of the primary functions of library information systems is the processing of bibliographic data. Bibliographic data consists of descriptions of original published material held by the library. A text server is a particular module of a library information system providing functions for bibliograpic data processing and retrieval.

The main subject of this monograph is a text server that emerged as a result of the development of the library information system BISIS, version 3. The BISIS system is being developed at the University of Novi Sad, and partly through a TEMPUS project *Web Based Interniversity Library Network* (TEMPUS JEP 16114). One of the main goals of this project is to install BISIS in the project's member institutions. Thus a library network providing the infrastructrure for library cooperation via the Internet would be formed.

This monograph presents a complete description of the text server for UNIMARC records used in the BISIS system version 3. It contains the following chapters:

1. Retrieval Models for Textual Documents
2. Representation and Retrieval of Bibliographic Data
3. Modelling the Text Server for UNIMARC Records
4. Implementation of the Text Server for UNIMARC Records

The first chapter presents an overview of the field of information retrieval dealing with the retrieval of textual documents. Library standards that were a foundation upon which the text server is built are presented in Chapter 2. A detailed model of the text server software module is presented in Chapter 3. The final chapter deals with the implementation of the text server and presents examples of its use within the BISIS system.

I would like to thank the reviewers for valuable suggestions that have improved the clarity of the text. Printing costs for the publication of this monograph were covered through the aforementioned TEMPUS project.

Novi Sad, 2004. *Author*

Предговор

Једна од основних функција библиотечких информационих система је обрада библиографске грађе коју чува библиотека. Библиографску грађу чине описи изворних библиографских јединица. Руковање библиографском грађом, што обухвата њено креирање, обраду и претраживање задатак је посебног модула библиотечког информационог система – текст сервера.

Текст сервер који је тема ове монографије настао је у току развоја библиотечког информационог система БИСИС, верзија 3. Систем БИСИС развија се на Универзитету у Новом Саду, а део развоја система реализује се и у оквиру TEMPUS пројекта под називом *Web Based Interuniversity Library Network* (TEMPUS JEP 16114). Један од циљева пројекта је да се систем БИСИС инсталира у институцијама учесника пројекта. На тај начин била би формирана библиотечка мрежа која би обезбедила сарадњу свих библиотека у тој мрежи путем Интернета.

Основни циљ ове монографије је да се на једном месту нађе комплетан опис текст сервера за UNIMARC записе који се користи у оквиру система БИСИС вер. 3. Монографија садржи следећа поглавља:

1. Модели претраживања текстуалних докумената
2. Репрезентација и претраживање библиографске грађе
3. Моделирање текст сервера UNIMARC записа
4. Имплементација текст сервера UNIMARC записа

У првом поглављу дат је преглед дела области проналажења информација (*information retrieval*) који се бави претраживањем текстуалних докумената. Стандарди из области библиотекарства који су били окосница развоја овог текст сервера приказани су у другом поглављу. Детаљни модел текст сервера, укључујући функционалне карактеристике, дат је у трећем поглављу. Имплементација текст сервера, уз примере његовог коришћења у оквиру система БИСИС, разматрана је у четвртом поглављу.

Захваљујем рецензентима на корисним примедбама и сугестијама које су допринеле да је коначна верзија текста прегледнија и разумљивија. Трошкове штампања ове монографије покрива наведени TEMPUS пројекат.

Нови Сад, 2004. *Аутор*

Садржај

Поглавље 1

Модели претраживања текстуалних докумената

Област проналажења информација (*information retrieval*, IR) бави се проблемима репрезентације, складиштења, организације и приступа информацијама [BaezaYates99]. Репрезентација и организација објеката који су носиоци информација требало би да омогуће кориснику задовољење његове потребе за информацијама (*information need*) на једноставан начин. IR системи најчешће рукују документима као носиоцима информација. Класични IR системи рукују текстуалним документима, док се савремена истраживања у овој области баве различитим типовима колекција докумената, као што су колекције слика, видео записа и структурираних мултимедијалних докумената.

Класична монографија из ове области [vanRijsbergen79] и новија монографија [BaezaYates99] наглашавају разлику између проналажења података (*data retrieval*) и проналажења информација (*information retrieval*). Проналажење података се, у контексту IR система, посматра као одређивање докумената чије су карактеристике наведене у оквиру формулације упита којим се проналажење иницира. Са друге стране, корисник IR система је више заинтересован за добијање информација о некој теми него за добијање података који задовољавају постављени упит. Проналажење података има за циљ проналажење свих објеката који задовољавају јасно дефинисане услове. Уколико резултат проналажења података укључује бар један објекат који не задовољава постављене услове, сам поступак проналажења сматра се некоректним. У амбијенту IR система, међутим, поступак проналажења подразумева непрецизност и може да садржи грешке. Основни узрок ове особине проналажења је што се оно често бави садржајима формираним на природним језицима који могу бити непрецизни или семантички вишезначни.

Да би се постигла одређена ефикасност у коришћењу IR система, они морају бити у стању да „интерпретирају“ садржај који претражују и да га ран-

гирају према нивоу *релевантности* за упит корисника. Појам релевантности је један од централних појмова у области IR.

У [BaezaYates99] дата је следећа формална дефиниција модела проналажења информација. Модел проналажења информација је уређена четворка (D, Q, F, G) где је:

- D је скуп репрезентација докумената у колекцији.
- Q је скуп репрезентација корисничких потреба за информацијама. Ове репрезентације називају се *упити.*
- F је оквир (*framework*) за моделовање репрезентација докумената, упита и њихових веза.
- $G : Q \times D \to R$ је функција која додељује релан број сваком пару упита q_i и документа d_j. Ова функција дефинише рангирање докумената у односу на упит q_i.

Монографија [BaezaYates99] наглашава и разлику између два типа интеракције корисника са IR системом: проналажења (*retrieval*) и прегледања (*browsing*). Класични IR системи су оријентисани на проналажење, док су нпр. хипертекст системи обично конципирани као системи за преглед докумената. Комбиновање проналажења и прегледања је тренутно слабо заступљен приступ.

Чланак [Belkin92] разматра карактеристике два различита типа проналажења: тзв. *ad hoc* проналажење и филтрирање (*filtering*). Код конвенционалних IR система колекција докумената којима систем рукује релативно је споро променљива током времена, док су упити краткотрајна и променљива категорија. Овакав начин рада назван је *ad hoc* проналажење. Са друге стране, филтрирање подразумева систем у коме су упити релативно статични, док се нови документи релативно често додају систему. Системи за филтрирање користе интерне репрезентације корисничких потреба за информацијама назване кориснички профили (*user profiles*). Основна разлика профила и упита је у томе што су профили намењени описивању потреба за информацијама које су трајног карактера. Анализа карактеристика конвенционалних IR система и система за филтрирање спроведена у [Belkin92] показује да је сличност IR система и система за филтрирање велика; овакви системи на сличан начин решавају проблем поређења упита и садржаја докумената и њихово рангирање.

У чланку [Meghini01] идентификују се четири димензије докумената. У оквиру приказаног модела каже се да је документ *једноставан* ако се не може даље декомпоновати на друге документе. Једноставан документ је уређени

скуп симбола који носе информацију путем значења, тиме учествујући у ономе што се назива садржај документа. Једноставни документи имају две димензије: форму (или синтаксу) и садржај (или семантику). *Сложени* документи (или само документи) су структуре чији елементи су једноставни документи. Структура докумената се посматра као трећа димензија карактеризације докумената. Документи, једноставни или сложени, постоје као независни ентитети карактерисани атрибутима (*attributes*, *metadata*) који описују релевантне особине докумената. Скуп оваквих атрибута назива се профил (*profile*) документа и представља четврту димензију докумената.

Претраживање докумената по форми узима у обзир синтаксне особине докумената. Упити за претраживање по форми могу сами по себи бити документи (тзв. узорци, *samples*), нарочито у случају претраживања мултимедијалних докумената. Поређење упита и документа се тада своди на поређење њихових интерних репрезентација.

Претраживање засновано на семантици подразумева коришћење симболичких репрезентација значења докумената, односно описа формулисаних на неком од језика за репрезентацију знања. Пример оваквог претраживања је претраживање засновано на анализи просторних односа између објеката на слици [Gudivada95]. Типично се репрезентације значења конструишу ручно (од стране човека), уз евентуалну употребу помоћног алата.

Претраживање докумената по форми може бити схваћено као алтернативни приступ проблему кога решава претраживање по садржају, у смислу одређивања релевантности, односно везе између значења садржаног у упиту и у документима. Овакав приступ подразумева постојање систематске везе између „једнакости“ особина ниског нивоа (датих формом) и „једнакости“ на нивоу значења. У литератури из ове области постоји терминолошка неусаглашеност пре свега у погледу вишедимензионалне карактеризације докумената. Тако се дешава да неки аутори под проналажењем по садржају (*content-based retrieval*) подразумевају оно што је овде дефинисано као проналажење по форми (*form-based retrieval*).

За ефикасно проналажење тражених објеката најчешће се користе специјализоване структуре података назване *индекси*. Данас је познат велики број оваквих структура података; могу се поделити на индексе опште намене (нпр. бинарна стабла и *hash*-табеле [Knuth98]) и индексе специјализоване за одређене специфичне примене (нпр. R-стабла [Guttman84] и 2D-стрингови [Chang87]). Преглед алгоритама и структура података који се користе у класичним (текстуалним) IR системима дат је у [Frakes92].

Системи за индексирање могу се класификовати према три аспекта [Tudhope99, vanRijsbergen79]:

- према томе да ли је садржај индекса генерисан аутоматски или ручно,
- према томе да ли елементи индекса припадају контролисаном речнику или не и
- према томе да ли се елементи индекса могу комбиновати у низ елемената који представља нови концепт приликом индексирања (*pre-coordinated*) или се они комбинују приликом извршавања упита (*post-coordinated*).

Најзначајнија подела је према начину генерисања садржаја индекса. Ручно генерисање најчешће подразумева употребу знања експерта из одговарајуће области. Експерти су у могућности да индексом представе семантику докумената које обрађују. Квалитет ових информација је далеко већи него код система са аутоматском екстракцијом семантике. Са друге стране, трошкови ангажовања експерата и време потребно за формирање индекса су довољан разлог за истраживања у области аутоматске екстракције семантике докумената.

Формулација упита, без познавања колекције докумената која се претражује и начина функционисања IR система, показује се као велики проблем за корисника. У случају да се приликом првог упита не добију потребни резултати, корисник често приступа реформулацији упита. Оваква анализа понашања корисника доноси идеју да се прва формулација упита третира као иницијални (и обично слабо успешан) покушај корисника. Технике за аутоматизацију процеса реформулације упита су у значајној мери изучаване у истраживањима, али су слабо заступљене у комерцијалним IR системима. Технике се могу поделити у три категорије [BaezaYates99]: (а) технике засноване на повратној информацији (*feedback*) корисника, (б) технике засноване на анализи скупа докумената који су иницијално пронађени (често званог локални скуп докумената) и (в) технике засноване на анализи целокупне (глобалне) колекције докумената.

Технике за реформулацију упита засноване на повратној информацији корисника о релевантности пронађених докумената називају се *relevance feedback* (RF) технике. Оне представљају најчешће изучавани начин за реформулацију упита. Прва истраживања RF техника спроведена су у оквиру SMART система [Rocchio71], али је и данас то активно поље истраживања, пре свега за претраживање слика [Lundquist97, Wood98]. Основа свих RF техника је итеративни приступ формулацији упита, при чему корисник само иницијални упит формулише коришћењем упитног језика; касније реформулације упита се израчунавају у оквиру система на основу података о реле-

вантности пронађених докумената које је формирао корисник. Примена RF техника највише је изучавана у оквиру проналажења текстуалних докумената и слика. У [Meghini01] дат је следећи опис RF процеса који важи како за текст, тако и за слике:

1. Корисник поставља упит систему.
2. Систем проналази k најбоље рангираних докумената, сортираних по нивоу процењене релевантности за корисника. Ако је корисник задовољан резултатима претраге, процес проналажења је завршен.
3. За $p < k$ најбоље рангираних докумената корисник уноси сопствену процену њихове релеватности. Најчешће је у питању бинарна вредност (релевантан / није релевантан) али код неких система то може бити и *fuzzy* вредност. Могуће је и не изразити никакву процену за документ.
4. Систем мења своје стање узимајући у обзир податке добијене од корисника.
5. Корак 2 се понавља да би се пронашло нових k најбоље рангираних докумената према текућем стању система.

Технике за реформулацију упита засноване на анализи локалног или глобалног скупа докумената полазе од идеје о уочавању међусобне сличности докумената по неком критеријуму. Реформулација упита обухвата проналажење нових израза који ће, у оквиру упита, боље репрезентовати тражене документе. Рад [Xu96] испитује ефикасност метода локалне и глобалне анализе докумената и наводи боље резултате метода заснованих на анализи локалног скупа. Анализа локалног или глобалног скупа се често третира као проблем формирања кластера докумената у простору дефинисаном на одговарајући начин. Претпоставка од које се полази [Lu96] је да ако су најбоље рангирани документи релевантни, формираће кластер који неће обухватати нерелевантне документе. Кластери се, поред тога, могу користити за формирање структура налик тезаурусима које се, у овом случају, не односе на појмове (тј. речи) него на документе као целине и представљају асоцијације сличности међу њима.

1.1. Модели за претраживање неструктурираних докумената

Класични IR системи омогућавају претрагу текстуалних докумената. При томе се под текстуалним документима подразумевају неструктурирани документи, чији садржај је слободан текст. Колекције докумената као што су TREC [Harman95, Voorhees97] и CACM [Fox83b] намењене су вредновању

перформанси текстуалних IR система. Њих чине документи који имају одређене елементе структуре намењене за метаподатке (нпр. идентификатор документа, порекло), док је сам садржај документа неструктуриран.

У литератури је дефинисан одређени број модела за претраживање неструктурираних текстуалних докумената. Неки од ових модела се могу модификовати тако да се прилагоде и другачијим типовима докумената, што ће бити и изложено у наредним одељцима. У [BaezaYates99] приказана је систематизација до сада познатих IR модела за неструктуриране текстуалне документе:

- Класични модели
  - буловски модел
  - векторски модел
  - пробабилистички модел
- Алтернативни модели
  - засновани на теорији скупова
    - фази модел
    - проширени буловски модел
  - Алгебарски
    - генерализовани векторски модел
    - *latent semantic indexing* модел
    - неуронске мреже
  - Пробабилистички
    - *inference network* модел
    - *belief network* модел

Буловски модел представљен је на више места [Salton83, Wartick92]. Векторски модел, настао као део истраживања везаних за SMART систем [Salton71], дефинисан је у [Salton68]. Пробабилистички модел приказан је у [Robertson76], а детаљно анализиран у [vanRijsbergen79]. Фази модел предложен је у [Ogawa91], мада је и пре њега било истраживања на тему проширивања Буловског модела фази концептима [Radecki76, Radecki77]. Проширени Буловски модел описан је у [Salton83b]. Генерализовани векторски модел дефинисан је у [Wong85]. *Latent semantic indexing* модел приказан је у [Furnas88]. Модел заснован на неуронским мрежама дат је у [Wilkinson91]. Пробабилистички модели засновани су на Бајесовим мрежама [Pearl88]. *Inference network* модел приказан је у [Turtle90, Turtle91]. Његова генерализација, *belief network model*, дат је у [RibeiroNeto96]. У наставку ће бити приказане основне карактеристике неких од поменутих модела који су од интереса за даље излагање у оквиру ове монографије.

Класични IR системи полазе од идеје да је сваки документ представљен скупом репрезентативних речи названих изрази индекса (*index terms*). Израз индекса је, према [BaezaYates99], реч из документа чија семантика помаже у памћењу основне теме документа. За дати скуп израза индекса уочава се да нису сви изрази једнако корисни за описивање садржаја документа. Због тога се изразима додељују нумеричке тежине (*weights*), као квантитативна мера релевантности израза индекса. Следећа дефиниција даје однос појмова израза индекса, документа и тежине.

Нека је $K = \{k_1, k_2, ..., k_t\}$ скуп израза индекса свих докумената у систему. Тежина $w_{ij} \geq 0$ додељена је изразу индекса k_i у документу d_j. За израз индекса који се не појављује у документу d_j важи да је $w_{ij} = 0$. Документу d_j придружен је вектор $\vec{d}_j = (w_{1j}, w_{2j}, ..., w_{tj})$. Функција f_i је таква да је $f_i(\vec{d}_j) = w_{ij}$.

Буловски модел је заснован на теорији скупова и Буловој алгебри. Упити се формирају као логички изрази који имају прецизну семантику. Једноставан је за имплементацију и користи га већина старијих комерцијалних система као што је Dialog [Dialog]. У оквиру буловског модела изрази индекса или јесу присутни у документу или нису. Као последица, придружене тежине узимају вредност из скупа {0, 1}. Упит q се састоји из израза индекса и логичких оператора *and*, *or* и *not*. Упит је, дакле, логички израз који се може представити и у дисјунктивној нормалној форми. На пример, упит $q = k_1 \wedge (k_2 \vee \neg k_3)$ може се представити као $\vec{q}_{dnf} = (1,1,1) \vee (1,1,0) \vee (1,0,0)$, где су конјуктивне компоненте представљене одговарајућим вектором тежина асоцираним са уређеном тројком (k_1, k_2, k_3). Следећа дефиниција даје меру сличности документа из колекције и датог упита.

Нека је q логички израз и $\vec{q}_{dnf}$ његова дисјунктивна нормална форма. Нека је q_{cc} било која конјуктивна компонента из $\vec{q}_{dnf}$. Сличност документа d_j са упитом q изражава се вредношћу функције

$$sim(d_j, q) = \begin{cases} 1, \exists \vec{q}_{cc} \mid (\vec{q}_{cc} \in \vec{q}_{dnf}) \wedge (\forall k_i \ f_i(\vec{d}_j) = f_i(\vec{q}_{cc})) \\ 0, \text{ inace} \end{cases}$$

Буловски модел омогућава да се документ у колекцији за дати упит карактерише као релевантан или нерелевантан. Немогућност изражавања делимичног поклапања документа са упитом (*partial match*) представља његову основнуману.

Векторски модел исправља основнуману буловског модела тако што омогућава додељивање не-бинарних тежина као и изражавање делимичног поклапања документа са упитом. Тежине се, у оквиру овог модела, додељују како изразима индекса, тако и изразима упита.

Тежина $w_{ij} \geq 0$ додељује се пару (k_i, d_j). Тежина $w_{iq} \geq 0$ додељује се пару (k_i, q). Вектор упита $\vec{q}$ дефинише се као $\vec{q} = (w_{1q}, w_{2q}, ..., w_{tq})$ где је t укупан број израза индекса у систему. Вектор документа $\vec{d}_j$ дефинише се као $\vec{d}_j = (w_{1j}, w_{2j}, ..., w_{tj})$.

На овај начин, вектор документа и вектор упита могу се посматрати као вектори у t-димензионалном простору. Мера сличности између два вектора најчешће се изражава на следећи начин [Salton88]:

$$sim(d_j, q) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{ij} \cdot w_{iq}}{\sqrt{\sum_{i=1}^{t} w_{ij}^2} \cdot \sqrt{\sum_{i=1}^{t} w_{iq}^2}}$$

Вредност $sim(d_j, q)$ налази се у интервалу [0, 1]. Уколико се, приликом постављања упита, дефинише минимална вредност ове функције, могу се као резултат издвојити само они документи чија сличност са упитом прелази дати праг. Резултат упита се, поред тога, може сортирати по сличности са упитом у опадајућем редоследу тако да се најбоље рангирани документи нађу на врху листе резултата.

Одређивање вредности тежина за изразе индекса може се обавити на више начина. У [Salton83] дат је приказ различитих начина за ову калкулацију. Често коришћен начин заснива се на формирању кластера (*clusters*) докумената. Проблем формирања кластера овде се своди на поделу целокупне колекције докумената на два дела: кластер са документима који су релеватни за конкретну потребу корисника и остатак колекције докумената. Као мера сличности између докумената који се налазе унутар траженог кластера (*intra-cluster similarity*) користи се учестаност појављивања израза k_i у документу d_j. Ова мера, у литератури обично названа *tf factor*, одређује колико добро дати израз описује садржај документа. Мера различитости између докумената из различитих кластера (*inter-cluster dissimilarity*) је инверзна фреквенција израза k_i (тзв. *idf factor*). Идеја формулације ове мере је да се изрази који се појављују у свим документима тешко могу искористити за разликовање релеватних докумената од нерелевантних. У наставку је дата формална дефиниција ових величина.

Нека је N укупан број докумената у колекцији и n_i број докумената у којима се појављује израз индекса k_i. Нека је $freq_{ij}$ број појављивања израза k_i у документу d_j. Нормализована учестаност појављивања f_{ij} израза k_i у документу d_j дата је као

$$f_{ij} = \frac{freq_{ij}}{\max\limits_{l \in d_j} \{\text{freq}_{\text{ij}}\}}$$

где се максимум броја појављивања одређује из скупа свих израза који се јављају у документу d_j. Инверзна учестаност израза k_i, у ознаци idf_i, дефинише се на следећи начин:

$$idf_i = \log \frac{N}{n_i}$$

Вредности за тежине w_{ij} додељене изразима одређују се као:

$$w_{ij} = f_{ij} \cdot \log \frac{N}{n_i}$$

Вредности за тежине додељене упиту w_{iq} дате су изразом [Salton88]:

$$w_{iq} = \left(0.5 + \frac{0.5 \cdot freq_{iq}}{\max\limits_{l \in q} freq_{lq}} \right) \cdot \log \frac{N}{n_i}$$

Пробабилистички модел, познат у литератури и као *binary independence retrieval* модел, полази од појма *идеалног скупа* пронађених докумената за дати упит кога чине сви документи које би корисник оценио као релевантне; нерелевантни документи нису елементи овог скупа. Како особине тог скупа нису познате унапред, потребно је учинити иницијално погађање. Након тога могуће је пронаћи иницијални скуп докумената. Затим се интеракција са корисником одвија у циљу унапређивања пробабилистичког описа идеалног скупа.

За дати упит q и документ d_j потребно је одредити вероватноћу да ће документ d_j бити релевантан за корисника. Претпоставка је да ова вероватноћа зависи искључиво од карактеристика упита и документа. Даље, претпоставља се да идеалан скуп погодака R постоји.

У пробабилистичком моделу, тежине додељене изразима индекса и елементима упита су бинарне, тј. $w_{ij} \in \{0,1\}$ и $w_{iq} \in \{0,1\}$. Упит q представља

подскуп скупа свих израза индекса. Нека је R скуп докумената који се сматрају релевантним. Нека је $\overline{R}$ комплемент скупа R. Нека је $P(R|d_j)$ вероватноћа да је документ d_j релевантан за упит q и $P(\overline{R}|d_j)$ вероватноћа да d_j није релевантан за q. Сличност документа d_j са упитом q изражава се као

$$sim(d_{j,}q) = \frac{P(R \mid d_j)}{P(\overline{R} \mid d_j)}$$

У [BaezaYates99] дат је изведени облик претходног израза који гласи:

$$sim(d_j, q) \approx \sum_{i=1}^{t} w_{iq} w_{ij} \left(\log \frac{P(k_i \mid R)}{1 - P(k_i \mid R)} + \log \frac{1 - P(k_i \mid \overline{R})}{P(k_i \mid R)} \right)$$

где је $P(k_i|R)$ вероватноћа да је израз k_i присутан у документу случајно изабраном из скупа R. Аналогно томе дефинисана је и величина $P(k_i|\overline{R})$. Како скуп R није познат на почетку претраживања, потребно је одредити иницијалне вредности за ове величине. Пре првог претраживања, у тренутку када још нема пронађених докумената, може се претпоставити да је $P(k_i|R)$ једнако за све изразе k_i (типично једнако 0.5) и да се расподела израза индекса међу нерелевантним документима може апроксимирати дистрибуцијом свих израза индекса међу свим документима у колекцији. На тај начин може се писати

$$P(k_i \mid R) = 0.5$$

$$P(k_i \mid \overline{R}) = \frac{n_i}{N}$$

Након што се пронађе иницијални скуп докумената који задовољавају упит, могуће је дефинисати скуп V као скуп од првих r рангираних докумената где је r унапред дефинисан праг. Нека је V_i скуп оних докумената из V који садрже израз k_i. Усвајају се следеће претпоставке: а) $P(k_i|R)$ може се апроксимирати расподелом израза k_i међу пронађеним документима, и б) $P(k_i|\overline{R})$ може се одредити на основу претпоставке да су сви непронађени документи уједно и нерелевантни. Имајући у виду ове претпоставке, можемо писати:

$$P(k_i \mid R) = \frac{|V_i|}{|V|}$$

$$P(k_i \mid \overline{R}) = \frac{n_i - |V_i|}{N - |V|}$$

Претходне формуле се, због проблема које у пракси стварају одређене вредности за V и V_i [BaezaYates99], модификују тако да гласе:

$$P(k_i \mid R) = \frac{|V_i| - n_i / N}{|V| + 1}$$

$$P(k_i \mid \overline{R}) = \frac{n_i - |V_i| + n_i / N}{N - |V| + 1}$$

Последње формуле могу се користити у свим наредним итерацијама претраживања.

Као мане пробабилистичког модела у [BaezaYates99] наведене су следеће карактеристике: а) потреба за погађањем иницијалне поделе докумената на релевантне и нерелевантне, б) модел не узима у обзир учестаност понављања израза у документу (тежине w_{ij} су бинарне) и в) претпоставка да су изрази индекса међусобно независни у смислу њиховог појављивања у документима. Главна врлина овог модела је што се документи рангирају према вероватноћи да су релеватни за корисника.

Карактеристике овде приказаног пробабилистичког модела експериментално су анализиране у [SparckJones79]. У чланку [Croft79] приказана је варијанта модела која не користи повратне информације корисника за процену вероватноћа. Исти аутор у [Croft83] разматра додавање тежина везаних за учестаност појављивања унутар документа у модел.

У литератури постоји опште мишљење да је буловски модел најслабији од класичних метода, пре свега због непостојања могућности за делимично поклапање документа са упитом. У раду [Losee88] разматра се интеграција Буловских упита у пробабилистички модел. Анализа експеримената спроведена у [Croft83] наводи да пробабилистички модел има боље перформансе од векторског. Каснији рад [Salton88] оповргава ову тврдњу и показује да се може сматрати да векторски модел има боље перформансе од пробабилистичког за опште колекције докумената. Векторски модел се може данас сматрати најпопуларнијим моделом међу истраживачима, али и у пракси, пре свега у пројектима веб претраживача [BaezaYates99].

1.2. Модели за претраживање структурираних докумената

У овом одељку биће речи о IR моделима проналажења структурираних текстуалних докумената. Модели проналажења мултимедијалних докумената

најчешће обухватају претрагу по тексту и по структури, али ће о њима бити речи у посебном одељку. Развој IR модела намењених проналажењу структурираних докумената започео је знатно касније у односу на развој класичних IR модела. Рани радови [Stonebraker83, Desai86] баве се интеграцијом функција претраживања текстуалних докумената у класичне релационе системе за управљање базама података. Данас важи став да комбиновање садржаја и структуре докумената у процесу проналажења докумената пружа веће могућности у претраживању него сваки механизам понаособ. При томе се за претраживање по структури сматра да представља претраживање података, а не претраживање информација, с обзиром на то да сви модели подразумевају егзактно поређење структура. Чланак [BaezaYates96] садржи анализу класичних IR модела и закључак да они не могу ефикасно да подрже захтеве који се постављају пред системе за проналажење структурираних докумената. Поред тога, наглашава правило реципрочног односа између изражајности модела и ефикасности његове имплементације које важи за анализиране моделе за структуриране документе.

Хибридни модел (*hybrid model*, [BaezaYates94]) посматра колекцију докумената који могу имати дефинисана поља. Поља не морају прекривати текст у потпуности, могу се угњеждавати и преклапати. Упитни језик је дефинисан као алгебра над паровима (*D*, *M*) где је *D* скуп докумената а *M* скуп позиција у тексту које могу бити поређене са речима или шаблонима (*patterns*). Дефинисан је одређени број операција којима се такве позиције генеришу, нпр. претраживање по префиксу речи, близински оператори итд. Скуповне операције уније, пресека, разлике и комплемента дефинисане и за скупове докумената и за скупове позиција (за ограничавање позиција само унутар датих поља или за проналажење поља која садрже дату позицију). Због своје једноставности модел је једноставан за имплементацију.

PAT изрази (*PAT expressions*, [Salminen92]), имплементирани у оквиру система *PAT Text Searching System* [Fawcett89], садрже индексе само над позицијама, а не и над структуром документа. Приказани језик омогућава динамичко дефинисање структуре засновано на изразима за дефинисање позиција које одређују почетак и крај континуалног региона текста. Изрази морају бити специфичног облика, тако да су везани за одговарајући тип означавања блокова. Систем је успешно примењен у пројекту OED (*Oxford English Dictionary*) [Gonnet87].

Преклапајуће листе (*overlapped lists*) приказане су у [Clarke95a, Clarke95b]. Структура докумената представља се помоћу више листа које се састоје од дисјунктних, континуалних региона текста. Дефинисане операције над регионима су релативно једноставне: селекција датог региона или речи;

селекција региона који (не) садржи дати регион; селекција региона који (ни)је садржан у другом региону. Поред тога, дефинисане су и класичне IR операције као што је рангирање по релевантности.

Листе референци (*lists of references*, [MacLeod91, MacLeod90]) се, приликом моделирања докумената, ослањају на концепте дефинисане SGML стандардом [SGML86] и концепте познате из објектно-оријентисаних база података, мада нису директно везане за њих. Документи поседују тачно једну хијерархијску структуру, где хијерархија означава везу садржавања међу регионима документа. Чворови структуре могу поседовати атрибуте који се могу употребљавати у претраживању. Постоји и концепт линкова преузет из хипертекст система. Модел поседује велику изражајност у моделирању докумената и формирању упита али доноси проблеме приликом дизајна ефикасне имплементације [MacLeod91].

Поређење стабала (*tree matching*, [Kilpelainen93]) користи концепт садржавања стабла (*tree inclusion*) за описивање структуре упита и базе података, при чему се као основни проблем посматра проналажење оних делова структуре у бази података који одговарају шаблону датим помоћу упита. Разматране су две варијанте; уређено садржавање (*ordered inclusion*) води рачуна о редоследу чворова потомака, док неуређено садржавање (*unordered inclusion*) тај редослед занемарује. Листови шаблона упита могу бити изрази који садрже текстуални шаблон. Резултати упита су такође стабла. Упитни језик поседује концепт променљивих које омогућавају формирање израза за једнакост упита само са појединим деловима тражене структуре, као и операције уније и пресека и симулацију операције споја познате из релационог модела података. Рад [Kilpelainen93] је формиран пре свега као анализа особина процеса поређења стабала. Према анализи у [Kilpelainen95], неуређено садржавање је NP-комплетан проблем [Garey79]. Самим тим, модел није погодан за имплементацију у реалним системима.

Радови [Navarro97, Navarro95] представљају модел блиских чворова (*proximal nodes*) који обухвата модел докумената и језик претраживања. Основна намера аутора је да се дефинише модел који има равнотежу у погледу изражајности моделирања структуре докумената и упита и ефикасне имплементације. Колекција (тј. база у датој терминологији) докумената поседује две компоненте:

– *Текст* који се посматра као секвенца симбола (били они карактери, речи, или нешто друго). Није битно да ли текст садржи ознаке структуре (*markup*) или не; оне се сматрају делом текста.

– *Структуру* која представља скуп међусобно независних хијерархија. Региони текста које покривају чворови различитих хијерархија могу се преклапати, али се то не може десити унутар исте хијерархије. Хијерархије не морају да покрију целокупан текст.

Дефинисана је алгебра над скуповима чворова, са одговарајућим операторима над тим скуповима. Приказана је и софтверска архитектура могуће имплементације оваквог система. Перформансе система су анализиране на нивоу модела; имплементација система (у тренутку писања овог текста) не постоји као ни подаци о њеним перформансама. Концепт модела је такав да се ослања на неки од постојећих модела за физичко руковање подацима (релациони, објектно оријентисани, итд.) и понавља тезу истих аутора дату у [BaezaYates96] да је пожељно искористити постојеће моделе података и њихове имплементације за оне сегменте руковања подацима у којима су се показали као ефикасни. У раду се такође разматра и могућност употребе овог модела за рад са мултимедијалним документима.

Рад [BaezaYates96] доноси и систематизацију модела намењених проналажењу структурираних текстуалних докумената. Систематизација је дата по више аспеката који ће и овде бити наведени.

Изражајност модела структуре. Како у тренутку објављивања рада [BaezaYates96] није постојао консензус око структурирања базе докумената, извршена је анализа изражајности појединих модела. Посматране су следеће карактеристике:

– *Тип структуре докумената*
  – **Раван.** Случај када један елемент структуре не може да садржи други елемент.
  – **Хијерархијски.** Најчешће коришћени тип структуре, где хијерархија представља однос садржавања елемената. Модели се међусобно разликују по томе да ли дозвољавају рекурзивне структуре, постојање више независних хијерархија или преклапање региона текста који одговарају елементима структуре.
  – **Мрежни.** Ниједан од разматраних модела не поседује карактеристике мрежног модела, мада листе референци поседују могућност произвољног повезивања елемената али пре свега за потребе навигације.
– *Имплицитна или експлицитна структура*. Имплицитну структуру користе модели који се ослањају на парсирање текста или посебне ознаке структуре у тексту (*markup*). Другим речима, не поседују одвојене податке о структури докумената, већ се они налазе унутар докумената. Модели са имплицитном структуром не могу имати особине које се тешко могу представи-

ти *markup* механизмом, као што су рекурзивне структуре. Са друге стране, преклапање елемената структуре се једноставно изражава на овај начин. Израчунавање упита је други важан аспект ове класификације: модели са имплицитном структуром могу структурне упите свести на изразе за поређење шаблона (*pattern matching*) и тиме користити имплементације познате из класичних IR система, док модели са експлицитном структуром могу једноставније одговорити на упите о односима предак/потомак међу елементима структуре.

- *Статичка или динамичка структура*. Неки модели функционишу довољно добро уз претпоставку да је структура докумената мање или више статичка, тј. непроменљива током времена. Други су више оријентисани на манипулацију динамичким структурама. Може се уопштено рећи да модели са имплицитном структуром користе динамичке структуре док модели са експлицитном структуром очекују статичке структуре.
- *Веза између садржаја и структуре*. Овај аспект приказује важну особину сваког модела у погледу његове оријентације ка одређеној врсти упита. Неки модели су оријентисани више ка претраживању текста, док су други намењени претраживању по структури. Подељени су у три категорије:
  - доминантно текстуални (*strongly text-bound*)
  - доминантно структурни (*strongly structure-bound*)
  - средњи (*intermediate*)
- *Структура резултата*. Структура резултата у појединим моделима не мора бити еквивалентна структури докумената. Уочена су три типа структуре резултата:
  - равни (*flat*)
  - преклапајући (*overlapped*)
  - угњеждени (*intermediate*)

Упитни језик. Анализа упитног језика обухвата неколико особина:

- *Претраживање текста*. Овај аспект анализира могућности упитног језика које се односе на поређење текста (*pattern matching*).
- *Манипулација скуповима*. Сви модели представљају резултат упита као скуп ентитета. Модели који немају угњеждене резултате представљају тај скуп као уређену листу. Већина модела дефинише операције уније, пресека и разлике скупова резултата, мада сваки поседује и своје специфичне могућности.
- *Везе садржавања*. Проналажење елемената структуре који садрже друге елементе или су садржани у другим елементима подразумева познавање веза садржавања у документима. Приказани модели највише се разликују по третману овог концепта.

– *Растојања*. Могућност изражавања ограничења на међусобно растојање текста или структуре је на различит начин заступљена код различитих модела. Неки модели занемарују ову могућност која се у пракси показује као изузетно важна.

Сложеност израчунавања упита. Типично је изражајност модела и упитног језика обрнуто сразмерна рачунској сложености израчунавања упита. Уочено је неколико класа сложености:

– $O(n)$: хибридни модел, PAT изрази и преклапајуће листе.
– Скоро увек $O(n)$: блиски чворови омогућавају имплементацију линеарне сложености у већини ситуација.
– $O(n \log n)$: већина операција у моделу листе референци има ову сложеност.
– Не-полиномијална: поређење стабала дефинише операције које имају сложеност већу од полиномијалне.

Ниједан од приказаних модела не разматра проблем рангирања докумената приликом формирања резултата упита. Једна варијанта рангирања структурираних докумената дата је у [Kaszkiel99]. Документи се, за потребе рангирања, посматрају као низови пасуса, где се под пасусом подразумева низ речи фиксне дужине који се појављује било где у тексту. Приликом проналажења докумената могуће је кориснику приказати само одговарајући пасус, тако да је овај систем посебно погодан у ситуацијама где се рукује документима изузетно велике дужине (нпр. судски стенограми). Иако је приказани метод намењен пре свега рангирању неструктурираних докумената, за потребе израчунавања ранга користи се структура над текстом.

1.3. XML стандард

Након објављивања XML препоруке [XML], интересовање за системе за руковање структурираним текстуалним документима је нагло порасло. Изузетна подршка софтверске индустрије овом стандарду, реализована кроз развојне алате, алате за крајње кориснике, мноштво стандарда за размену докумената у појединачним областима (финансије, медицина, хемија, географски информациони системи) заснованих на XML-у установили су консензус у истраживачкој заједници око усвајања XML-а као стандардног модела структурираних докумената. Иако SGML [SGML86] стандард постоји од 1986. године, због своје сложености био је ограничен на мали број специфичних примена. XML, замишљен као функционални подскуп SGML-а, за кратко време је усвојен као стандардан језик у великом броју примена захваљујући, између осталог, и својој једноставности.

Као средство за дефинисање структуре XML докумената иницијално је коришћен DTD (*Document Type Definition*) формат, саставни део XML препоруке. Ограничене могућности DTD-а у погледу дефинисања ограничења на типове података који се користе у документима довеле су до доношења новог стандарда за дефиницију структуре докумената, XML Schema [XMLSchema1, XMLSchema2].

XML је праћен одређеним бројем других препорука које се баве појединим аспектима руковања XML документима. XSLT [XSLT] је намењен за трансформације XML докумената, XLink [XLink] дефинише линкове међу њима, а XPath [XPath] синтаксу за референцирање појединих делова документа. Питање стандардног упитног језика решено је дефинисањем XQuery језика [XQuery], мада је пре њеног усвајања дефинисано неколико упитних језика за XML (XQL [XQL], XML-QL [XMLQL], итд.) са донекле различитим карактеристикама. Компаративна анализа карактеристика неколико предложених упитних језика за XML дата је у [Bonifati00].

Интерес истраживача из области IR текао је у правцу адаптирања постојећих модела структурираних докумената за рад са XML документима. Рад [BaezaYates00] доноси анализу могућности примене модела блиских чворова за имплементацију XQL упитног језика. Јединствени модел структуре XML докумената не постоји чак ни када су у питању формалне спецификације стандарда. У оквиру активности W3 конзорцијума развијена су четири модела структуре докумената: *XML Information Set* модел [XMLInfoset], XPath 1.0 модел [XPath], DOM модел [DOM1] и XQuery 1.0 модел [XQueryDataModel]. Новина коју XML доноси у односу на раније приказане моделе структурираних докумената је концепт атрибута везаног за елемент документа. Атрибути се, према [Salminen01], типично користе за метаподатке, мада је и њихова употреба за смештање података некад могућа (да би се избегло наметање редоследа међу чворовима потомцима, или за референцирање на екстерно смештене делове документа).

Питање еквиваленције XML докумената нема јединствен одговор. Неки аутори занемарују редослед чворова потомака [Abiteboul01], док га други узимају у обзир [Fan01]. W3C препорука *Canonical XML* [CanonicalXML] дефинише тзв. каноничку форму докумената који учествују у поређењу. Међутим, превођење садржаја документа у каноничку форму занемарује неке податке из оригиналног документа.

Примена XML језика као стандарда за размену података између разнородних информационих система привукла је и истраживаче из области база података [Vianu01]. У раду [Salminen01] анализиране су потребне каракте-

ристике система за руковање XML документима. Дата је дефиниција базе XML докумената (*XML document database*) као „колекција XML докумената и њихових делова којом рукује систем способан за управљање колекцијом и информацијама репрезентованим колекцијом“. Рад [ArnoldMoore99] наводи следеће особине система за руковање структурираним документима:

- креирање докумената на захтев (*on-the-fly creation of renditions*),
- аутоматске трансформације докумената,
- контрола приступа на нивоу елемената,
- приступ само појединим елементима,
- верзије докумената,
- описи промена докумената читљиви за човека,
- подршка за рад заснован на токовима документима и
- проширене могућности претраживања (комбиновање претраге по садржају и структури, рангирање погодака).

Рад [Salminen01] идентификује важне карактеристике XML база података. Приказане карактеристике се не односе ни на један постојећи систем, већ су намењене за креирање контекста за вредновање појединих система. Као основне карактеристике наведене су следеће:

- подршка за богат скуп основних типова података, нпр. какав дефинише XML Schema [XMLSchema2],
- могућност дефинисања више типова докумената, помоћу њихових DTD-а или XML Schema описа,
- руковање колекцијама докумената, при чему се колекције докумената виде као „равне“, тј. не постоји могућност угњеждавања колекција,
- могућност руковања колекцијама типова докумената, нпр. за потребе дефинисања структуре више верзија истог документа,
- рад са више нивоа провере валидности документа,
- правилно руковање *entity reference* пољима у XML документу и URI [URI] адресама,
- подршка за XML просторе имена (*namespaces*, [XMLNamespaces]),
- могућност избора различитих типова индекса приликом индексирања и претраживања докумената,
- контрола приступа заснована на корисничким улогама (*role-based access control*),
- претраживање коришћењем неког од упитних језика,
- трансформације докумената за потребе приказивања, интеграције са другим системима, еволуције структуре докумената и генерисања погледа,

- ажурирање докумената, са провером референцијалног интегритета у XML смислу (за IDREF атрибуте, *entity reference* поља и линкове ка документима који су у истој бази података) и трансакционим режимом рада.

У литератури је објављен и одређени број радова на тему мапирања модела XML докумената на релациони модел података. Могу се уочити два приступа решавању овог проблема:

- мапирање логичке структуре докумената (дате DTD-ом или XML Schema документом) на релациону шему и
- мапирање општег модела XML докумената (као што је DOM) на релациону шему.

Мапирање логичке структуре докумената на релациону шему подразумева креирање засебне релационе шеме за сваки тип документа. Такво мапирање обично обухвата креирање посебне релације за сваки тип елемента у документима. Овакав метод приказан је у радовима [Christophides94, Abiteboul97, Kovcic00, Varlamis01]. Нешто сложенији метод мапирања, који узима у обзир детаљну анализу DTD-а дат је у [Shanmugasundaram99].

Мапирањем општег модела XML докумената на релациону шему добија се јединствена релациона шема која се користи за све типове докумената. Неколико радова бави се овим начином мапирања [Zhang95, Florescu99b, Bourret01, Yoshikawa01].

Рад [Yoshikawa01] као ситуације погодне за усвајање првог приступа наводи велике колекције докумената који припадају малом броју различитих типова, при чему су типови непроменљиви током времена. Са друге стране, апликације које користе већи број типова докумената или где су типови докумената непознати или променљиви током времена могу боље искористити методе мапирања засноване на другом принципу.

Системи за управљање релационим базама података су тренутно најраширенији [Leavitt00]. Истраживања о мапирању структуре XML докумената на друге типове модела података (нпр. објектно-оријентисани) нису толико бројна.

Претраживање XML докумената засновано је на неком од предложених упитних језика. У [Yoshikawa01] приказан је алгоритам за генерисање SQL упита на основу датих XQL упита за XRel систем. Рад [Varlamis01] приказује архитектуру X-Database система. Упити у овом систему формирају се као XML документи структурирани према датој XML Schema спецификацији. Рад [Wong01] приказује особине SODA2 система, који користи проширену

XQL синтаксу за формирање упита. Најзначајније проширење XQL језика представља могућност употребе регуларних израза (*regular expressions*). Дефинисана је и одговарајућа индексна структура за подршку оваквим упитима. Радови [Schlieder00, Schlieder01] приказују ApproXQL, проширење XQL-а које омогућава упите по структури докумената који проналазе и парцијална слагања докумената са упитом. Имплементација се заснива на проблему неуређеног садржавања стабала анализираног и раније у оквиру [Kilpelainen93, Kilpelainen95], али користи нови алгоритам који је експоненцијалне сложености само у најгорем случају.

Квалитативна новина коју доноси ApproXQL је могућност парцијалног слагања документа са упитом и рангирање докумената према степену слагања са упитом. Рангирање пронађених XML докумената обрађивано је у оквиру конструкције још два упитна језика, XXL [Theobald00] и ELIXIR [Chinenyanga01].

Оријентација свих поменутих упитних језика, осим ApproXQL, на егзактно поређење података последица је полазног приступа њихових аутора који долазе из области база података. Третман употребе XML докумената у том амбијенту обично се назива *data-centric*: XML се углавном посматра као нови модел података који има већу изражајност у односу на релациони модел. Са друге стране, третирање XML језика као стандарда за формирање структурираних текстуалних докумената, у којима се претраживање схвата као проналажење информација а не као проналажење података, назива се *document-centric*. Аутори *document-centric* оријентације обично долазе из области IR.

1.4. Relevance feedback у претраживању текста

1.4.1 Технике са интеракцијом са корисником

У раду [Lundquist97] дата је следећа дефиниција за *relevance feedback*: „RF је процес у коме се упит селективно модификује да би пронашао релевантније документе у колекцији него његова иницијална верзија“. У текстуалним системима модификација упита обухвата две операције: 1) промену тежинских коефицијената додељених елементима упита (*term reweighting*) и 2) додавање нових израза у упит (*query rewriting*). Нови изрази се проналазе у оквиру иницијално пронађених докумената за које се сматра да су релевантни. Модификација тежинских коефицијената у оквиру класичног векторског модела први пут је разматрана у [Rocchio71, Ide71]. Ови резултати и данас

представљају основу за имплементацију RF технике у векторском моделу. Нека је, у оквиру векторског модела, дефинисан следећи скуп величина:

- D_r: скуп релевантних докумената као подскуп скупа пронађених докумената, према процени корисника,
- D_n: скуп нерелевантних докумената у оквиру скупа пронађених докумената,
- C_r: скуп релевантних докумената у оквиру целокупне колекције,
- $|D_r|$, $|D_n|$, $|C_r|$: број елемената скупова D_r, D_n и C_r, тим редоследом и
- α, β, γ: константе за подешавање.

Идеалан случај представља ситуација када је скуп C_r познат унапред за дати упит q. Може се показати да је најбољи вектор упита који раздваја релевантне документе од нерелевантних гласи [BaezaYates99]:

$$\vec{q}_{opt} = \frac{1}{|C_r|}\sum_{\forall \vec{d}_j \in C_r} \vec{d}_j - \frac{1}{N-|C_r|}\sum_{\forall \vec{d}_j \notin C_r} \vec{d}_j$$

Како скуп C_r није познат унапред, RF техника покушава да инкрементално дође до идеалног решења полазећи од иницијалног упита. Инкременталне модификације заснивају се на подацима о релевантним и нерелевантним документима међу онима који су управо пронађени. У литератури су познате три класичне формуле за модификацију вектора упита:

$$\vec{q}_m = \alpha\vec{q} + \frac{\beta}{|D_r|}\sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{\gamma}{|D_n|}\sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

$$\vec{q}_m = \alpha\vec{q} + \beta\sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma\sum_{\forall \vec{d}_j \in D_n} \vec{d}_j$$

$$\vec{q}_m = \alpha\vec{q} + \beta\sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \gamma \max_{nonrelevant}(\vec{d}_j)$$

где је $\max_{\text{nonrelevant}}(\vec{d}_j)$ ознака за најбоље рангирани нерелевантни документ. У оригиналним формулацијама [Rocchio71, Ide71] вредности константи за подешавање су α = β = γ = 1. Данас се сматра да све три технике дају приближно једнаке резултате [BaezaYates99]. Приказане технике карактерише једноставност (модификације вектора се рачунају на основу скупа пронађених докумената) и добри резултати који су експериментално установљени. Са друге стране, не постоји формулација критеријума оптималности за итеративни процес.

Приказ RF технике за пробабилистички модел дат је у одељку 1.1. Као мане овог приступа у [BaezaYates99] наводи се следеће:

- тежине израза у документима се не узимају у обзир током *feedback* циклуса,
- тежине додељене изразима упита у претходним итерацијама се не узимају у обзир и
- упит се не проширује новим изразима.

Као резултат ових особина, RF техника у пробабилистичком моделу је мање ефикасна од технике развијене за векторски модел. Рад [Harman92] анализира експерименте са RF техникама за векторски и пробабилистички модел и закључује да векторски модел показује добре резултате са већином стандардних тест колекција, док пробабилистички има проблема са одређеним колекцијама. Варијанте проширивања упита новим изразима у пробабилистичком моделу предложене су у [Wu81, Harper78]. Проширивање буловског модела пробабилистичким техникама процене тежина за RF истраживано је у више радова [Radecki83, Salton84, Croft91]. Рад [Haines93] анализира употребу RF технике код *inference network* модела. Радови [Haines93, Croft91] приказују RF технике за структуриране упите али под структуром подразумевају фразе као секвенце речи у неструктурираним текстуалним системима. Такав појам структуре не може се повезати са моделима за структуриране документе приказаним у одељку 1.2.

Relevance feedback технике у моделима са структурираним документима нису разматране. Већина ових модела нема могућност парцијалног слагања докумената са упитом, па тако ни могућност рангирања резултата, односно употребе RF метода за корекцију резултата. Језик ApproXQL [Schlieder00, Schlieder01] омогућава парцијално слагање докумената са упитом и има механизам рангирања резултата али употреба RF техника није разматрана.

Рад [Spink98] анализира *relevance feedback* са становишта интеракције корисника са IR системом и уочава пет типова интеракције:

- *Content relevance feedback* (CRF). Корисник анализира одговор система на иницијални упит и процењује релевантност пронађених докумената. Ти подаци се користе за нови *feedback* циклус.
- *Term relevance feedback* (TRF). Корисник анализира одговор система на иницијални упит и бира нове изразе из пронађених докумената за упит у наредном циклусу.
- *Magnitude feedback* (MF). Корисник анализира величину приказаног скупа погодака и захтева мањи, већи или скуп непромењене величине у наредном циклусу.

- *Tactical review feedback* (TCR). Корисник анализира изразе коришћене у претходним циклусима претраге ради одређивања даље стратегије формирања упита.
- *Term review feedback* (TMR). Корисник одређује даљу стратегију претраге након прегледа израза који се налазе у индексу.

Ефикасност различитих типова интеракције и њихов удео у укупном броју интеракција анализирани су на узорку упита које су студенти редовних и последипломских студија постављали Диалог систему. Као основна мера ефикасности интеракције посматран је број *feedback* петљи (*loops*). Аутори закључују да највећи број циклуса био MF и CRF типа. TRF тип, иако детаљно обрађен у истраживањима, није био значајно заступљен. Последња два типа, усмерени ка формирању даље стратегије претраге, иако ређе заступљени од осталих, уочени су као важни за успешан исход процеса претраге у неким ситуацијама.

1.4.2. Аутоматске технике

RF технике које врше реформулацију упита не могу се називати аутоматским јер користе процену корисника о релевантности пронађених докумената. Аутоматске RF технике подразумевају потпуно одсуство учешћа корисника у току реформулације упита. Такве технике полазе од претпоставке да је одређени број најбоље рангираних докумената релевантан. Системи засновани на векторском моделу користе класичне RF формуле за векторски модел у којима је део који се односи на нерелевантне документе занемарен. Чланак [Salton90] анализира учинак аутоматске RF технике засноване на формулама за ручне технике са константом $\gamma = 0$. Иако је постигнути резултат слабији него када се узму у обзир и нерелевантни документи, постигнуто је побољшање у односу на проналажење докумената без RF технике.

1.5. Кластери докумената

Кластер анализа (*cluster analysis*) је назив за велику фамилију метода за класификацију објеката. Објекти који су предмет класификације се, за потребе кластеровања, посматрају као тачке у одговарајућем метричком или векторском простору. У [Everitt80] кластери су описани као континуални региони у простору који имају релативно велику густину тачака, раздвојени од других таквих региона регионима који имају релативно малу густину тачака.

Алгоритми за кластеровање који се користе у IR системима могу се грубо класификовати у две основне групе [Wen02, Hearst96, Leuski01]. Прву групу чине итеративни партициони алгоритми, који формирају кластере почевши од полазног скупа тачака и иницијалне партиције простора. Потом се итеративним поступком врши оптимизација изабраног критеријума. Алгоритми ове групе разликују се, пре свега, у погледу избора оптимизационог критеријума и начина одређивања иницијалног решења. Најпознатији представник ове групе алгоритама је *k-means* алгоритам који кластере репрезентује својим центроидима, тачкама простора које представљају средњу вредност координата свих тачака чланица кластера.

k-means формира партицију простора са n тачака у k кластера на следећи начин:

1. Одреди се иницијалних k кластера. Иницијално решење најчешће се одређује као k међусобно најудаљенијих тачака или првих k тачака у посматраном скупу.
2. Све тачке посматраног скупа додељују се најближем кластеру, тј. кластеру чији је центроид најближи посматраној тачки.
3. Врши се поновна калкулација положаја центроида кластера.
4. Корак 2 се понавља док се критеријум заустављања не задовољи.

Основна врлина *k-means* алгоритма је линеарна временска зависност од броја тачака, што га чини погодним за кластеровање великих скупова. Са друге стране, овај алгоритам претпоставља да кластери имају сферни облик (у складу са коришћеном метриком), што не мора увек бити случај. Поред тога, резултујући број кластера k мора бити одређен унапред.

Хијерархијски алгоритми [Fasulo99, Hearst96] имају за циљ формирање стабла чији чворови су подскупови полазног скупа тачака. Корен стабла је комплетан полазни скуп. Листови стабла су појединачне тачке скупа. Група дивизионих алгоритама формира хијерархију полазећи од корена стабла. Много чешћи, агломеративни алгоритми формирају хијерархију полазећи од листова. Начин рада ових алгоритама је следећи:

1. Формира се почетни скуп кластера тако што се свака тачка скупа налази у посебном кластеру.
2. Два међусобно најближа кластера спајају се у један кластер.
3. Корак 2 се понавља све док се критеријум заустављања не задовољи.

Варијанте агломеративних хијерархијских алгоритама међусобно се разликују по начину израчунавања међусобног растојања кластера и критеријуму заустављања. Неке варијанте, као што је *complete-link* [Leuski01], омо-

гуђавају формирање кластера који нису сферног облика. Међутим, временска зависност ових алгоритама је $O(n^2)$ или $O(n^3)$. Као критеријум заустављања поступка узима се или максималан број кластера или минимално растојање између коначних кластера. Ако је минимално растојање између кластера константно током свих сесија проналажења, густина кластера тежи константи [Leuski01].

Посебан проблем свим алгоритмима за формирање кластера представљају усамљене тачке појединим деловима посматраног простора (*outliers*). Формирање кластера над скуповима са великим бројем оваквих тачака је сложен проблем [BaezaYates02]. Поред две приказане групе алгоритама, за кластеровање докумената користе се и Кохоненове самоорганизујуће мапе (*self-organizing maps*, SOM) [He02] и алгоритми засновани на теорији графова [He99].

Кластеровање примењено на колекције текстуалних докумената спроводи се на два начина: (1) кластеровање целе колекције унапред и (2) кластеровање пронађених докумената у циљу олакшавања прегледа резултата (*browsing*). Поред тога, могуће је вршити и кластеровање упита, ради побољшања перформанси класичних метода проналажења [Voorhees95], или ради формирања FAQ (*frequently asked questions*) мапа и реформулације упита [Wen02].

Већа ефикасност метода кластеровања у односу на класичне методе рангирања је ретко потврђивана, и то само у ситуацијама за специфичне колекције докумената или у ситуацијама када се кластеровање користи за преглед докумената у комбинацији са класичним рангирањем [Hearst96]. У [Voorhees85] употреба кластеровања испитана је на неколико различитих колекција, али није установљен напредак у односу на класичне методе рангирања.

1.6. Класификација текст сервера

Формални модели претраживања докумената налазе своју софтверску имплементацију у оквиру текст сервера – софтверских система чија је основна функција да омогуће ефикасно претраживање база текстуалних докумената. Текстуални документи се, у различитим текст серверима, третирају на различит начин. Један вид класификације текст сервера узима у обзир доступност оригиналног текста у електронском облику. Класификацијом према доступности оригиналног текста системи за претраживање су подељени у две групе:

1. сервере који рукују оригиналним документима у електронском облику, процесирајући целокупан њихов садржај (*full text*) и
2. сервере који рукују метаподацима, тј. подацима који на одговарајући начин репрезентују карактеристике оригиналног документа.

Мотивација за руковање метаподацима уместо целокупним садржајем докумената лежи у знатно мањим меморијским захтевима за складиштење метаподатака и захтевима за њихово процесирање. Са друге стране, како метаподаци представљају само посредничку репрезентацију садржаја датог оригиналним документом, *full text* системи су у могућности да избегну непрецизност или непотпуност описа докумената помоћу метаподатака рукујући комплетним садржајем оригиналних докумената, што представља претраживање по форми (*form-based retrieval*) у складу са таксономијом датом у [Meghini01] а приказаном на почетку овог поглавља.

Други вид класификације текст сервера узима у обзир начин формирања *индекса*, података који се директно користе приликом процеса претраге. У односу на начин формирања индекса, слично класификацији датој у [Tudhope99] а приказаној на почетку овог поглавља, системи се деле на:

1. системе који користе ручно формиране индексе, тј. индексе чији садржај формирају квалификована лица и
2. системе који користе аутоматски формиране индексе.

Текст сервери који користе ручно формиране индексе омогућавају уграђивање експертског знања у садржај индекса. Са друге стране, формирање оваквих индекса подразумева учешће човека и није увек изводљиво или оправдано. Садржај аутоматски формираних индекса најчешће нема квалитет као код претходне групе јер је аутоматска екстракција семантике из текстуалних докумената изузетно сложен проблем.

Трећи начин класификације текст сервера односи се на модел докумената. На овај начин системи се деле на:

1. системе који рукују неструктурираним документима и
2. системе који рукују структурираним документима.

Библиотечки информациони системи за подршку пословања класичних библиотека рукују само репрезентацијама оригиналних докумената које чува библиотека. Садржај тих репрезентација формирају библиотекари. Репрезентације самих докумената су сложени ентитети са унутрашњом структуром. Са становишта претходно изнетих начина класификације система за претраживање текста, библиотечки информациони системи рукују метаподацима (уместо садржајем оригиналних докумената) који представљају ручно фор-

мирани индекс (садржај метаподатака, тј. описа оригиналних докумената креирају библиотекари) а сами метаподаци су структурирани текстуални документи.

1.7. Библиотечки софтверски систем БИСИС

Током развоја библиотечког информационог система БИСИС приказаног у монографији [Surla04b] креиран је одређен број софтверских решења за претраживање YUMARC записа као структурираних текстуалних докумената. У раду [Milosavljević97a] приказан је текст сервер за претраживање записа помоћу префикса имплементиран у *Oracle* окружењу и базиран на ускладиштеним PL/SQL процедурама *Oracle* система. Другачији приступ имплементацији текст сервера UNIMARC записа, заснован на употреби *Oracle ConText* модула за претраживање текста, приказан је у радовима [Milosavljević97b, Milosavljević98]. Спецификација и имплементација текст сервера који се ослања на релациону базу података и представља централну тему ове монографије приказана је у радовима [Milosavljević99a, Milosavljević99b, Milosavljević99d, Milosavljević04a]. Део функционалности текст сервера посебно намењен приступу преко веба приказан је у [Milosavljević00a, Milosavljević00b, Milosavljević04b, Milosavljević04c]. Употреба текст сервера у процесу узајамне каталогизације разматрана је у [Milosavljević00c, Milosavljević00d, Vidaković02].

Систем Мрежне дигиталне библиотеке докторских, магистарских и дипломских радова [Surla04a, Diglib] развијен на Универзитету у Новом Саду такође користи текст сервер БИСИС-а за потребе претраживања структурираних метаподатака о електронским документима. Употреба текст сервера у систему Мрежне дигиталне библиотеке приказана је у [Zarić04].

Даљи развој система БИСИС усмерен је, између осталог, и у правцу увођења мултимедијалне функционалности у претраживање дигиталних библиотека. XMIRS модел намењен претраживању структурираних мултимедијалних докумената [Milosavljević02b, Milosavljević02c, Milosavljević03a] верификован је на систему Мрежне дигиталне библиотеке [Milosavljević03b, Milosavljević04c, Milosavljević04d]. Овај модел представља уопштење текст сервера развијеног за UNIMARC записе као систем који рукује документима кориснички дефинисане структуре, при чему садржај елемената документа не мора бити искључиво текстуални, већ може припадати и неком другом типу медија, нпр. слике или видео записи.

Други правац развоја библиотечких информационих система лежи у усвајању XML технологије за репрезентацију, складиштење, претраживање и размену библиографске грађе. Репрезентација библиографске грађе помоћу XML докумената који описују структуре UNIMARC формата разматрана је у [Milosavljević00e, Zeremski02]. Формални систем за контролу квалитета библиографских записа репрезентованих XML документима развијен је у [Budimir04]. Архитектура текст сервера који се може употребити за руковање XML библиографским записима приказана је у [Kovčić00, Milosavljević99c]. У радовима [Škrbić04a, Škrbić04b, Škrbić04c] разматрано је коришћење XML *native* база података за управљање XML документима. Показано је да се све функције за складиштење, ажурирање и претраживање XML библиографских записа могу реализовати помоћу XML *native* базе података *Tamino*.

Поглавље 2

Репрезентација и претраживање библиографске грађе

Библиографска грађа представља описе штампаних докумената који су формирани по одређеном стандарду. Библиографска грађа се може поделити на више врста зависно од природе докумената које описује. Тако постоје описи монографских, серијских публикација, географских публикација и друго. За сваку врсту грађе постоје дефинисани стандарди који прописују како се библиографски опис штампаног документа формира и чува. У наредним одељцима дат је кратак приказ стандарда који се користе за рачунарску репрезентацију библиографске грађе, њено претраживање и подршку различитим писмима.

2.1. UNIMARC формат

UNIMARC (*Universal Machine Readable Cataloguing Format*) [UNIMARC, Gredley90] је стандард који је првенствено намењен за међународну размену библиографских података у машински читљивом облику између националних библиографских институција. Он дефинише ознаке садржаја (поља, индикаторе и потпоља) које се користе за формирање библиографских записа и њихов логички и физички формат. Обухвата монографске, серијске и картографске публикације, музику, звучне записе, слике, видео материјале а оставља место и за додавање рачунарских датотека.

UNIMARC је замишљен као стандардни формат за размену библиографске грађе, и не прописује начин обраде и чувања грађе унутар појединих библиографских институција. Међутим, практично све институције које су прихватиле UNIMARC као стандард за размену користе UNIMARC или неку његову једноставну модификацију у оквиру својих система како би поједноставиле конверзију из UNIMARC-а у интерни формат и обрнуто. У Србији се као стандард користи формат настао на основу формата UNIMARC, формата COMARC [COMARC] и захтева који су проистекли из пројекта БМ СНТИС

[Lazarević96]. У даљем тексту ће бити приказане основне карактеристике UNIMARC-а које су релевантне за потребе спецификације текст сервера који је предмет ове монографије, а све што ће бити изнето односи се у потпуности и на YUMARC. Разлика између UNIMARC-а и YUMARC-а постоји само у формату физичког записа слога на рачунару, као и у оном сегменту међународног UNIMARC стандарда који је и предвиђен за проширења на националном нивоу.

У даљем тексту ће се под именом *UNIMARC формат* подразумевати формат којег прописује стандард, а под именом *UNIMARC запис* – једна библиографска јединица дата у UNIMARC формату, односно један слог.

2.1.1. Структура UNIMARC записа

Сваки UNIMARC запис се састоји од коначног скупа *поља*, при чему не морају сва поља бити дефинисана. Уколико садржај неког поља није дефинисан, оно се не наводи у запису. Поља се једнозначно одређују троцифреним бројем. Садржај поља чине два *индикатора* и коначан скуп *потпоља*. Потпоља су једнозначно одређена једним алфанумеричким знаком. Аналогно пољима, ако садржај неког потпоља није дефинисан, оно се не наводи у скупу потпоља одговарајућег поља. Садржај потпоља је текст. Садржај једног потпоља је основна јединица информације (*data element*) коју поседује UNIMARC запис. Индикатори су намењени да ближе опишу садржаје дате у потпољима. Разликују се по положају у запису (*први* и *други*) и сваки је представљен једним знаком.

Нека поља су *поновљива*, тј. могу се наћи више пута у једном запису. Исто важи и за одређена потпоља у оквиру поља. Такође постоје поља која су *обавезна*, односно она која се морају наћи у запису. Слично томе постоје и обавезна потпоља у оквиру појединих поља.

Поља 421, 423 и 469 у случају обраде серијских публикација садрже само поновљиво потпоље 1 чији садржај представљају *секундарна поља*, тј. структуре комплетног поља које је садржано у потпољу 1.

YUMARC формат дефинише и концепт *потпотпоља*, која представљају структурирани садржај потпоља 996d, 997d i 998d. Потпотпоље је једнозначно одређено једним алфанумеричким знаком а садржај потпотпоља је текст.

Поља UNIMARC-а су сврстана у десет група (блокова). Блокови се идентификују првом цифром у ознаци поља. Следи списак дефинисаних блокова.

0 – Блок за идентификацију;
1 – Блок кодираних информација;
2 – Блок главног описа;
3 – Блок напомена;
4 – Блок за повезивање каталошких јединица;
5 – Блок сродних наслова;
6 – Блок садржајне анализе;
7 – Блок података о интелектуалној одговорности;
8 – Блок за међународну употребу;
9 – Блок за националну употребу.

Формална спецификација YUMARC формата погодна за имплементацију у софтверским системима заснованим на објектно-оријентисаном приступу дата је у [Pupovac04].

YUMARC је дефинисан тако да може да складишти текстове писане на различитим језицима. Саставни део спецификације је и дефиниција карактер сетова који се могу користити за складиштење текста; у питању су карактер сетови прописани ISO стандардима. Мешање карактер сетова је засновано на концепту текућег карактер сета, који одређује подразумевани карактер сет за све карактере, све до следећег маркера који означава промену карактер сета. Ако се у запису не налази ниједан маркер текућег карактер сета, подразумева се ISO 646.

Други начин записивања вишејезичких текстова омогућава *Unicode* стандард [Unicode], који дефинише карактер сет који може да обухвати 65536 (2^{16}) различитих знакова одједном. Овај стандард такође дефинише и више начина кодирања оваквог карактер сета. Најчешће коришћене варијанте су UTF-16 (*Unicode Transformation Format*), 16-битни запис са карактерима фиксне ширине погодан за коришћење у оквиру рачунарских програма, и UTF-8, 8-битни формат са карактерима променљиве ширине погодан за рачунарску размену података. Конверзија из *Unicode* текста у UNIMARC и обрнуто је могућа за она писма која су обухваћена обама стандардима. *Unicode* је данас de facto стандард за репрезентацију вишејезичног текста. Стандард ISO-IEC 10646 [ISO10646] дефинише идентичак карактер сет као и *Unicode*, али не укључује алгоритме везане за имплементацију и друге корисне информације.

2.1.2. Примери UNIMARC записа

У наставку, на листингу 2.1, приказан је пример дела UNIMARC записа. Записи ће бити дати у више редова, по један ред за свако поље записа. У оквиру поља су прво наведени индикатори (ако им вредност није дефинисана, приказани су знаком #), а затим и сва дефинисана потпоља. Идентификатори потпоља су дати у угластим заградама.

```
101 ## [a]scr
200 1# [a]Francuski jezik[e]za rudare i geologe[f]Svetlana Jevremović
210 ## [a]Beograd[c]Rudarsko-geološki fakultet[d]1992
215 ## [a]264 str.[c]ilustr.[d]24 cm
300 1# [a]U 2 prim.
320 ## [a]Bibliografija: str. 263-264
320 ## [a]Registar
606 1# [a]Francuski jezik[x]rudarsko-geološka struka[w]udžbenici za fakultet
675 ## [a]55+622]=40(075.8)
700 #1 [a]Jevremović[b]Svetlana
996 21 [q]2[v]d[w]a[y]g7\h19950801[2]RGF[d]f2\n58471[f]4033054
[o]19950809[3]30
996 21 [q]2[v]d[w]a[y]g7\h19950801[2]RGF[f]4033055[o]19950809[3]30
```

Листинг 2.1. Пример UNIMARC записа

Поље 700 дефинише примарну интелектуалну одговорност (тј. главног аутора) за дати документ. Када други индикатор поља 700 има вредност 1, тада се у потпољу *a* наводи презиме, односно део имена по коме се врши сортирање у извештајима, а у потпољу *b* други део имена, у овом случају само име аутора. У примеру се види да је аутор *Светлана Јевремовић*.

Поље 200 дефинише стварни наслов и податке о одговорности. У овом примеру главни стварни наслов (потпоље *a*) је *Француски језик*, поднаслов (потпоље *e*) је *за рударе и геологе*, а први податак о одговорности (потпоље *f*) је *Светлана Јевремовић*. Поље 101 дефинише језик документа. Потпоље *a* наводи шифровано име за српски језик (латиница). Слично претходном, на основу дефиниције UNIMARC формата се могу извући и остали потребни подаци о приказаној библиографској јединици.

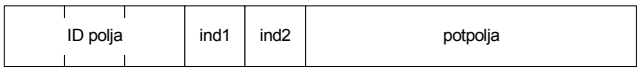
2.1.3. Физичка структура записа YUMARC формата

YUMARC формат садржи и дефиницију физичке структуре записа намењене електронској размени података. У овом одељку приказана је ова структура.

Један запис је представљен континуалним низом карактера. Овај низ се састоји из дефиниција појединих поља међусобно раздвојених делимитером поља. Као делимитер поља користи се карактер са кодом 30 децимално (1E хексадецимално). Делимитер поља не појављује на самом почетку и самом крају записа. У том смислу, један запис изгледа као на слици 2.1.

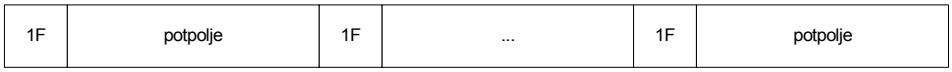
Слика 2.1. Физички формат записа: поља и делимитери поља

Даље ћемо посматрати само сегмент записа који представља једно поље, дакле сегмент ограничен делимитерима 1E. Прва три карактера овог сегмента представљају идентификатор поља. Наредна два карактера су вредности првог и другог индикатора, тим редоследом. Уколико вредност неког индикатора није дефинисана, на одговарајуће место се уписује знак бланко (децимални код 32). Остатак садржаја представља скуп потпоља. Слика 2.2 представља овај део структуре записа.



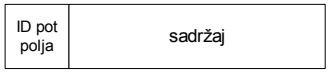
Слика 2.2. Физички формат записа: запис једног поља

Сегмент који садржи скуп потпоља користи карактер са кодом 31 децимално (1F хексадецимално) као делимитер потпоља. За разлику од делимитера поља, делимитер потпоља се налази и испред првог потпоља. Слично делимитеру поља, делимитер потпоља се не налази иза последњег потпоља. Слика 2.3 представља низ потпоља раздвојених одговарајућим делимитером.



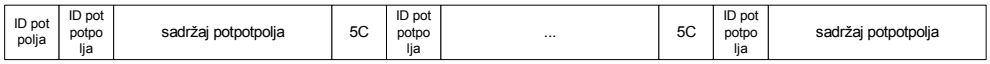
Слика 2.3. Физички формат записа: низ потпоља једног поља

Сегмент за опис потпоља користи први карактер за смештање идентификатора потпоља, док је преостали низ карактера (променљиве дужине) намењен за складиштење садржаја потпоља. Овај део структуре записа илуструје слика 2.4.



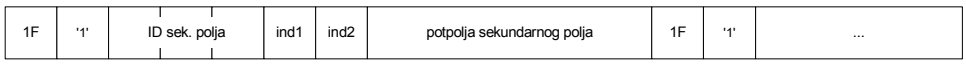
Слика 2.4. Физички формат записа: структура потпоља

Уколико посматрано потпоље уместо текста садржи потпотпоља, његова физичка структура је нешто другачија. Садржај потпоља је подељен на потпотпоља која су међусобно омеђена делимитером потпоља, карактером са кодом 92 (5C хексадецимално). Овај делимитер се не налази испред првог потпотпоља, нити иза последњег. Сегмент потпотпоља користи први карактер за смештање идентификатора потпотпоља, док је преостали низ карактера (променљиве дужине) намењен за складиштење садржаја потпотпоља. Слика 2.5 илуструје ову особину записа.



Слика 2.5. Физички формат записа: структура потпоља са потпотпољима

Секундарна поља се у запису смештају као садржај потпоља ‘1’ у неким пољима. Како је потпоље ‘1’ у том случају поновљиво, садржај потпоља ‘1’ се простире све до следеће дефиниције потпоља ‘1’ или до дефиниције новог поља записа. Слика 2.6 илуструје овакву ситуацију: иза делимитера потпоља налази се дефиниција потпоља ‘1’, чији садржај је секундарно поље, са својим троцифреним идентификатором, два индикатора и низом потпоља. Потпоље ‘1’ је поновљиво, тако да се ова секвенца може понављати.



Слика 2.6. Физички формат записа: секундарно поље

Треба обратити пажњу да се у оквиру секундарног поља не сме појавити потпоље са ознаком ‘1’, јер се такво потпоље сматра новим потпољем основног, а не секундарног поља. Заправо се потпоља секундарног поља ређају све до секвенце 1F (делимитер потпоља) ‘1’ (ознака потпоља), која се интерпретира као почетак новог потпоља основног поља, а не као ново потпоље текућег секундарног поља.

Најчешће се више записа пакује у једну датотеку где су записи међусобно раздвојени карактером LF (10 децимално) или комбинацијом карактера CR/LF (13/10).

Листинг 2.2 представља пример записа чија логичка структура је представљена у претходном одељку. Карактери са специјалним значењем (делимитери поља и потпоља) су представљени својим хексадецималним кодовима наведеним између витичастих заграда.

```
101  {1F}ascr{1E}2001 {1F}aFrancuski jezik{1F}eza rudare i geologe{1F}fSvetl
ana Jevremović{1E}210 {1F}aBeograd{1F}cRudarsko-geološki fakultet{1F}d1992{1
E}215  {1F}a264 str.{1F}cilustr.{1F}d24 cm{1E}3001 {1F}aU 2 prim.{1E}320  {1
F}aBibliografija: str. 263-264{1E}320  {1F}aRegistar{1E}6061 {1F}aFrancuski
jezik{1F}xrudarsko-geološka struka{1F}wudžbenici za fakultet{1E}675  {1F}a55
+622=40(075.8){1E}700 1{1F}aJevremović{1F}bSvetlana{1E}99621{1F}q2{1F}vd{1F}
wa{1F}yg7\h19950801{1F}2RGF{1F}df2\n58471{1F}f4033054{1E}99621{1F}q2{1F}vd{1
F}wa{1F}yg7\h19950801{1F}2RGF{1F}f4033055{1F}o19950809{1F}330
```

Листинг 2.2. Пример YUMARC записа са аспекта његове физичке структуре

2.1.4. YUMARC формат и различита писма

YUMARC је дефинисан тако да може да складишти текст дат различитим писмима. Саставни део спецификације је и дефиниција карактер сетова који се могу користити за складиштење текста; у питању су карактер сетови прописани ISO стандардима. Мешање карактер сетова је засновано на концепту текућег карактер сета, који одређује подразумевани сет за све карактере у посматраном стрингу, све до следећег маркера који означава промену карактер сета. Тренутно су дефинисани карактер сетови *латиница*, *ћирилица* и *грчки алфабет*. Ако се у запису не налази ниједан маркер текућег карактер сета, подразумева се латиница. Овај механизам мешања карактер сетова се донекле разликује од UNIMARC спецификације [UNIMARC94]. Маркери за ове карактер сетове су дати у табели 2.1.

Језик	**Хексадецимални код**
Латиница	1C
Ћирилица	1D
Грчки алфабет	1

Табела 2.1. Маркери за дефинисане карактер сетове

Пример текста *Petar Petrović Njegoš* представљеног латиницом је приказан на слици 2.7. У горњем реду су дате хексадецималне вредности карактера у стрингу који представља дати текст, а у доњем реду графички симболи који описују карактере. Назив маркера за избор карактер сета је дат у угластим заградама.

1C	50	65	74	61	72	20	50	65	74	72	6F	76	69	C9	20	4E	6A	65	67	6F	E9
[lat]	P	e	t	a	r		P	e	t	r	o	v	i	ć		N	j	e	g	o	š

Слика 2.7. Латинични текст представљен YUMARC распоредом

Исти овај текст представљен ћирилицом (*Петар Петровић Његош*) приказан је на слици 2.8.

1D	50	65	74	61	72	20	50	65	74	72	6F	76	69	C9	20	4E	6A	65	67	6F	E9
[cir]	П	е	т	а	р		П	е	т	р	о	в	и	ћ		Њ		е	г	о	ш

Слика 2.8. Ћирилични текст представљен YUMARC распоредом

Претходни примери илуструју једну особину YUMARC распореда изузетно значајну за српски језик: ако се из текста уклоне маркери за језик (у овом случају латиницу и ћирилицу), добија се истоветан стринг. Захваљујући специјалном кодирању ћириличних карактера Љ, Њ, и Џ двобајтним кодовима омогућено је директно пресликавање латиничног текста у ћирилични уз промену маркера за језик. Ова особина се може искористити за процес претраживања текста тако да се омогући проналажење погодака писаних латиницом када је упит постављен ћирилицом и обрнуто. Детаљи имплементације руковања вишејезичним текстом помоћу YUMARC и *Unicode* распореда приказани су у [Milosavljević02a].

2.2. Концепт префикса и претраживање записа

Претраживање библиографске грађе, тј. UNIMARC записа, не врши се коришћењем претходно описаних концепата поља и потпоља, већ префикса. Посматрано са овог аспекта, запис се састоји из скупа префикса. Сваки префикс је једнозначно одређен називом од два знака. Садржај префикса је текст. Један префикс може имати више садржаја у запису. Префикси који нису дефинисани се не наводе. Основна јединица информације у оваквом погледу на запис је један садржај једног префикса.

Списак постојећих префикса, као и пресликавање UNIMARC концепата (скупа поља и потпоља) на скуп префикса су дефинисани у [Surla03]. Како ово пресликавање није бијективно, инверзно пресликавање не постоји.

Запис са листинга 2.1 би, са становишта префикса, имао изглед као на листингу 2.2.

```
AU Jevremović, Svetlana
DA 19950809
KW Francuski jezik
KW rudarsko-geološka struka
KW udžbenici za fakultet
LA scr
ND g7\h19950801
ND g7\h19950801
PP Beograd
PU Rudarsko-geološki fakultet
PY 1992
SG f2\n58471
SU za rudare i geologe
TI Francuski jezik
TN Francuski jezik
TN rudarsko-geološka struka
TN udžbenici za fakultet
```

Листинг 2.2. Пример записа посматраног преко префикса

Претраживање базе записа представља проналажење оних записа који задовољавају дати упит, тј. критеријум претраживања. Упит је *квалификован* уколико се наведе у ком префиксу се тражи дати израз. У другом случају каже се да је упит *неквалификован*. Префикси су подељени у два дисјунктна скупа: *базни* и *додатни*. Базни префикси су они префикси који се претражују приликом обраде неквалификованих упита.

Појам *речи* унутар садржаја префикса се дефинише као низ знакова који је са обе стране ограничен *делимитерима речи*. Делимитери речи су знаци који нису саставни део речи, а налазе се непосредно поред дате речи у низу знакова који чине садржај префикса. Аналогно се дефинише и појам *реченице*: она представља низ речи који је ограничен *делимитерима реченице*. *Фраза* представља низ суседних речи.

Претраживање записа помоћу концепта префикса треба да има следеће могућности:

- претраживање се врши по речима или фразама, уз коришћење џокер-знакова;
- претраживање може бити квалификовано, тј. може се навести у ком префиксу се тражи дати садржај; уколико се квалификација не наведе, претраживање се врши по свим *базним* префиксима;
- од елементарних упита, који се састоје од једне тражене речи (терма), могуће је саставити сложене коришћењем логичких и близинских оператора;

- за сложене упите може се добити информација о резултатима претраге за подупите;
- могуће је добити листу свих префикса који почињу датим изразом.

За формирање упита може се користити синтакса упитног језика *Dialog* [Dialog]. *Dialog* упитни језик је производ фирме *Dialog Corporation*. Фирма *Dialog* се бави пројектовањем и одржавањем специјализованих база докумената. Приступ овим базама ради претраживања је омогућен кроз специјално корисничко окружење. Претраживање је засновано на концепту префикса. Интерни формат смештања записа у бази није јавно доступан.

Ово окружење је заправо интерпретер команди *Dialog* језика предвиђен за интерактиван рад. Након пријављивања на систем, корисник уноси команду и добија резултат њеног извршавања. Систем је тада спреман да прими нову команду.

Број команди које поседује *Dialog* систем је релативно велики. Оне омогућавају претраживање, листање резултата, додатно руковање резултатима, подешавање корисничког окружења итд. За текст сервер који је тема ове монографије интересантне су команде SELECT и EXPAND које омогућавају унос упита за претраживање. Комплетан приказ *Dialog* корисничког окружења и команди интерпретера је дат у [Dialog].

Спецификација команде SELECT дефинише и синтаксу упита. Упити се састоје од термова повезаних операторима. Елементарни упит се састоји од само једног терма. Терм представља реч која се тражи у записима. Речи се могу навести са џокер-знаком ‘?’ на крају који замењује било који текст. Следи пример једног елементарног, неквалификованог упита:

```
select eclipse
```

Dialog разликује префиксе који су индексирани по речима и по фразама. Фраза се овде дефинише слично као реч, као текст омеђен *делимитерима фразе*. Уколико се претражује по префиксу који је индексиран по фразама, мора се навести потпуна фраза, као у следећем примеру:

```
select au=preisendorfer, rudolph w.
```

Претходни пример илуструје и начин квалификације упита. У овом случају претрага се врши по префиксу AU (аутор).

Сложени упити се састављају од елементарних који се повезују операторима у инфиксној нотацији. Сви оператори су бинарни. Могуће је користити заграде за управљање редоследом извршавања операција. *Dialog* дефинише следеће операторе:

- AND – израчунава пресек скупова погодака свог левог и десног операнда;
- OR – израчунава унију скупова погодака свог левог и десног операнда;
- NOT – израчунава скуповну разлику скупова погодака свог левог и десног операнда;
- T – користи се за претраживање имена хемијских једињења; налази записе који садрже једињење чији је један део имена дат левим операндом, а други десним;
- W – захтева да изрази дати левим и десним операндом буду суседни, при чему је битан редослед;
- N – захтева да изрази дати левим и десним операндом буду суседни, при чему редослед није битан; може се користити и за проналажење идентичних израза;
- S – захтева да изрази дати левим и десним операндом буду у истом потпољу (дефиниција потпоља зависи од конкретне базе података); у *full-text* претраживању захтева да изрази буду у истом пасусу;
- L – захтева да изрази дати левим и десним оператором буду у истој дескрипторској јединици (дефиниција дескрипторске јединице зависи од конкретне базе података);
- F – захтева да се изрази дати левим и десним операндом налазе у истом пољу (дефиниција поља зависи од конкретне базе података).

Следе примери сложених упита.

```
select moon or lunar
```

Претходни упит иницира претраживање по свим базним префиксима (ниједан терм није квалификован) где се тражи бар једна од речи *moon* или *lunar*. Следећи упит тражи записе који садрже речи *long* и *range* као суседне, где је редослед битан:

```
select long (w) range
```

Уколико се тражи фраза која садржи неку од резервисаних речи *Dialog*-а, фраза се мора навести између наводника, као у следећем примеру:

```
select "climate and glaciation"
```

Резултат извршавања SELECT наредбе је извештај о броју докумената у бази који задовољавају дати упит, али и о броју докумената који задовољавају сваки од подупита. Следи пример који илуструје унос једне SELECT наредбе и исписивање резултата њеног извршавања.

```
?select moon or lunar
```

```
      1661  MOON
      2044  LUNAR
  S3  2741  MOON OR LUNAR
```

Коначном резултату из горњег примера је додељена ознака S3 и та ознака се може даље користити да означи скуп докумената добијен претходном SELECT наредбом. На пример:

```
?select s3 and eclipse
      2741  S3
       415  ECLIPSE
  S4   207  S3 AND ECLIPSE
```

У претходним примерима само је коначном резултату додељивана посебна ознака која омогућава коришћење добијеног резултата у даљим упитима. Међутим, могуће је и свим резултатима подупита доделити овакве ознаке. Ово се постиже посебном варијантом SELECT наредбе, SELECT STEPS. Следи пример:

```
?select steps sun or solar
  S5   5255  SUN
  S6  49461  SOLAR
  S7  52119  SUN OR SOLAR
```

Наредба EXPAND је намењена за листање садржаја текст индекса. Она приказује списак термова који се налазе у текст индексу. Наредба EXPAND прима само један параметар и то неквалификовани терм. Претрага се врши по свим префиксима, без обзира да ли су базни или додатни. Резултат извршавања наредбе EXPAND је списак са 12 ставки, где је трећа ставка дати терм (претходне две ставке се налазе „испред“ у индексу; индекс је сортиран по абецеди). Следи пример једне EXPAND наредбе:

```
?expand au=einstein, a
Ref   Items  Index-term
E1        7  AU=EINSPRUCH, NORMAN G.
E2        1  AU=EINSTEIN KRAHN, DOROTHEE
E3        0 *AU=EINSTEIN, A
E4       33  AU=EINSTEIN, A.
E5      229  AU=EINSTEIN, ALBERT
E6        1  AU=EINSTEIN, DAVID M.
E7        2  AU=EINSTEIN, GILLES O.
E8        1  AU=EINSTEIN, H. E.
E9        7  AU=EINSTEIN, J.
E10       1  AU=EINSTEIN, J. R.
E11       2  AU=EINSTEIN, S.
E12       1  AU=EINTERT, DAGMAR
```

Кроз овако добијени списак може се кретати према почетку и према крају одговарајућим командама. У приказаном примеру види се да се тражени терм не појављује ниједном у индексу, због разлике у записивању иницијала.

Резултати добијени наредбом EXPAND могу се даље користити у наредби SELECT уз помоћ коришћења меморисаних међурезултата. Наредба *select e5 and e7* ће вратити пресек скупова погодака именованих са E5 и E7 претходно добијених EXPAND наредбом.

Поглавље 3

Моделирање текст сервера UNIMARC записа

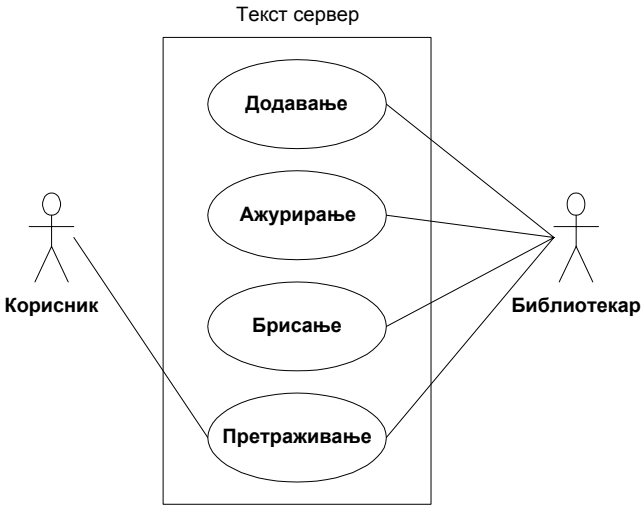
Текст сервер за UNIMARC записе представља, према класификацији датој у одељку 1.6, систем за руковање метаподацима који описују садржај оригиналног документа. Метаподаци су садржани у оквиру самог UNIMARC записа који се третира као структурирани текстуални документ. Садржај метаподатака формирају стручна лица – библиотекари, па се може рећи и да се овде ради о систему са ручним формирањем индекса од стране експерата за дату област. Индекс формиран на овакав начин има сврху да обезбеди прецизно и ефикасно претраживање колекције оригиналних докумената које чува дата библиотека. У овом поглављу је представљена спецификација текст сервера UNIMARC записа који је саставни део библиотечког информационог система БИСИС. Претраживање текста у овом систему има карактер претраживања подаката (*data retrieval*), онако како је то дефинисано на почетку првог поглавља. Самим тим, поређење текстуалних фрагмената подразумева методе за егзактно поређење алфанумеричких стрингова. Поред тога, модел претраживања текст сервера подразумева бинарно изражавање сличности документа са упитом, тако да рангирање докумената по релевантности није присутно.

3.1. Функције текст сервера

Основна намена текст сервера је да омогући складиштење и претраживање библиотечке грађе која се обрађује и чува у датом библиотечком систему. Овакав текст сервер користе две класе корисника: библиотекари и (спољни) корисници. Библиотекари имају приступ свим функцијама текст сервера које укључују додавање нових докумената у индекс, ажурирање или уклањање постојећих докумената и претраживање. Библиотекари текст серверу приступају са клијената који се налазе у локалној рачунарској мрежи. Корисници имају право само да претражују индекс текст сервера, при чему

им је омогућен приступ било из локалне рачунарске мреже (интранета), било са Интернета. Ови захтеви у погледу коришћења текст сервера представљени су дијаграмом случајева коришћења са слике 3.1.

Документи којима рукује текст сервер су UNIMARC записи. Претраживање базе записа се врши коришћењем модификованог *Dialog* упитног језика [Dialog] приказаног у претходном поглављу. Основне карактеристике претраживања су следеће.



Слика 3.1. Дијаграм случајева коришћења текст сервера

Расположиве су команде SELECT и EXPAND дефинисане у *Dialog*-у. Команда SELECT као параметар има текст који представља упит. Упит може да садржи следеће операторе:

- AND – израчунава пресек скупова погодака свог левог и десног операнда;
- OR – израчунава унију скупова погодака свог левог и десног операнда;
- NOT – израчунава скуповну разлику скупова погодака свог левог и десног операнда;
- [F] – захтева да се изрази дати левим и десним операндом налазе у истом префиксу;
- [S] – захтева да се изрази дати левим и десним операндом налазе у истој реченици;

- [Wn] – захтева да изрази дати левим и десним операндом буду удаљени највише *n* речи један од другог; ако се *n* не наведе, подразумева се 1 (суседне речи).

Приоритет оператора је следећи (оператори су наведени у опадајућем редоследу својих приоритета):

[Wn]

[S]

[F]

AND, OR, NOT

На редослед извршавања операција може се утицати коришћењем заграда на уобичајен начин.

За потребе физичког смештања података, текст сервер се ослања на неки од постојећих система за управљање базама података (СУБП). Подразумева се да је СУБП заснован на релационом моделу података који је данас најчешћи у употреби. Текст сервер је пројектован тако да у што већој мери буде независан од конкретног СУБП. Како сви комерцијално расположиви СУБП имају одређене специфичности које се огледају у начину њиховог коришћења, текст сервер има могућност да се те специфичности искористе са циљем постизања квалитетних перформанси.

3.2. Модел података текст сервера

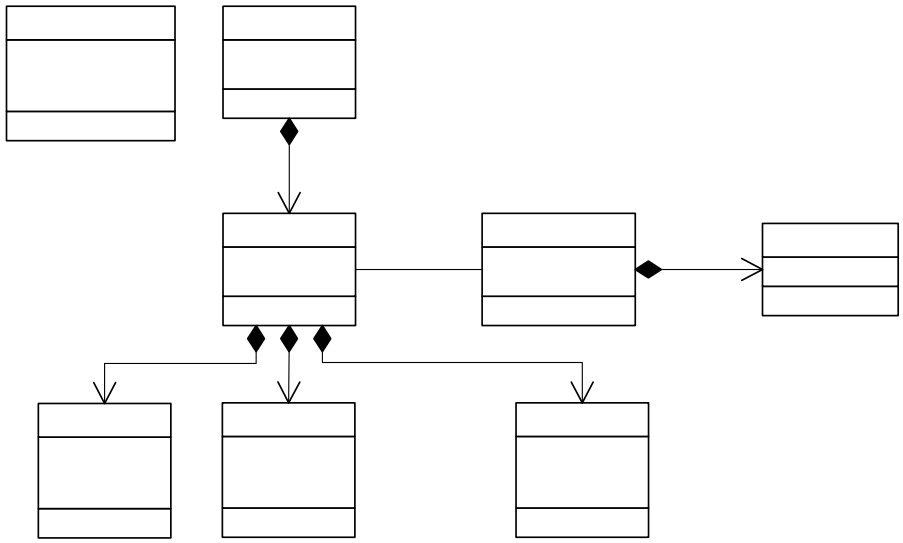
Модел података текст сервера је предвиђен за имплементацију на неком од комерцијално доступних СУБП заснованих на релационом моделу података. Спецификација која ће овде бити изложена не зависи од конкретног СУБП. На слици 3.2 приказан је дијаграм класа који представља модел података текст сервера.

Класа *Document* представља документ којим рукује текст сервер, тј. UNIMARC запис. Њени атрибути су *docID* (интерни идентификатор документа) и *content* (садржај документа, тј. сам запис).

Класа *Prefix* представља префикс који се користи приликом претраживања. Њени атрибути су *prefName* (назив префикса, идентификатор) и *prefType* (тип префикса – базни или додатни, онако како су дефинисани у уводном одељку).

Класа *PrefixMap* је намењена за смештање пресликавања префикса на потпоља UNIMARC формата. Једини атрибут је *subfield* (потпоље које се пресликава на дати префикс).

Класа *PrefixContent* представља један садржај неког од префикса који припадају датом документу. Њени атрибути су *prefID* (идентификатор), и *content* (садржај префикса).



Слика 3.2. Модел података текст сервера

Класе *Pref_AB*, *Pref_AU* итд. до *Pref_Y2* представљају речи које припадају одговарајућем префиксу (AB, AU, итд. до Y2). Свака од њих има исте атрибуте: *wordPos* (редни број речи у префиксу), *sentPos* (редни број реченице у којој се налази дата реч) и *content* (садржај, тј. сама реч).

Класа *Delimiters* представља конфигурацију парсера текста. Њени атрибути су *configID* (идентификатор), *wordDelims* (карактери који представљају делимитере речи) и *sentDelims* (карактери који представљају делимитере реченице). Подразумева се да су делимитери реченице уједно и делимитери речи.

Како су данас најчешће у употреби СУБП засновани на релационом моделу података, претходно описани дијаграм класа је искоришћен за генерисање шеме релационе базе података. Шема је генерисана у оквиру одговарајућег CASE алата. Шеме релација добијене на овај начин приказане су на

листингу 3.1. Курзивом су дати називи атрибута који улазе у састав примарног кључа.

```
DELIMITERS(config_id, word_delims, sent_delims)
PREFIXES(pref_name, pref_type)
PREFIX_MAP(pref_name, subfield)
DOCUMENTS(doc_id, content)
PREFIX_CONTENTS(doc_id, pref_id, pref_name, content)
PREF_AB(doc_id, pref_id, word_pos, sent_pos, content)
PREF_AU(doc_id, pref_id, word_pos, sent_pos, content)
…
PREF_Y2(doc_id, pref_id, word_pos, sent_pos, content)
```

Листинг 3.1. Шеме релација текст сервера

3.3. Подсистем за руковање записима

Руковање UNIMARC записима обухвата следеће операције: додавање, уклањање и ажурирање записа. На слици 3.3 је приказан дијаграм класа овог подсистема.

Интерфејс *TextServerI* представља везу овог система са корисником. Он дефинише методе које су кориснику (тј. клијент-апликацији) доступне. Класа *TextServer* имплементира овај интерфејс. Више детаља о разлозима зашто се комуникација са корисником одвија преко интерфејса (а не директно са главном класом) дато је у поглављу о имплементацији.

Класа *Config* је намењена за чување података о конфигурацији текст сервера. Конфигурацију чине параметри текст сервера који зависе од конкретне инсталације програма (адреса сервера са базом података, тип сервера, кориснички налог на серверу итд.).

Подаци о конфигурацији програма се чувају у датотеци која има формат INI датотека познатих из фамилије *Windows* оперативних система. INI датотеке су текстуалне датотеке подељене на више секција, где су у свакој секцији дефинисани одговарајући параметри. Пример једне INI датотеке дат је на листингу 3.2.

Класа *Converter* садржи методе за међусобну конверзију различитих кодних распореда. Интерфејс *DocumentManager* дефинише све операције које остатак система упућује према серверу базе података. Његова намена је да одвоји онај део система који зависи од конкретног употребљеног СУБП од остатка система. Овај интерфејс имплементирају различите класе намењене раду са СУБП различитих произвођача. На дијаграму је приказана класа

OracleDocumentManager која имплементира овај интерфејс у смислу дефинисања операција које се врше над базом података којом управља СУБП компаније *Oracle*. Систем може да поседује више оваквих класа. Избор конкретне класе која имплементира дати интерфејс (а биће коришћена у одређеној инстанци програма) врши се приликом стартовања програма. Тада се анализира INI датотека и на основу типа СУБП који је у њој наведен бира се одговарајућа класа. На дијаграму је, као пример још једне класе која имплементира интерфејс *DocumentManager* приказана класа *SQLServerDocumentManager*.

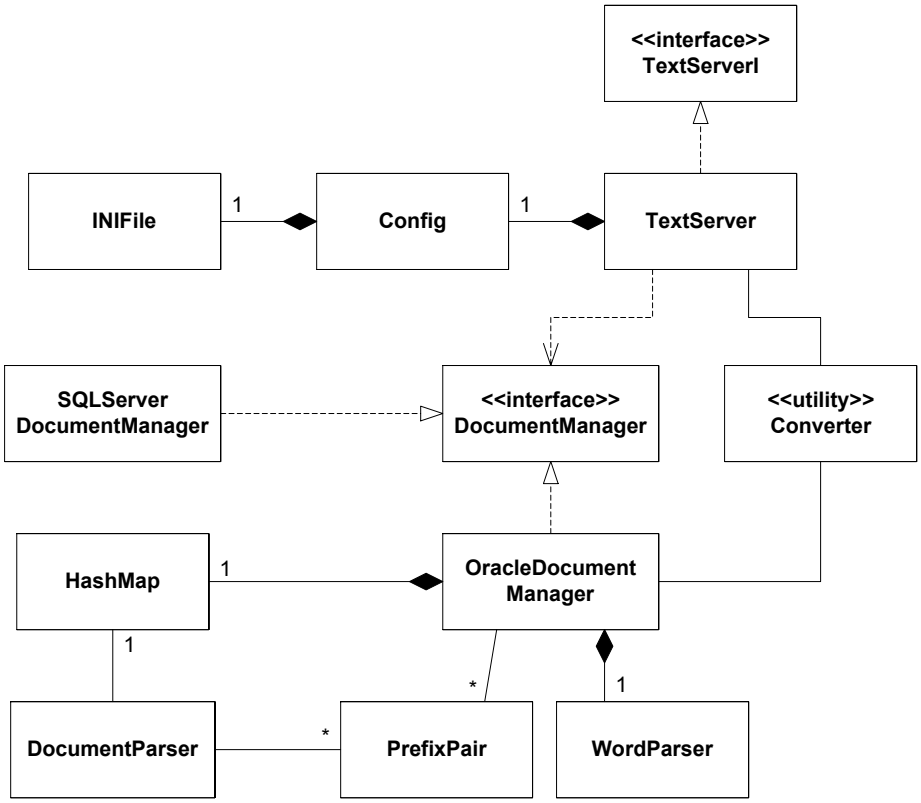
```
[database]
dbtype = oracle  ; tip baze: {oracle,sqlserver,informix}
jdbc = oracle.jdbc.driver.OracleDriver
connection = jdbc:oracle:thin:@sunce.tmd.ns.ac.yu:1526:BIS8
username = mbranko
password = D23F44G

[config]
parserID = 1
hexconv  = 0
```

Листинг 3.2. Пример INI датотеке

Класа *DocumentParser* је намењена за конверзију UNIMARC записа у префиксе. Класа *PrefixPair* представља уређени пар (префикс, садржај). Низ објеката ове класе представља парсиран UNIMARC запис. Класа *WordParser* служи за парсирање текста на речи и реченице у складу са дефинисаним делимитерима. Користи се за парсирање префикса у процесу индексирања. Класа *HashMap* садржи мапирање потпоља на префиксе.

Процес додавања записа у базу се састоји од конверзије UNIMARC записа у префиксе, парсирања добијених префикса на речи и њиховог смештања у базу. Процес се иницира позивом методе *insertDocument* интерфејса *DocumentManager*, односно неке од класа које имплементирају овај интерфејс. Процес конверзије записа у префиксе се иницира позивом методе *parse* класе *DocumentParser*. Уклањање записа из базе се своди на уклањање одговарајућих *n*-торки у релацијама базе података. Ажурирање записа је заправо секвенца коју чине његово брисање и поновно додавање (са новим садржајем).



Слика 3.3. Дијаграм класа подсистема за руковање записима

3.4. Подсистем за претраживање записа

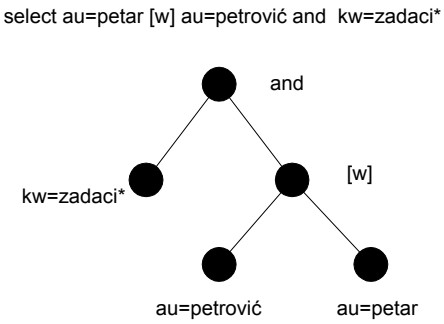
3.4.1. SELECT упити

Процес претраживања базе записа у највећој мери одређују два захтева: рад са упитима формираним по модификованој *Dialog* синтакси и потреба за познавањем резултата претраге по свим елементарним упитима који чине дати упит. У монографији [Lazarević96] је приказан начин обраде упита који је и овде усвојен уз одређене модификације.

Процес обраде SELECT упита има следеће фазе:

1. *Лексичка анализа*. У овој фази се добијени упит парсира и формира се листа чији су елементи термови претраге (квалификовани или неквалификовани), оператори и евентуалне заграде. Такође се открива и евентуална употреба непостојећих префикса.
2. *Синтаксна и семантичка анализа*. На основу претходно генерисане листе, формира се синтаксно стабло упита које представља и план извршавања упита. Поред тога, врши се и контрола исправности употребе појединих оператора.
3. *Интерпретација плана извршавања упита*. На основу креираног синтаксног стабла врше се одговарајуће претраге у бази података и израчунава се коначан резултат претраживања.

Пример једног SELECT израза и одговарајућег синтаксног стабла дат је на слици 3.4. У примеру се траже записи чијем се префиксу AU налази реч *petar* одмах поред речи *petrović*, а истовремено се у префиксу KW налази реч која почиње са *zadaci*. На основу датог упита формирано је стабло које представља и план извршавања упита: стабло се рекурзивно обилази и за сваки чвор се израчунава скуп погодака. Скуп погодака корена представља резултат извршавања упита. Скупови погодака у листовима стабла представљају резултате извршавања елементарних подупита.



Слика 3.4. Пример синтаксног стабла упита

За потребе овог текст сервера под поготком се подразумева следећа уређена четворка: (*doc_id*, *pref_id*, *word_pos*, *sent_pos*).

Елементи уређене четворке су дефинисани на следећи начин:

- *doc_id*: идентификатор документа,
- *pref_id*: идентификатор префикса,
- *word_pos*: редни број речи у префиксу и

– *sent_pos*: редни број реченице у којој се реч налази.

Скуп погодака у листовима стабла се добија директно из базе података, постављањем одговарајућег SQL упита. Један од SQL упита који би био постављен у горњем примеру је:

```
SELECT doc_id, pref_id, word_pos, sent_pos
  FROM pref_AU
  WHERE content = 'PETROVIĆ';
```

Из примера се види да је одговарајући SQL упит могуће једноставно генерисати – једини променљиви елементи у изразу су назив табеле у којој се врши претрага и стринг који се тражи.

У чворовима стабла који нису листови налазе се оператори. За ове чворове се одговарајући скуп погодака израчунава на основу скупова погодака два подређена чвора. Начин израчунавања резултујућег скупа погодака зависи од врсте оператора.

За оператор AND резултујући скуп се рачуна као пресек скупова погодака подређених чворова, при чему су два поготка једнака ако им је једнак *doc_id*.

За оператор OR резултујући скуп се рачуна као унија скупова погодака подређених чворова, при чему су два поготка једнака ако им је једнак *doc_id*.

За оператор NOT резултујући скуп се рачуна као разлика скупова погодака чвора који одговара левом операнду и чвора који одговара десном операнду, при чему су два поготка једнака ако им је једнак *doc_id*.

За оператор [F] резултујући скуп се рачуна као пресек скупова погодака подређених чворова, при чему су два поготка једнака ако су им једнаки одговарајући *doc_id* и *pref_id*.

За оператор [S] резултујући скуп се рачуна као пресек скупова погодака подређених чворова, при чему су два поготка једнака ако су им једнаки одговарајући *doc_id*, *pref_id* и *sent_pos*.

За оператор [Wn] резултујући скуп се рачуна као пресек скупова погодака подређених чворова, при чему су два поготка једнака ако су им једнаки одговарајући *doc_id* и *pref_id*, а апсолутна вредост разлике *word_pos* је мања или једнака *n*.

Из описаног начина израчунавања резултата се види да се израчунавање скупа погодака "не-лист" чворова стабла своди на скуповне операције, при чему критеријум једнакости два елемента скупа зависи од коришћеног

оператора. Табела 3.1 сумира претходно изложене операторе и одговарајуће критеријуме једнакости погодака.

3.4.2. Коректност SELECT упита

На основу дефиниције приказаних оператора види се да није могуће комбиновати операторе у оквиру упита на произвољан начин зато што се приликом израчунавања неког међурезултата одређени подаци одбацују. Следи пример једног коректног упита:

```
select au=petar [w] au=petrović and kw=zadaci          (*)
```

Оператор	Критеријум једнакости	Скуповна операција
AND	doc_id1 = doc_id2	пресек
OR	doc_id1 = doc_id2	унија
NOT	doc_id1 = doc_id2	разлика
[F]	doc_id1 = doc_id2 pref_id1 = pref_id2	пресек
[S]	doc_id1 = doc_id2 pref_id1 = pref_id2 sent_pos1 = sent_pos2	пресек
[Wn]	doc_id1 = doc_id2 pref_id1 = pref_id2 |word_pos1 – word_pos2| <= n	пресек

Табела 3.1. Преглед коришћених операција приликом израчунавања скупова погодака

Приликом обраде претходног упита прво ће се обрађивати термови са елементарним упитима, затим ће се рачунати скуп погодака за оператор [W] и, потом, скуп погодака за оператор AND (стабло овог упита је дато на слици 3.4). Овде је важно приметити да је оператор [W] употребљен *пре* оператора AND. Сада следи пример једног некоректног упита:

```
select au=petar [w] (au=petrović and kw=zadaci)        (**)
```

Уколико би овакав упит био допустив, тада би се прво израчунавао скуп погодака за оператор AND па затим за оператор [W]. Приликом рачунања скупа погодака за оператор AND погоци са једнаким елементом *doc_id* би се сматрали једнаким, и само један од њих би ушао у резултујући скуп (дупликати у скупу нису дозвољени). Тај елемент, који је у резултујући скуп ушао као представник погодака са једнаким *doc_id,* не мора касније (приликом рачунања скупа погодака за [W]) бити једнак неком елементу из другог скупа погодака, што не гарантује да неки други елемент који је прет-

ходно елиминисан не би био једнак. Овај проблем ће сада бити илустрован на примеру.

Нека су скупови погодака за елементарне упите следећи:

терм	погоци
au=petar	(1, 1, 2, 1)
au=petrović	(1, 1, 1, 1)
kw=zadaci	(1, 2, 1, 1)

Како се, за упит (**), прво израчунава скуп погодака за оператор AND, (дакле *au=petrović and kw=zadaci*) рецимо да је у пресечни скуп ушао погодак (1, 2, 1, 1). Тада ће скуп погодака за оператор [W] бити празан скуп, јер (1, 2, 1, 1) није једнак поготку (1, 1, 2, 1) добијеном из терма *au=petar*. У случају да је у први пресечни скуп ушао погодак (1, 1, 1, 1), тада укупан резултујући скуп не би био празан.

Због претходно описаног проблема уводи се следеће ограничење: *није дозвољено да било који потомак чвора у стаблу има категорију већу од категорије претка*. Категорија се дефинише као додатни атрибут чвора и представљен је целим бројем. Вредност категорије зависи од типа чвора. Табела 3.2 наводи врсте чворова и њихове категорије.

тип чвора	категорија
and, or, not	1
[F]	2
[S]	3
[Wn]	4
list	5

Табела 3.2. Категорије чворова синтаксног стабла

Посебан проблем остаје случај узастопног коришћења истог оператора. Упит као што је:

```
select A and B and C
```

је дозвољен. Резултат упита је пресек скупова погодака, где је у оба случаја за једнакост елемената узиман критеријум за AND оператор. Међутим, формално је дозвољен и следећи упит:

```
select A [w] B [w] C                                    (***)
```

Резултат извршавања овог упита не мора бити познат, бар за претраживање онакво какво је до сада дефинисано. Проблем представља избор одговарајућег елемента који ће ући у пресечни скуп. Нека су скупови погодака за елементарне подупите претходног упита следећи:

терм	погоци
A	(1, 1, 1, 1)
B	(1, 1, 2, 1)
C	(1, 1, 3, 1)

Како се прво рачуна скуп погодака за израз **A [w] B**, у резултујући скуп може ући или погодак (1,1,1,1) или погодак (1,1,2,1). Ако се изабере први погодак, укупан резултат упита ће бити празан скуп (јер погодак (1,1,1,1) неће проћи поређење са поготком (1,1,3,1)), а ако се изабере други погодак, укупан резултат упита неће бити празан скуп.

Због овог проблема се, за израчунавање резултујућег скупа погодака за оператор [Wn], уводи додатно правило: *у резултујући скуп погодака улази онај погодак (од два једнака) који припада скупу погодака десног операнда*.

На овај начин је решен и проблем постављања упита типа (***), што је изузетно важно, јер упити овог типа представљају, заправо, претраживање по фразама. Узастопно коришћење свих других оператора осим [W] не ствара овај проблем, па се ово додатно правило на њих не мора односити.

3.4.3. EXPAND упити

Друга врста упита који се могу поставити текст серверу су EXPAND упити. Овакви упити, као што је раније речено, имају једноставну синтаксу: упит се састоји од само једног терма (квалификованог или неквалификованог). Резултат извршавања је скуп парова (број, префикс) где је *префикс* сваки префикс у бази који почиње датим термом, а *број* представља број појављивања тог префикса у бази података. Пример једног EXPAND упита би био:

```
expand au=petr
```

Резултат овог упита би били они садржаји префикса AU који почињу термом *petr* (на пример, *Петровић*, *Петрановић*, итд.). Овакав упит се, на основу шема релација приказаних на листингу 3.1 може конвертовати у следећи SQL исказ:

```
SELECT COUNT(*), a.content
  FROM Prefix_contents a, pref_AU b
  WHERE b.content LIKE 'petr%'
    AND b.word_pos = 1
    AND b.pref_id = a.pref_id
  ORDER BY a.content;
```

Из примера се види да је једини промењљиви део овог SQL исказа стринг који се тражи (*petr%*) и назив табеле са речима (*pref_AU*). Генерисање одговарајућег SQL упита на основу датог EXPAND упита је релативно једноставно.

3.4.4. Класе подсистема за претраживање записа

На слици 3.5 представљен је дијаграм класа подсистема за претраживање записа. Интерфејс *TextServerI* и класа *TextServer* представљају везу са корисником и већ су приказане на дијаграму класа подсистема за руковање записима (слика 3.3). Осим метода намењених руковању записима, овај интерфејс и класа дефинишу методе које се користе током претраживања. Интерфејс *DocumentManager* се такође појављује и у овом дијаграму. Он омогућава приступ конкретном коришћеном СУБП. Има дефинисане методе које се користе приликом претраживања базе података.

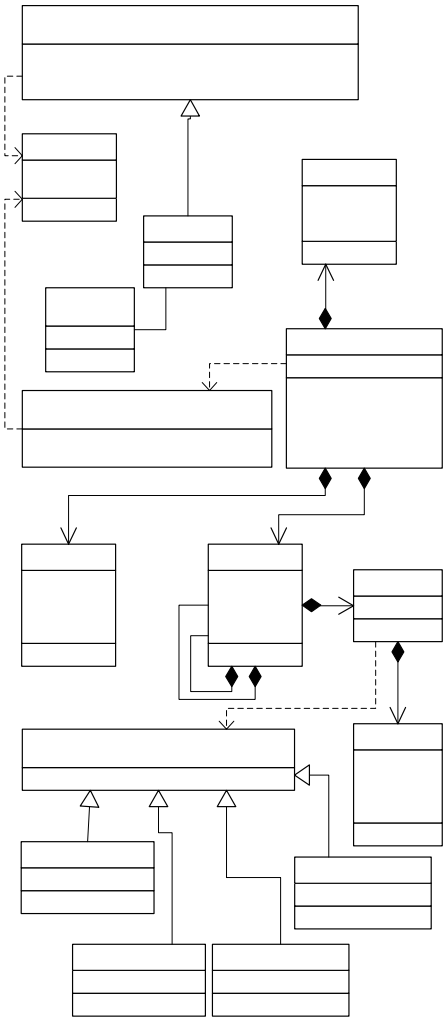
Један од главних атрибута класе *TextServer*, свакако најважнији за претраживање је атрибут класе *ExpressionTree*. Ова класа представља стабло претраге и дефинише методе за постављање упита и преузимање резултата. Класа *ListElem* се користи у процесу обраде упита за креирање листе термова, оператора и заграда који чине упит.

Класа *TreeNode* представља чвор стабла претраге. Сваки објекат класе *TreeNode* је повезан са још три чвора – непосредним родитељем и непосредним потомцима. Корен стабла је онај чвор чији родитељ није дефинисан. Један од атрибута класе *TreeNode* је и атрибут класе *HashSet* која је намењена за чување скупова погодака и имплементира потребне скуповне операције над њима. Класа *SearchHit* представља један погодак у процесу претраживања.

Интерфејс *BinaryPredicate* дефинише само једну методу. Она се користи за поређење елемената скупа унутар класе *HashSet*, приликом израчунавања скуповних операција. Овај интерфејс имплементира више класа. Класа *BooleanComparator* је намењена за поређење елемената приликом обраде логичких оператора (*and*, *or* и *not*). Класа *DistanceComparator* се користи приликом обраде оператора [Wn]. Класа *SentenceComparator* се користи приликом обраде оператора [S]. Класа *FieldComparator* се користи приликом обраде оператора [F].

Класа *SubqueryHit* представља један извештај о резултату претраге за елементарни подупит. Класа *ExpandPair* представља једну ставку извештаја

који представља резултат упита типа EXPAND. Класа *Converter* дефинише методе за међусобне конверзије различитих кодних распореда.

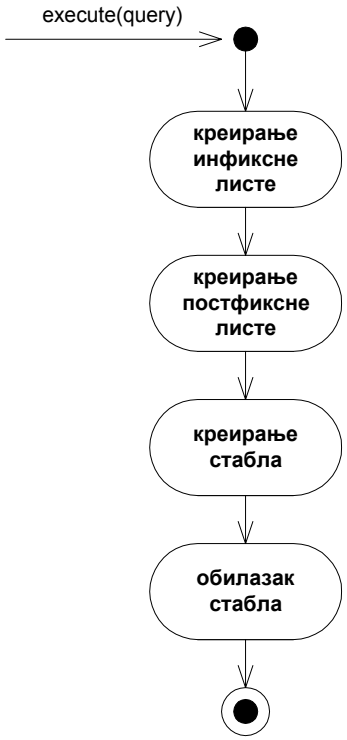


Слика 3.5. Дијаграм класа подсистема за претраживање записа

Процес обраде SELECT упита се иницира (са стране корисника) позивом одговарајуће методе интерфејса *TextServerI*, односно класе *TextServer*. Овај упит се одмах прослеђује класи *ExpressionTree* позивом одговарајуће методе. Класа *ExpressionTree* обраду упита спроводи у четири фазе (метода *execute*):

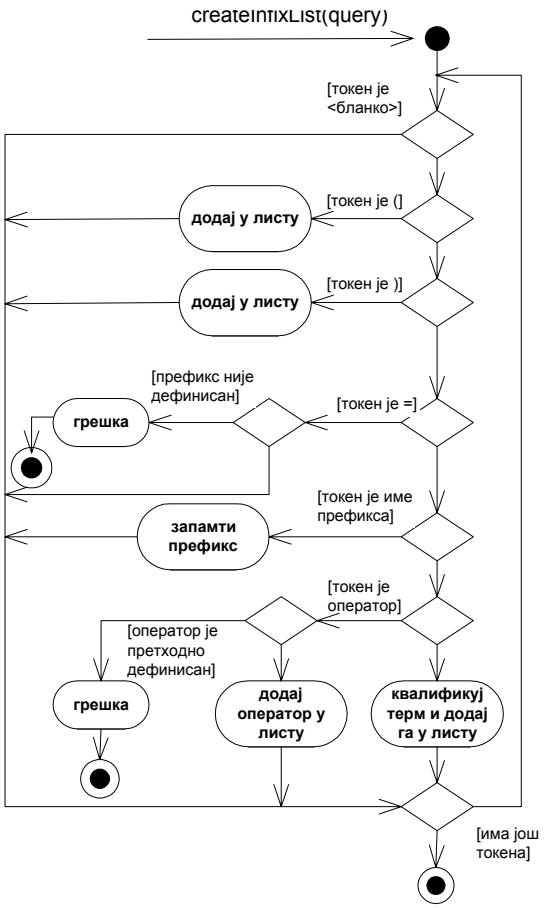
1. креирање инфиксне листе (метода *createInfixList*),
2. креирање постфиксне листе (метода *createPostfixList*),
3. креирање стабла претраге (метода *createTree*),
4. обилазак стабла и рачунање резултата (метода *traverse*).

Дијаграм активности на слици 3.6 приказује редослед операција унутар методе *execute*.



Слика 3.6. Дијаграм активности за методу *ExpressionTree.execute*

Прва фаза дефинисана је алгоритмом који је овде дат својим вербалним описом и дијаграмом активности на слици 3.7.



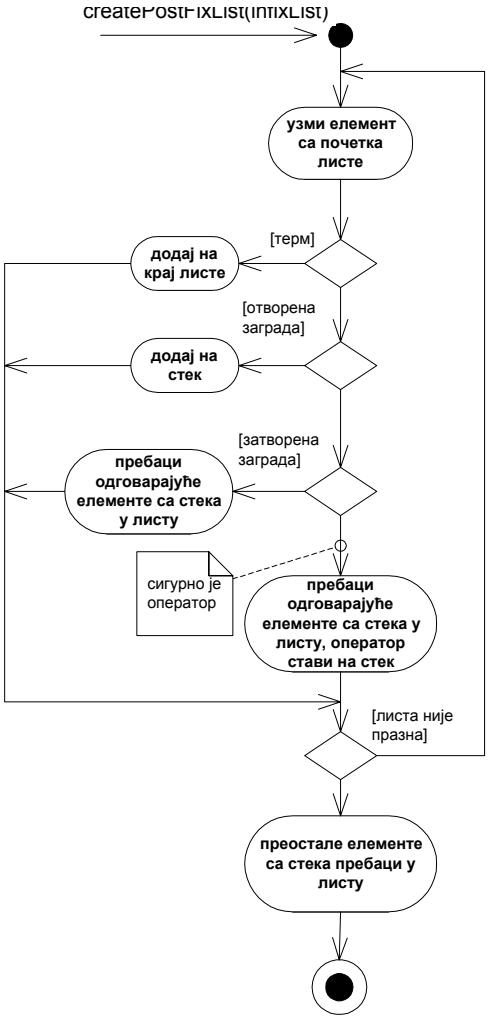
Слика 3.7. Дијаграм активности за методу *ExpressionTree.createInfixList*

Текст упита се подели на токене око делимитера (,), = и <бланко>. За сваки токен се ради следеће:

1. Ако је токен <бланко>, игнорише се.
2. Ако је токен отворена заграда, додаје се у инфиксну листу.
3. Ако је токен затворена заграда, додаје се у инфиксну листу.
4. Ако је токен знак једнакости, проверава се да ли је претходни токен био назив префикса. Ако није, генерише се извештај о грешци. Ако јесте, токен се игнорише.
5. Ако је токен назив префикса, памти се посебно.
6. Ако је токен оператор, додаје се у инфиксну листу.

7. Једина преостала могућност је да је токен терм за претрагу. Ако претходно додати елемент у инфиксној листи постоји и није оператор или отворена заграда, генерише се извештај о грешци. Иначе се терм смешта у инфиксну листу, уз евентуалну квалификацију префиксом ако је он претходно дефинисан.

Друга фаза представља алгоритам за конверзију израза из инфиксне у постфиксну нотацију. Овај алгоритам дат је својим дијаграмом активности на слици 3.8 и следећим вербалним описом.

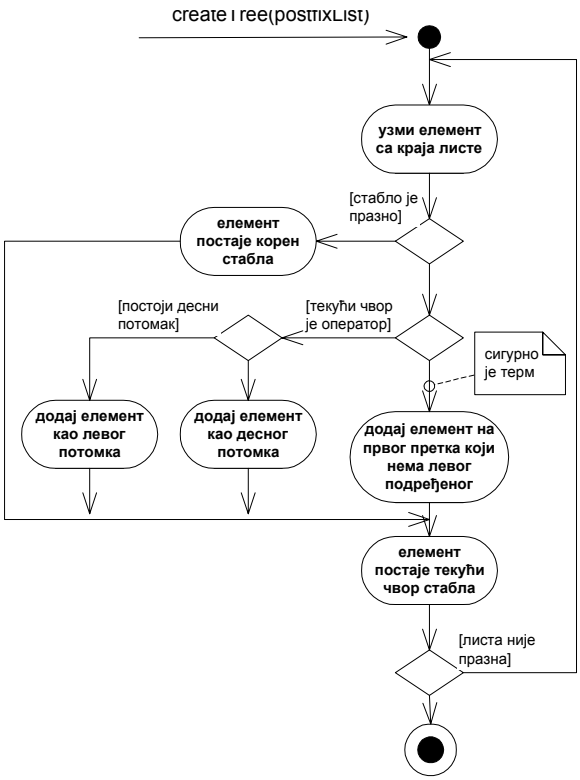


Слика 3.8. Дијаграм активности за методу *ExpressionTree.createPostfixList*

1. Узима се елемент из инфиксне листе, са почетка.
2. Ако је текући елемент терм, додаје се на крај постфиксне листе.
3. Ако је текући елемент отворена заграда, смешта се на стек.
4. Ако је текући елемент затворена заграда, са стека се скидају сви елементи до прве отворене заграде и додају на крај постфиксне листе у редоследу у ком су скидани са стека. Отворена заграда се скида са стека, али се не додаје у постфиксну листу.
5. Ако је текући елемент оператор, са стека се скидају елементи који имају већи приоритет од текућег оператора. Скидање се врши до првог елемента који има мањи приоритет или до прве отворене заграде. Елементи се додају на крај постфиксне листе по редоследу скидања са стека. Текући елемент се, затим, ставља на стек.
6. Кораци 1-5 се понављају док постоје елементи у инфиксној листи.
7. Преостали елементи са стека се додају на крај инфиксне листе у редоследу скидања.

Трећа фаза је алгоритам за креирање синтаксног стабла на основу постфиксне листе. Овај алгоритам приказан је дијаграмом активности на слици 3.9 и својим вербалним описом.

1. Из постфиксне листе се узима елемент са краја.
2. Ако је стабло празно, тај елемент постаје корен стабла, и текући чвор стабла постаје корен. Прелази се на корак 5.
3. Ако је текући чвор стабла оператор, провери се да ли постоји десни подређени чвор текућег чвора. Ако не постоји, елемент из листе постаје десни подређени чвор, а затим постаје и текући чвор стабла. Ако десни подређени чвор постоји, због природе алгоритма леви подређени сигурно не постоји и елемент из листе се смешта у стабло на место левог подређеног чвора, а затим постаје и текући чвор стабла. У овом кораку се врши и провера категорија родитељског и новог чвора. Прелази се на корак 5.
4. Ако је текући чвор стабла терм, тражи се први његов предак који нема левог подређеног а представља оператор. Када се такав елемент нађе, елемент из листе постаје његов леви подређени чвор, а затим постаје и текући чвор стабла.
5. Уколико постфиксна листа није празна, прелази се на корак 1. У другом случају стабло је формирано.



Слика 3.9. Дијаграм активности за методу *ExpressionTree.createTree*

Четврта фаза (обилазак стабла претраге) представља варијанту рекурзивног алгоритма за обилазак бинарног стабла и одређивања скупа погодака за сваки чвор стабла. Одређивање скупа погодака зависи од врсте чвора. Ако је чвор уједно и лист стабла (тј. представља елементарни упит), помоћу одговарајуће методе интерфејса *DocumentManager* се добија одговарајући скуп погодака директно из базе података. Ако чвор представља оператор, скуп погодака се рачуна на начин описан у одељцима 3.4.2 и 3.4.3. У ту сврху се користи класа *HashSet* и одговарајућа класа из групе класа које имплементирају интерфејс *BinaryPredicate*.

Имплементација обраде упита типа EXPAND је далеко једноставнија од обраде SELECT упита. Обрада EXPAND упита се ослања на одговарајућу методу интерфејса *DocumentManager* (и класе која га имплементира) која преузима упит и, на основу генерисаног SQL упита, враћа резултате. Генерисање оваквих SQL упита је приказано у одељку 3.4.3.

Поглавље 4

Имплементација текст сервера UNIMARC записа

Спецификација текст сервера дата у претходном поглављу довољно је флексибилна да омогућава његову имплементацију у различитим системима којима је потребно претраживање библиографске грађе у UNIMARC формату помоћу префикса. Приказани текст сервер користи се као саставни део имплементације библиотечког информационог система БИСИС и Мрежне дигиталне библиотеке докторских, магистарских и дипломских радова на Универзитету у Новом Саду. При томе је за имплементацију коришћен програмски језик Java и скуп сродних технологија за изградњу вишеслојних клијент/сервер система. У овом поглављу дату су детаљи везани за имплементацију релационе базе података који имају велики значај за перформансе система и приказана је имплементација клијената текст сервера у систему БИСИС.

4.1. Имплементација шеме базе података

У поглављу 3 на слици 3.3 приказан је модел података текст сервера у облику дијаграма класа. На слици 3.4 је дат скраћени приказ шема релација добијених уз помоћ одговарајућег CASE алата. CASE алати типично могу да генеришу само спецификације табела, примарних кључева и ограничења. Да би приступ подацима текст сервера био задовољавајуће ефикасан, неопходно је креирати додатне индексе над одговарајућим колонама.

У одељку 3.4.1 је, приликом описивања обраде SELECT упита, дат и пример SQL упита који се шаље ка СУБП:

```
SELECT doc_id, pref_id, word_pos, sent_pos
  FROM pref_AU
  WHERE content = 'PETROVIĆ';
```

Како се приликом обраде SELECT упита генеришу само SQL наредбе таквог облика – над *pref_XX* табелама које имају једнаке шеме релација –

види се да се приступ подацима у колони *content* може убрзати дефинисањем одговарајућег индекса над том колоном. Овакав индекс, код већине данашњих сервера, представља B^+ или B^*-стабло. У том случају, време приступа одређеној n-торци расте логаритамски са порастом броја n-торки у табели [Bender98]. Додавање једног индекса по свакој *pref_XX* табели не успорава у великој мери процес ажурирања табела, па самим тим ни процесе додавања, уклањања или ажурирања UNIMARC записа.

Приликом обраде EXPAND упита, описане у одељку 3.4.3, такође се поставља типски SQL упит:

```
SELECT COUNT(*), a.content
  FROM Prefix_contents a, pref_AU b
  WHERE b.content LIKE 'petr%'
    AND b.word_pos = 1
    AND b.pref_id = a.pref_id
  ORDER BY a.content;
```

На основу поступака за оптимизацију упита које поседују сви савремени сервери за руковање релационим базама података, приказаних у [Bender98], може се рећи да ће креирани индекси и у овом случају бити употребљени на одговарајући начин како би се постигле оптималне перформансе приликом обраде ових упита.

Неки сервери, као што је *Oracle*, омогућавају и анализу поступка обраде сваког упита и, у извесној мери, ручну промену плана обраде упита којег је креирао уграђени оптимизатор. У *Oracle* системима промена плана се врши коришћењем *хинтова* (посебних назнака оптимизатору) који се специфицирају у оквиру коментара који се могу налазити у SQL коду. У следећем примеру је приказан један SQL упит кога генерише сервлет у оквиру веб сајта за корисничко претраживање.

```
SELECT /*+USE_HASH(pref_AU) */ distinct doc_id
  FROM pref_AU
  WHERE content LIKE 'PETAR' AND pref_id IN
  (SELECT pref_id FROM pref_AU
    WHERE content LIKE 'PETROVIĆ%');
```

У датом примеру је коришћен хинт који назначава да се приликом операције споја користи *hash join* алгоритам уместо *nested loops join* алгоритма којег је изабрао оптимизатор упита. У овом случају *hash join* алгоритам даје далеко боље резултате. Различити алгоритми за извршавање операције споја приказани су у [Bender98].

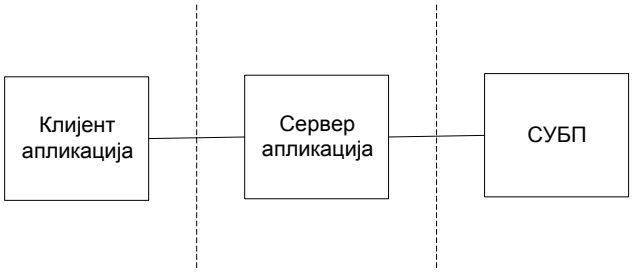
Једна од специфичности сваког од савремених релационих СУБП је начин аутоматског генерисања јединствених нумеричких идентификатора. За ту намену *Oracle* сервери користе тзв. секвенце. Секвенца је објекат базе података намењен генерисању јединствених целих бројева у вишекорисничком окружењу. Секвенце су, у имплементацији шеме базе података за *Oracle* системе, употребљене на одговарајући начин за генерисање јединствених вредности за нпр. *doc_id* (идентификатор записа) и *pref_id* (идентификатор садржаја префикса). Други СУБП нуде другачије технике за генерисање јединствених идентификатора. Коришћење оваквих специфичних техника је задатак класа које наслеђују интерфејс *DocumentManager* (в. одељак 3.3).

4.2. Текст сервер и трослојна архитектура

Основна одредница имплементације текст сервера је трослојна архитектура укупног система (*three-tier architecture*). Трослојна архитектура подразумева поделу система на три, у великој мери независна, подсистема. У питању су следећи подсистеми:

1. подсистем за интеракцију са корисником (имплементира функције корисничког интерфејса);
2. подсистем за имплементацију основних функција система (имплементира тзв. "пословну логику");
3. подсистем за руковање подацима, при чему се пре свега мисли на физички смештај података (ово је заправо систем за управљање базама података).

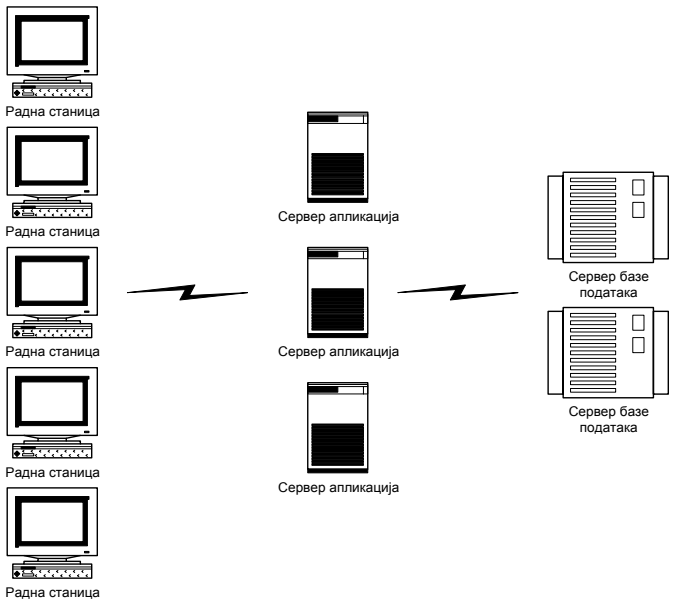
На слици 4.1 је приказан однос ова три подсистема. Са слике се види да не постоји директна веза између подсистема за интеракцију са корисником и подсистема за руковање подацима. Због оваквог међусобног односа, ови подсистеми се често називају *слојеви*.



Слика 4.1. Трослојна архитектура система

Трослојне архитектуре информационих система подразумевају, пре свега, ослањање на стандарде у одговарајућим областима. Најчешће су у питању системи засновани на Интернет технологијама. Ослањање на стандарде омогућава интеграцију информационих система хетерогених у погледу коришћене хардверске и софтверске опреме. На пример, рачунарска мрежа оваквог система може бити заснована на стандардној TCP/IP фамилији протокола. Сервери у мрежи могу бити од различитих произвођача, све док обезбеђују стандардне сервисе предвиђене протоколом.

Друга важна карактеристика трослојних система је скалабилност. Пре свега, повећавање броја клијената је једноставно. Повећавање пропусне моћи сервера средњег слоја је могуће кроз додавање нових серверских машина. Аналогно томе могуће је повећавати и пропусну моћ задњег слоја. Слика 4.2 приказује једну од могућих конфигурација оваквог система.



Слика 4.2. Скица конфигурације система са трослојном архитектуром

Даљим проширивањем концепта трослојних система долази се до појма вишеслојних система (*multitier architecture*), где се врши даља подела на компоненте у оквиру средњег слоја са циљем још већег повећања скалабилности, односно перформанси.

4.3 Имплементација текст сервера

Текст сервер UNIMARC записа имплементиран је у Java окружењу чиме је постигнута независност од коришћене рачунарске платформе. На тај начин се овај сервер може користити на различитим серверским машинама, у складу са потребама и могућностима конкретног информационог система. Текст сервер се, као EJB компонента, уклапа у J2EE архитектуру за изградњу вишеслојних клијент/сервер апликација.

Поред тога, текст сервер је изграђен тако да буде у великој мери независан од коришћеног система за управљање релационим базама података. Додавање подршке за нови СУБП се своди на писање одговарајућег, прецизно дефинисаног модула који се једноставно интегрише у постојећи систем.

Текст сервер се користи се у следећим модулима БИСИС-а:

- корисничко претраживање и
- обрада библиографске грађе.

4.3.1. Корисничко претраживање

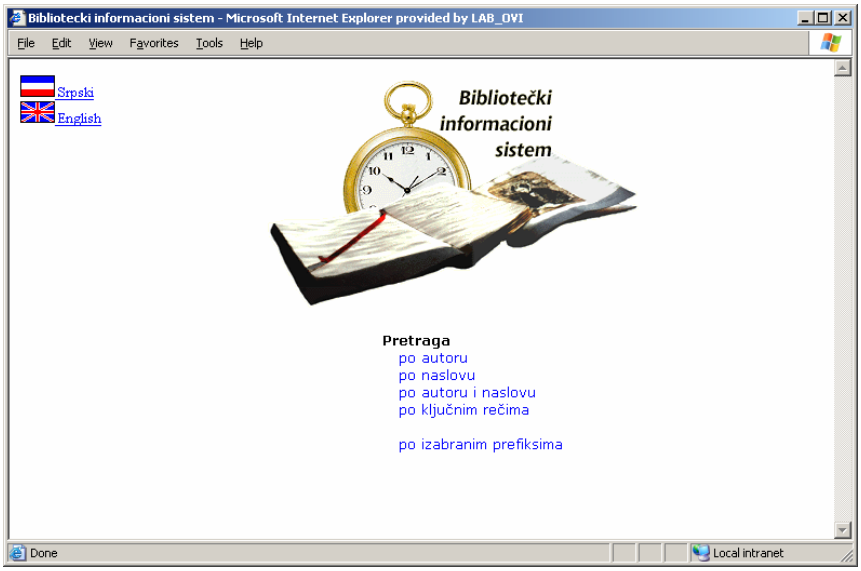
Корисничко претраживање је модул софтверског система БИСИС који омогућава претраживање библиотечког фонда путем веб сајта. Намењен је за најшири круг корисника, тј. за све кориснике библиотеке. Основна страница сајта приказана је на слици 4.3. На овој страници сајта корисник може да бира језик којим се исписују текстуалне поруке. Основна страница нуди неколико начина претраживања. Могуће је постављати једноставне упите (по једном критеријуму, као што је аутор или наслов дела), или сложеније упите који могу да садрже више критеријума по којима се врши претрага.

Избором линка **po izabranim prefiksima** на основној страници сајта приказује се страница за формирање упита који омогућава комбиновање различитих префикса у претрази, као на слици 4.4.

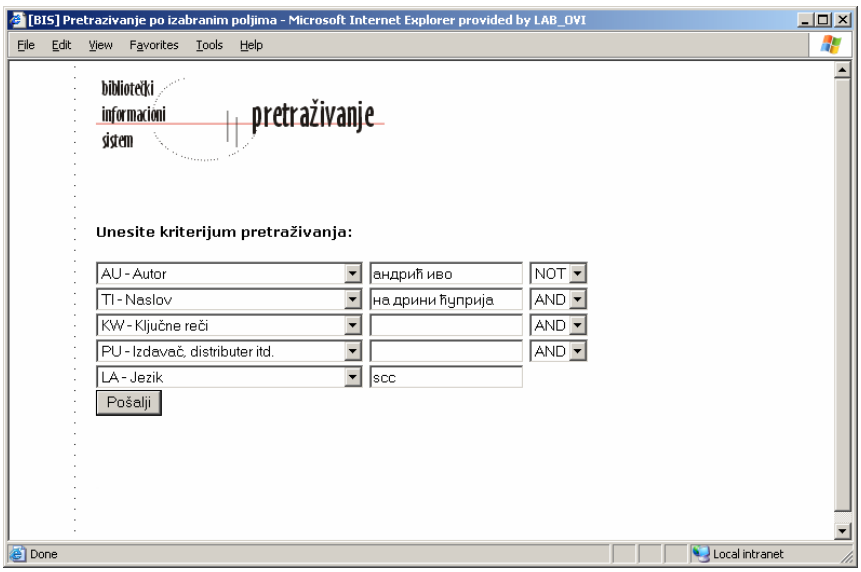
Страница за формирање упита по изабраним префиксима омогућава комбиновање претраге по највише пет различитих префикса. Избор префикса по коме се врши претрага за дати садржај обавља се помоћу падајућих листа на левој страни. Тражени садржај наводи се у оквиру едитора који се налазе у истом реду са падајућом листом. Повезивање са осталим елементима упита помоћу логичких оператора (*and*, *or*, *not*, односно *и*, *или*, *не*) обавља се помоћу падајуће листе на десној страни која се налази у истом реду са одговарајућим едитором. Уколико је неки од едитора празан, одговарајући префикс се

игнорише током претраге. Слика 4.4 садржи пример упита у коме се траже све библиографске јединице за које важи следеће:

– аутор библиографске јединице је Иво Андрић (у падајућој листи у првом реду је изабран префикс *AU - Autor*, а у одговарајућем едитору је унет текст *андрић иво*),
– наслов библиографске јединице **није** „На Дрини ћуприја“ (у претходној падајућој листи са логичким операторима изабрана је опција *NOT*, изабран је префикс *TI – Naslov*, а у одговарајућем едитору је унет текст *на дрини ћуприја*)
– језик библиографске јединице је српски, писан ћирилицом (у петој падајућој листи за избор префикса изабран је префикс *LA – jezik*, у одговарајућем едитору је унет текст *scc*, што је одговарајућа шифра језика, а у претходној падајућој листи са логичким операторима, која је у другом реду, изабран је оператор *AND*).



Слика 4.3. Основна страница сајта за корисничко претраживање



Слика 4.4. Формирање упита по изабраним префиксима

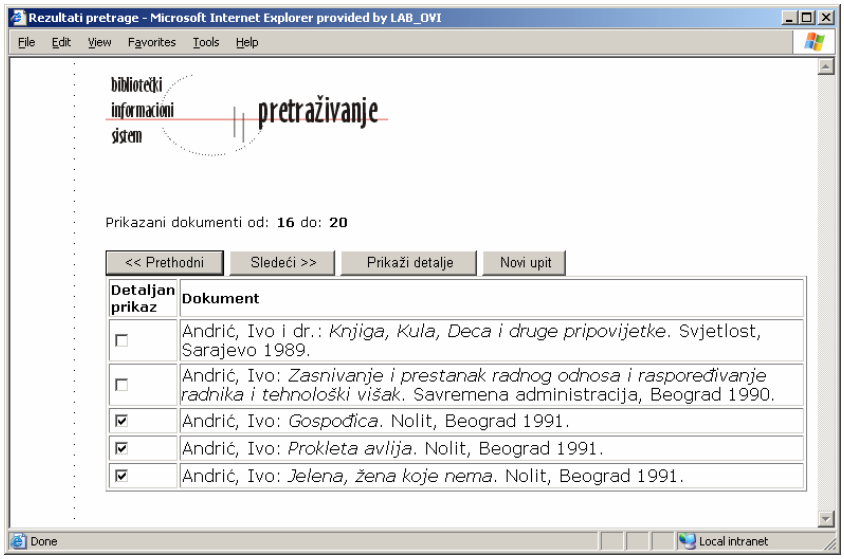
Упит формиран путем екранске форме приказане на слици 4.4 може се изразити еквивалентним SELECT упитом текст серверу који гласи:

```
SELECT (AU=andrić [W1] AU=ivo) AND
       (TI=na [W1] TI=drini [W1] TI=ćuprija) AND LA=scc
```

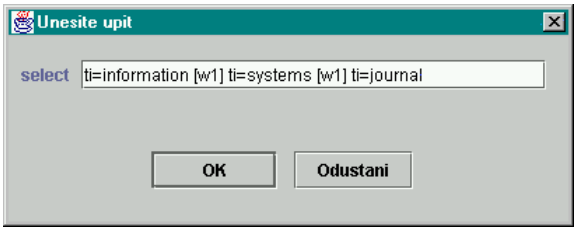
Овако формиран упит упућује се текст серверу који врши његово интерпретирање и израчунавање резултата претраге. Кориснику се, као резултат претраге, приказује извештај о броју пронађених записа са могућношћу њиховог прегледања као на слици 4.5.

4.3.2. Обрада библиографске грађе

Библиотечки информациони систем БИСИС поседује наменску апликацију за обраду библиографске грађе коју користе библиотекари. Ова апликација омогућава и претраживање фонда библиографске грађе у датој бази података. Претраживање се може вршити директно помоћу SELECT упита *Dialog* језика. Пример са слике 4.6 приказује претраживање у коме се траже записи у чијем се наслову (префикс TI) налази фраза *information systems journal* што се у упитном језику изражава као дати низ речи на међусобном растојању од 1, тј. повезаних оператором [w1].



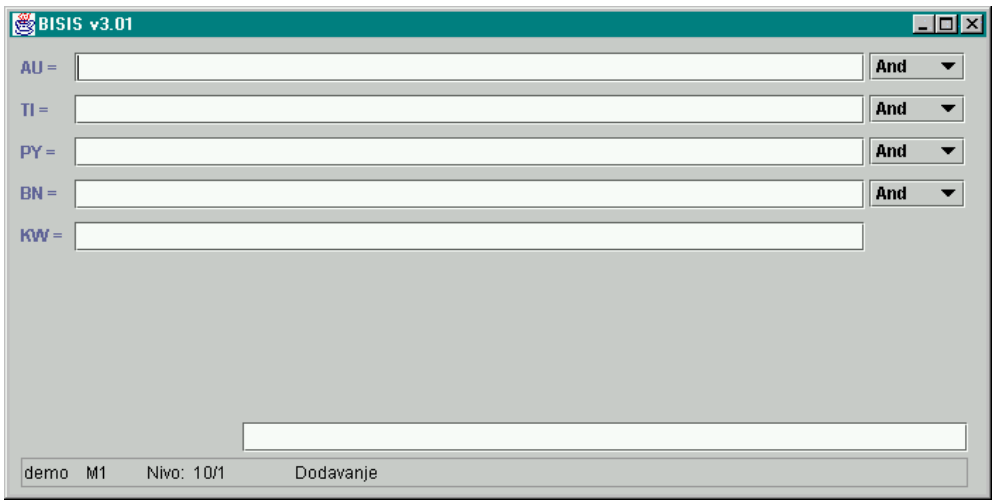
Слика 4.5. Преглед пронађених библиографских записа



Слика 4.6. Унос SELECT упита за претраживање грађе

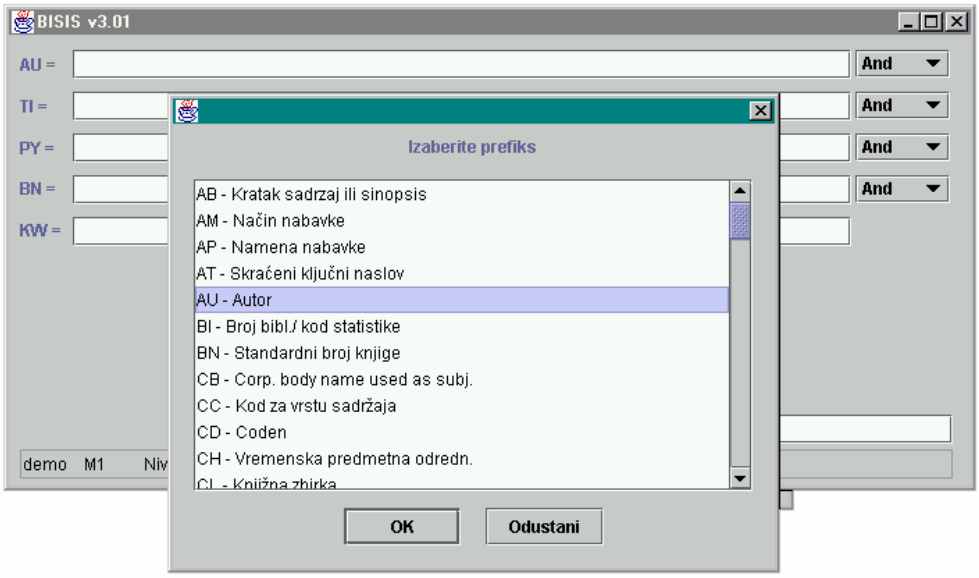
Унос упита директним навођењем SELECT команде може бити сувише сложен за крајње кориснике система. Стога је као основни начин претраживања, и у апликацији за обраду библиографске грађе, усвојена екранска форма која је функционално еквивалентна форми из модула за корисничко претраживање (слика 4.4). Ова форма је приказана на слици 4.7.

Префикси по коме ће се вршити претраживање могу бити из скупа свих дефинисаних префикса. Избор префикса за претраживање обавља се у посебном прозору који се отвара притиском на тастер **<F9>**. Прозор за избор префикса приказан је на слици 4.8.

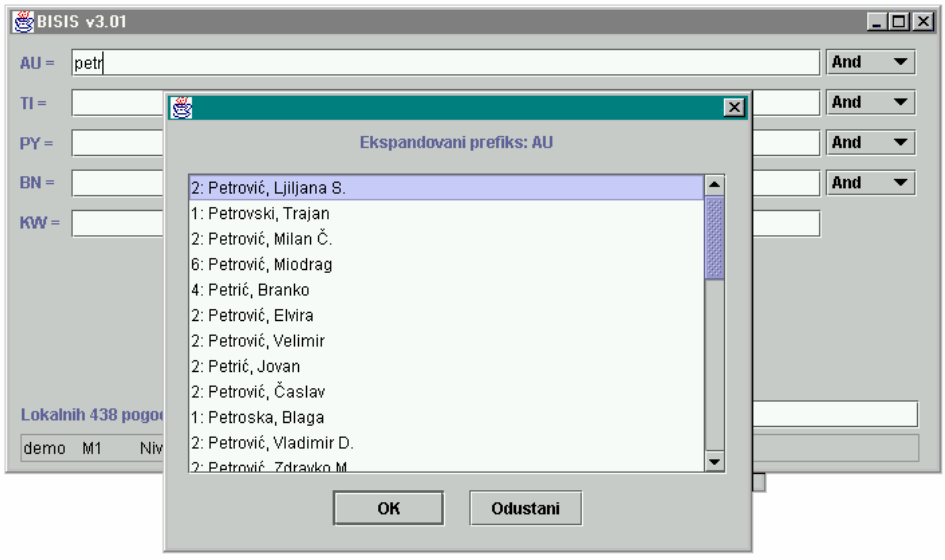


Слика 4.7. Основна екранска форма за претраживање

Упити типа EXPAND за изабрани префикс се могу поставити у оквиру исте форме. Навођењем садржаја упита у поље за унос префикса и притиском на тастер **<F8>** формира се EXPAND упит и прослеђује текст серверу. Резултат упита приказан је у посебном прозору. У примеру са слике 4.9 извршен је EXPAND упит за префикс AU и садржај *petr*.Жељени садржај префикса, из скупа садржаја префикса који чине резултат, може се преузети у основну форму ради даљег претраживања помоћу SELECT упита.



Слика 4.8. Избор префикса за претраживање



Слика 4.9. Резултат EXPAND упита

Литература

[Abiteboul01] S. Abiteboul, L. Segoufin, V. Vianu. *Representing and Querying XML with Incomplete Information*. Proceedings of 20th Symposium on Principles of Database Systems, 2001, pp. 150-161.

[Abiteboul97] S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte, J. Siméon. *Querying Documents in Object Databases*. International Journal on Digital Libraries, Vol. 1, No. 1, 1997. pp. 5-19.

[ArnoldMoore99] T. Arnold-Moore, M. Fuller, R. Sacks-Davis. *Approaches for Structured Document Management*. Proceedings of Markup Technologies, 1999.

[BaezaYates00] R. Baeza-Yates, G. Navarro. *XQL and Proximal Nodes*. Proceedings of the XML Workshop of 23rd ACM SIGIR Conference on Research and Development in Information Retrieval, 2000.

[BaezaYates02] R. Baeza-Yates, B. Bustos, E. Chávez, N. Herrera, G.Navarro. *Clustering in Metric Spaces with Applications to Information Retrieval*. In W. Wu, H. Xiong, S. Shekhar, editors, *Information Retrieval and Clustering*. Kluwer Academic Press, 2002.

[BaezaYates94] R. Baeza-Yates. *A Hybrid Query Model for Full Text Retrieval Systems*. Tech Report No. DCC-1994-2, Dept. of Computer Science, Univ. of Chile, 1994.

[BaezaYates96] R. Baeza-Yates and G. Navarro. *Integrating Contents and Structure in Text Retrieval*. ACM SIGMOD Record, Vol. 25, No. 1, 1996. pp. 67-79.

[BaezaYates99] R. Baeza-Yates, B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, New Providence, 1999.

[Belkin92] N.J. Belkin, W.B. Croft. *Information Filtering and Information Retrieval: Two Sides of the Same Coin?*. Communications of the ACM, Vol. 35, No. 12, 1992. pp. 29-38.

[Bender98] M. Bender. *Fizičko projektovanje relacionih baza podataka*. Magistarska teza, Fakultet tehničkih nauka, Novi Sad, 1998.

[Bonifati00] A. Bonifati, S. Ceri. A Comparative Study of Five XML Query Languages. ACM SIGMOD Record, Vol. 29, No. 1, 2000. pp. 68-79.

[Bourret01] R. Bourret. *Mapping DTDs to Databases*. 2001. http://www.xml.com/pub/a/2001/05/09/dtdtodbs.html

[Budimir04] Г. Будимир. *Контрола квалитета XML библиографских записа*. Магистарска теза, Природно-математички факултет, Департман за математику и информатику, Нови Сад, 2004. http://diglib.ns.ac.yu/ndltd/docs/set3/ndltd409/kontrola_kvaliteta.pdf

[CanonicalXML] J. Boyer. *Canonical XML Version 1.0*. W3C Recommendation, 2001. http://www.w3.org/TR/2001/REC-xml-c14n-20010315/

[Chang87] S.-K. Chang, Q.-Y. Shi, C.-W. Yan. *Iconic Indexing by 2-D Strings*. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 9, No. 3, 1987. pp. 413-428

[Chinenyanga01] T.T. Chinenyanga and N. Kushmerick. *Expressive and Efficient Ranked Queries for XML Data*. Proceedings of 4th International Workshop on the Web and Databases (WebDB'01), 2001.

[Christophides94] V. Christophides, S. Abiteboul, S. Cluet, M. Scholl. *From Structured Documents to Novel Query Facilities*. ACM SIGMOD Record, Vol. 23, No. 2, 1994. pp. 313-324.

[Clarke95a] C. Clarke, G. Cormack, F. Burkowski. *An Algebra for Structured Text Search and a Framework for its Implementation*. The Computer Journal, 1995.

[Clarke95b] C. Clarke, G. Cormack, F. Burkowski. *Schema-Independent Retrieval from Heterogenous Structured Text*. Proceedings of 4th Annual Symposium on Document Analysis and Information Retrieval, 1995.

[COMARC] *COMARC/R Format*, COBISS, Univerzitetni institut informacijskih znanosti, IZUM, Maribor 1992.

[Croft79] W.B. Croft, D.J. Harper. *Using Probabilistic Models of Retrieval Without Relevance Information*. Journal of Documentation, Vol. 35, No. 4, 1979. pp. 285-295.

[Croft83] W.B. Croft. *Experiments with Representation in a Document Retrieval System*. Information Technology: Research and Development, Vol. 2, No. 1, 1983. pp. 1-21.

[Croft91] W.B. Croft, H.R. Turtle, D.D. Lewis. *The Use of Phrases and Structured Queries in Information Retrieval*. Proceedings of SIGIR-91, 14th ACM Conference on Research and Development in Information Retrieval, 1991. pp. 32-45.

[Desai86] B. Desai, P. Goyal, S. Sadri. *A Data Model for Use with Formatted and Textual Data*. Journal of the American Society for Information Science, Vol. 37, No. 3, 1986. pp. 158-165.

[Dialog] *Successful Searching on Dialog*. Dialog Corporation, 1998. http://support.dialog.com/searchaids/success/

[Diglib] *Mrežna digitalna biblioteka doktorskih, magistarskih i diplomskih radova*. Univerzitet u Novom Sadu. http://diglib.ns.ac.yu

[DOM1] A. Le Hors, P. Le Hégaret, L. Wood, G. Nicol, J. Robie, M. Champion, S. Byrne. *Document Object Model (DOM) Level 2 Core Specification Version 1.0*. W3C Recommendation, 2000. http://www.w3.org/TR/DOM-Level-2-Core

[Everitt80] B. Everitt. *Cluster Analysis*. Halsted Press, 1980.

[Fan01] W. Fan, L. Libkin. *On XML Integrity Constraints in the Presence of DTDs*. Proceedings of 20th Symposium on Principles of Database Systems, 2001. pp. 114-125.

[Fasulo99] D. Fasulo. *An Analysis of Recent Work on Clustering Algorithms*. Tech Report No. 01-03-02, Dept. of Computer Science and Engineering, University of Washington, 1999.

[Fawcett89] H. Fawcett. *PAT 3.3 User's Guide*. UW Centre for New OED and Text Research, Univ. of Waterloo, 1989.

[Florescu99b] D. Florescu, D. Kossmann. *Storing and Querying XML Data Using an RDBMS*. IEEE Data Engineering Bulletin, Vol. 22, No. 3, 1999. pp. 27-34.

[Fox83b] E.A. Fox. *Characterization of Two New Experimental Collections in Computer and Information Science Containing Textual and Bibliographical Concepts*. Tech Report, NCSTRL, 1983.

[Frakes92] W.B. Frakes, R. Baeza-Yates, editors. *Information Retrieval: Data Structures & Algorithms*. Prentice Hall, 1992.

[Furnas88] G.W. Furnas, S. Deerwester, S.T. Dumais, T.K. Landauer, R.A. Harshman, L.A. Streeter, K.E. Lochbaum. *Information Retrieval Using a Singular Value Decomposition Model of Latent Semantic Structure*. Proceedings of the 11th ACM SIGIR Conference on Research and Development in Information Retrieval, 1988. pp. 465-480.

[Garey79] M. Garey, D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[Gonnet87] G. Gonnet. *Examples of PAT Applied to the Oxford English Dictionary*. Tech Report No. OED-87-02, UW Centre for New OED and Text Research, Univ. of Waterloo, 1987.

[Gredley90] E. Gredley, A. Hopkinson. *Bibliographic Data Exchange: MARC and Other International Formats*. The Library Association, London, 1990.

[Gudivada95] V.N. Gudivada, V.V. Raghavan. *Design and Evaluation of Algorithms for Image Retrieval by Spatial Similarity*. ACM Transactions on Information Systems, Vol. 13, No. 2, 1995. pp. 115-144.

[Guglielmo96] E.J. Guglielmo, N.C. Rowe. *Natural-Language Retrieval od Images Based on Descriptive Captions*. ACM Transactions on Information Systems, Vol. 14, No. 3, 1996. pp. 237-267.

[Guttman84] A. Guttman. *R-Trees: A Dynamic Index Structure for Spatial Searching*. Proceedings of ACM SIGMOD Conference on Management of Data, 1984. pp. 47-57.

[Haines93] D. Haines, W.B. Croft. *Relevance Feedback and Inference Networks*. Proceedings of the SIGIR-93, 16th ACM Conference on Research and Development in Information Retrieval, 1993. pp. 2-11.

[Harman92] D. Harman. *Relevance Feedback Revisited*. Proceedings of the SIGIR-92, 15th ACM Conference on Research and Development in Information Retrieval, 1992. pp. 1-10.

[Harman95] D.K. Harman. *Overview of the Third Text Retrieval Conference*. Proceedings of the 3rd Text REtrieval Conference (TREC-3), 1995. pp. 1-19.

[Harper78] D.J. Harper and C.J. van Rijsbergen. *An Evaluation of Feedback in Document Retrieval Using Co-occurence Data*. Journal of Documentation, Vol. 34, No. 3, 1978. pp. 189-216.

[He02] J. He, A.-H. Tan, C.-L. Tan, S.-Y. Sung. *On Quantitative Evaluation of Clustering Systems*. In W. Wu, H. Xiong, S. Shekhar, editors, *Information Retrieval and Clustering*. Kluwer Academic Press, 2002.

[He99] Q. He. *A Review of Clustering Algorithms as Applied in IR*. Tech Report No. UIUCLIS-1999/6+IRG, Univ. of Illinois at Urbana-Champaign, 1999.

[Hearst96] M.A. Hearst, J.O. Pedersen. *Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results*. Proceedings of SIGIR-96, 19th ACM Conference on Research and Development in Information Retrieval, 1996. pp. 76-84.

[Ide71] E. Ide. *New Experiments in Relevance Feedback*. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, 1971. pp. 337-354.

[ISO10646] *Information technology – Universal Multiple-Octet Coded Character Set (UCS)*. ISO/IEC 10646:2003. http://www.iso.org

[Kaszkiel99] M. Kaszkiel, J. Zobel, R. Sacks-Davis. *Efficient Passage Ranking for Document Databases*. ACM Transactions on Information Systems, Vol. 17, No. 4, 1999. pp. 406-439.

[Kilpelainen93] P. Kilpeläinen, H. Manilla. *Retrieval from Hierarchical Texts by Partial Patterns*. Proceedings of SIGIR-93, 16th ACM Conference on Research and Development in Information Retrieval, 1993. pp. 214-222.

[Kilpelainen95] P. Kilpeläinen, H. Manilla. *Ordered and Unordered Tree Inclusion*. SIAM Journal on Computing, Vol. 24, No. 2, 1995. pp. 340-356.

[Knuth98] D.E. Knuth. *Sorting and Searching.* Volume 3 of *The Art of Computer Programming*, Second edition, Addison Wesley, 1998.

[Kovčić00] A. Kovčić, B. Milosavljević. *Data Model for Indexing and Searching XML Documents*. Novi Sad Journal of Mathematics, Vol. 31, No. 1, 2000. pp. 151-158.

[Leavitt00] N. Leavitt. *Whatever Happened to Object-Oriented Databases?*. IEEE Computer, Vol. 33, No. 8, 2000. pp. 16-19.

[Lazarević96] B. Lazarević, redaktor. *Formiranje i pretraživanje baza podataka u Sistemu naučnih i tehnoloških informacija Srbije*. Ministarstvo za nauku i tehnologiju Republike Srbije, Beograd, 1996.

[Leuski01] A. Leuski. *Evaluating Document Clustering for Interactive Information Retrieval*. Proceedings of CIKM'01, 10th ACM Conference on Information and Knowledge Management, 2001. pp. 33-40.

[Losee88] R.M. Losee, A. Bookstein. *Integrating Boolean Queries in Conjunctive Normal Form with Probabilistic Retrieval Models*. Information Processing & Management, Vol. 24, No. 3, 1988. pp. 315-321.

[Lu96] A. Lu, M. Ayoub, J. Dong. *Ad Hoc Experiments Using EUREKA*. TREC-96, 1996.

[Lundquist97] C. Lundquist, D.A. Grossman, O. Frieder. *Improving Relevance Feedback in the Vector Space Model*. Proceedings of CIKM-97, 6th ACM Conference on Information and Knowledge Management, 1997. pp. 16-23.

[MacLeod90] I. MacLeod. *Storage and Retrieval of Structured Documents*. Information Processing & Management, Vol. 26, No. 2, 1990. pp. 197-208.

[MacLeod91] I. MacLeod. *A Query Language for Retrieving Information from Hierarchic Text Structures*. The Computer Journal, Vol. 34, No. 3, 1991. pp. 254-264.

[Meghini01] C. Meghini, F. Sebastiani, U. Straccia. *A Model of Multimedia Information Retrieval*. Journal of the ACM, Vol. 48, No. 5, 2001. pp. 909-970.

[Milosavljević00a] B. Milosavljević, Z. Konjović. *Arhitektura WWW prezentacije bibliotečkog informacionog sistema*. Zbornik radova YU Info 2000, Kopaonik, 2000.

[Milosavljević00b] B. Milosavljević, Z. Konjović. *Arhitektura WWW prezentacije za pretraživanje u bibliotečkom informacionom sistemu*. Info Science, Vol. 8, No. 4, 2000. pp. 18-23.

[Milosavljević00c] B. Milosavljević and Z. Konjović. *Modeling and Implementation of a System for Bibliographic Records Interchange*. Novi Sad Journal of Mathematics, Vol. 31, No. 1, 2000. pp. 159-166.

[Milosavljević00d] Б. Милосављевић. *Узајамна каталогизација: преузимање библиографских записа*. Инфотека, Vol. 1, No. 1, 2000. pp. 19-23.

[Milosavljević00e] B. Milosavljević, M. Zeremski. *Definicija tipa dokumenta UNIMARC formata*. Zbornik radova YU INFO 2000, Kopaonik, 2000.

[Milosavljević02b] B. Milosavljević, Z. Konjović. *Design of an XML-Based Extensible Multimedia Information Retrieval System*. IEEE Multimedia Software Engineering (MSE2002), Newport Beach, CA, 2002. pp. 114-121.

[Milosavljević02c] B. Milosavljević, Z. Konjović. *Design of an Extensible Multimedia Information Retrieval System*. 6th Balkan Conf. on Operational Research, Thessaloniki, 2002. (in print).

[Milosavljević02a] B. Milosavljević, M. Vidaković, D. Surla. *Rukovanje višejezičnim tekstom u Bibliotečkom informacionom sistemu BISIS, ver. 3*. Zbornik radova Internet i ćirilica: srpski jezik, pismo i kultura u savremenim informacionim tehnologijama, Narodna biblioteka Srbije, Beograd, 2002. http://www.rastko.org.yu/projekti/cirilica2/net-cirilica2002/bmilosavljevic.html.

[Milosavljević03a] B. Milosavljević. *Proširivi sistem za pronalaženje multimedijalnih dokumenata*. Doktorska disertacija, Fakultet tehničkih nauka, Univerzitet u Novom Sadu, 2003.

[Milosavljević03b] B. Milosavljević, Z. Konjović. *Multimedia Document Retrieval in the Networked Digital Library of Theses and Dissertations*. 4th Intl. Conf. on Informatics and Information Technology, Bitola, 2003. (in print)

[Milosavljević04a] Б. Милосављевић. *Текст сервер UNIMARC записа*. У Д. Сурла, З. Коњовић, уредници, *Дистрибуирани библиотечки информациони систем БИСИС*, Група за информационе технологије, Нови Сад, 2004.

[Milosavljević04b] Б. Милосављевић. *Корисничко претраживање*. У Д. Сурла, З. Коњовић, уредници, *Дистрибуирани библиотечки информациони систем БИСИС*, Група за информационе технологије, Нови Сад, 2004.

[Milosavljević04c] B. Milosavljević. *User Search in the Library Information System BISIS*. Intl. Conf. on Distributed Library Information Systems, Ohrid, June 1-6, 2004. (in print).

[Milosavljević04d] Б. Милосављевић, З. Коњовић. *Претраживање мултимедијалних информација у систему дигиталних библиотека магистарских и докторских радова*. ИНФОТЕКА, Београд, бр. 1-2/2004. стр. 87-98.

[Milosavljević97a] B. Milosavljević, D. Surla. *Implementacija tekst servera za indeksiranje i pretraživanje bibliotečke građe u Oracle okruženju*. Zbornik radova Info-Teh '97, Vrnjačka Banja 1997. str. 452-456.

[Milosavljević97b] B. Milosavljević, Z. Konjović. *Tekst server bibliotečkog informacionog sistema zasnovan na Oracle ConText Option-u*. Zbornik apstrakata Sinfon '97, Zlatibor 1997.

[Milosavljević98] B. Milosavljević. *Tekst server bibliotečkog informacionog sistema zasnovan na Oracle ConText Option-u*. Zbornik radova YU Info 98, Kopaonik 1998.

[Milosavljevic99] Б. Милосављевић. *Текст сервер UNIMARC записа*. Магистарска теза, Факултет техничких наука, Нови Сад, 1999.

[Milosavljević99a] B. Milosavljević, Z. Konjović. *Specifikacija sistema za indeksiranje UNIMARC zapisa*. Zbornik radova YU Info 99, Kopaonik 1999.

[Milosavljević99b] B. Milosavljević, Z. Konjović. *Specifikacija pretraživanja u tekst serveru za UNIMARC zapise*. Zbornik radova Sym-op-is 99, Beograd, 1999. str. 145-148.

[Milosavljević99c] B. Milosavljević, D. Surla. *Arhitektura tekst servera za XML dokumente*. Zbornik radova Sym-op-is 99, Beograd, 1999. str. 149-152.

[Milosavljević99d] Б. Милосављевић. *Текст сервер UNIMARC записа*. Магистарска теза, Факултет техничких наука, Нови Сад, 1999.

[Navarro95] G. Navarro, R.Baeza-Yates. *A Language for Queries on Structure and Contents of Textual Databases*. Proceedings of SIGIR-95, 18th ACM

Conference on Research and Development in Information Retrieval, 1995. pp. 93-101.

[Navarro97] G. Navarro, R. Baeza-Yates. *Proximal Nodes: A Model to Query Document Databases by Content and Structure*. ACM Transactions on Information Systems, Vol. 15, No. 4, 1997. pp. 400-435.

[Ogawa91] Y. Ogawa, T. Morita, K. Kobayashi. *A Fuzzy Document Retrieval System Using the Keyword Connection Matrix and a Learning Method*. Fuzzy Sets and Systems, Vol. 39, 1991. pp. 163-179.

[Pearl88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Pupovac04] Б. Пуповац. *YUMARC формат.* У Д. Сурла, З. Коњовић, уредници, *Дистрибуирани библиотечки информациони систем БИСИС*. Група за информационе технологије, Нови Сад, 2004.

[Radecki76] T. Radecki. *Mathematical Model of Information Retrieval System Based on the Concept of Fuzzy Thesaurus*. Information Processing & Management, Vol. 12, 1976. 313-318.

[Radecki77] T. Radecki. *Mathematical Model of Time-Effective Information Retrieval System Based on the Theory of Fuzzy Sets*. Information Processing & Management, Vol. 13, 1977. 109-116.

[Radecki83] T. Radecki. *Incorporation of Relevance Feedback into Boolean Retrieval Systems*. Proceedings of SIGIR-83, 6th ACM Conference on Research and Development in Information Retrieval, 1983. pp. 133-150.

[RibeiroNeto96] B. Ribeiro-Neto, R. Muntz. *A Belief Network Model for IR*, Proceedings of the 19th ACM SIGIR Conference on Research and Development in Information Retrieval, 1996. pp. 253-260.

[Robertson76] S.E. Robertson, K. Sparck Jones. *Relevance Weighting of Search Terms*. Journal of the American Society for Information Science, Vol. 27, No. 3, 1976. pp. 129-146.

[Rocchio71] J.J. Rocchio. *Relevance Feedback in Information Retrieval*. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, 1971. pp. 313-323.

[Salminen01] A. Salminen, F.Wm. Tompa. *Requirements for XML Document Database Systems*. Proceedings of the ACM Symposium on Document Engineering, 2001, pp. 85-94.

[Salminen92] A. Salminen, F.Wm. Tompa. *PAT Expressions: An Algebra for Text Search*. Proceedings of COMPLEX-92, 1992. pp. 309-332.

[Salton68] G. Salton, M.E. Lesk. *Computer Evaluation of Indexing and Text Processing*. Journal of the ACM, Vol. 15, No. 1, 1968. pp. 8-36.

[Salton71] G. Salton. *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice-Hall, 1971.

[Salton83] G. Salton, M.J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[Salton83b] G. Salton, E.A. Fox, H. Wu. *Extended Boolean Information Retrieval*. Communications of the ACM, Vol. 26, No. 11, 1983. pp. 1022-1036.

[Salton84] G. Salton, E. Voorhees, E. Fox. *A Comparison of Two Methods for Boolean Query Relevance Feedback*. Information Processing & Management, Vol. 20, No. 5/6, 1984. pp. 637-651.

[Salton88] G. Salton, C. Buckley. *Term-Weighting Approaches in Automatic Retrieval*. Information Processing & Management, Vol. 24, No. 5, 1988. pp. 513-523.

[Salton90] G. Salton, C. Buckley. *Improving Retrieval Performance by Relevance Feedback*. Journal of the American Society for Information Science, Vol. 41, No. 4, 1990. pp. 288-297.

[Schlieder00] T. Schlieder, F. Naumann. *Approximate Tree Embedding for Querying XML Data*. ACM SIGIR Workshop On XML and Information Retrieval, 2000.

[Schlieder01] T. Schlieder. *ApproXQL: Design and Implementation of an Approximate Pattern Matching Language for XML*. Tech Report No. B 01-02, Freie Universität Berlin, 2001. http://www.inf.fu-berlin.de/inst/ag-db/publications/2001/report-B-01-02.ps

[SGML86] International Standards Organization, *Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML)*. ISO 8879-1986, 1986.

[Shanmugasundaram99] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D.J. DeWitt, J.F. Naughton. *Relational Databases for Querying XML Documents: Limitations and Opportunities*. Proceedings of the 25th International Conference on Very Large Databases, 1999. pp. 302-314.

[SparckJones79] K. Sparck Jones. *Experiments in Relevance Weighting of Search Terms*. Information Processing & Management. Vol. 15, No. 13, 1979, pp. 133-144.

[Spink98] A. Spink, T. Saračević. *Human-Computer Interaction in Information Retrieval: Nature and Manifestations of Feedback*. Interacting with Computers, Vol. 10, 1998. pp. 249-367.

[Stonebraker83] M. Stonebraker, H. Stettner, N. Lynn, J. Kalash, A. Guttman. *Document Processing in a Relational Database System*. ACM Transactions on Office Information Systems, Vol. 1, No. 2, 1983. pp. 143-158.

[Surla03] Д. Сурла, З. Коњовић, Б. Пуповац, Б. Милосављевић, М. Видаковић, Т. Зубић, Т. Тошић. *Упутство за коришћење библиотечког софтверског система БИСИС вер. 3*. Група за информационе технологије, Нови Сад, 2003.

[Surla04a] Д. Сурла, З. Коњовић, Б. Милосављевић, М. Зарић, Г. Сладић, З. Протић, С. Комазец, Д. Окановић. *Приказ реализације мрежне дигиталне библиотеке докторских, магистарских и дипломских радова.* Инфотека, бр. 1-2, Београд, 2004. стр. 75-86.

[Surla04b] Д. Сурла, З. Коњовић, уредници. *Дистрибуирани библиотечки информациони систем БИСИС*. Група за информационе технологије, Нови Сад, 2004.

[Škrbić04a] S. Škrbić, D. Surla. *Upravljanje XML bibliografskim zapisima u XML native tehnologiji*. Zbornik radova Sym-op-is '04, 2004. (prihvaćeno za štampu).

[Škrbić04b] S. Škrbić. *Storing and Searching XML Documents*. XVI Conference on Applied Mathematics, Budva, 2004.

[Škrbić04c] S. Škrbić, D. Surla. *Storing XML Bibliographic Records*. XVI Conference on Applied Mathematics, Budva, 2004.

[Theobald00] A. Theobald, G. Weikum. *Adding Relevance to XML*. Proceedings of 3rd International Workshop on the Web and Databases (WebDB'00), 2000.

[Tudhope99] D. Tudhope, D. Cunliffe. *Semantically Indexed Hypermedia: Linking Information Disciplines*. ACM Computing Surveys, Vol. 31, No. 4es, 1999. pp. 1-6.

[Turtle90] H. Turtle, W.B. Croft. *Inference Networks for Document Retrieval*. Proceedings of the 13th ACM SIGIR Conference on Research and Development in Information Retrieval, 1990. pp. 1-24.

[Turtle91] H. Turtle, W.B. Croft. *Evaluation of an Inference Network-Based Retrieval Model*. ACM Transactions on Information Systems, Vol. 9, No. 3, 1991. pp. 187-222.

[Unicode] The Unicode Consortium. *The Unicode Standard, Version 4.0*. Addison Wesley, Reading, 2003. http://www.unicode.org/versions/Unicode4.0/

[UNIMARC] *UNIMARC Manual: Bibliographic format*. International Federation of Library Association and Institutions, IFLA Universal Bibliographic

Control and International MARC Programme, New Providence, London, 1994.

[URI] T. Berners-Lee, R. Fielding, L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax*. IETF RFC 2396. http://www.ietf.org/rfc/rfc2396.txt

[vanRijsbergen79] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, 1979.

[Varlamis01] I. Varlamis, M. Vazirgiannis. *Bridging XML-schema and Relational Databases: A System for Generating and Manipulating Relational Databases Using Valid XML Documents*. Proceedings of the ACM Symposium on Document Engineering, 2001. pp. 105-114.

[Vianu01] V. Vianu. *A Web Odyssey: From Codd to XML*. Proceedings of 20th Symposium on Principles of Database Systems, 2001. pp. 1-15.

[Vidaković02] M. Vidaković, B. Milosavljević, Z. Konjović. *Implementacija servera za uzajamnu katalogizaciju u EJB tehnologiji*. Zbornik radova YU INFO'02, Kopaonik, 2002.

[Voorhees85] E.M. Voorhees. *The Cluster Hypothesis Revisited*. Proceedings of SIGIR-85, 8th ACM Conference on Research and Development in Information Retrieval, 1985. pp. 188-196.

[Voorhees95] E.M. Voorhees, N.K. Gupta, B.Johnson-Laird. *The Collection Fusion Problem*. Proceedings of the Third Text Retrieval Conference (TREC-3), 1995. pp. 95-104.

[Wartick92] S. Wartick. *Boolean Operations*. In W.B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structures & Algorithms*. Prentice-Hall, 1992.

[Wen02] J.-R. Wen, H.-J. Zhang. *Query Clustering in the Web Context*. In W. Wu, H. Xiong, S. Shekhar, editors, *Information Retrieval and Clustering*. Kluwer Academic Press, 2002.

[Wilkinson91] R. Wilkinson, P. Hingston. *Using the Cosine Measure in a Neural Network for Document Retrieval*. Proceedings of the 14th ACM SIGIR Conference on Research and Development in Information Retrieval, 1991. pp. 202-210.

[Wong01] R.K. Wong. *The Extended XQL for Querying and Updating Large XML Databases*. Proceedings of the ACM Symposium on Document Engineering, 2001, pp. 95-104.

[Wong85] S.K.M. Wong, W. Ziarko, P.C.N. Wong. *Generalized Vector Space Model in Information Retrieval*. Proceedings of 8th ACM SIGIR Conference on Research and Development in Information Retrieval, 1985. pp. 18-25.

[Wood98] M.E.J. Wood, N.W. Campbell, B.T. Thomas. *Iterative Refinement by Relevance Feedback in Content-Based Digital Image Retrieval*. Proceedings of MM-98, 6th ACM International Conference on Multimedia, 1998. pp. 13-20.

[Wu81] H. Wu, G. Salton. *The Estimation of Term Relevance Weights Using Relevance Feedback*. Journal of Documentation, Vol. 37, No. 4, 1981. pp. 194-214.

[XLink] S. DeRose, E. Maler, D. Orchard. *XML Linking Language (XLink)*. W3C Recommendation, 2001. http://www.w3.org/TR/xlink

[XML] T. Bray, J. Paoli, C.M. Sperber-McQueen, E. Maler. *Extensible Markup Language (XML) 1.0 (Second Edition)*. W3C Recommendation, 2000. http://www.w3.org/TR/REC-xml

[XMLInfoset] J. Cowan, R. Tobin. *XML Information Set*, W3C Recommendation, 2001. http://www.w3.org/TR/xml-infoset/

[XMLNamespaces] T. Bray, D. Hollander, A. Layman. *Namespaces in XML*. W3C Recommendation, 1999. http://www.w3.org/TR/1999/REC-xml-names-19990114/

[XMLQL] A. Deutsch, M. Fernandez, D. Florescu, A.Y. Levy, D. Suciu. *XML-QL: A Query Language for XML*. Submission to W3C, 1998. http://www.w3.org/TR/NOTE-xml-ql/

[XMLSchema1] H.S. Thompson, D. Beech, M. Maloney, N. Mendelsohn. *XML Schema Part 1: Structures*. W3C Recommendation, 2001. http://www.w3.org/TR/\-xmlschema-1

[XMLSchema2] P.V. Biron, A. Malhotra. *XML Schema Part 2: Datatypes*. W3C Recommendation, 2001. http://www.w3.org/TR/xmlschema-2

[XPath] J. Clark, S. DeRose. *XML Path Language (XPath) Version 1.0*. W3C Recommendation, 1999. http://www.w3.org/TR/xpath

[XQL] J. Robie. *XQL (XML Query Language)*. 1999. http://www.ibiblio.org/xql/xql-proposal.html

[XQuery] D. Chamberlin, J. Clark, D. Florescu, J. Robie, J. Siméon, M. Stefanescu. *XQuery 1.0: An XML Query Language*. W3C Working Draft, 2001. http://www.w3.org/TR/2001/WD-xmlquery-req-20010215/

[XQueryDataModel] M. Fernández, A. Malhotra, J. March, M. Nagy, N. Walsh. *XQuery 1.0 and XPath 2.0 Data Model*. W3C Working Draft, 2002. http://www.w3.org/TR/query-datamodel/

[XSLT] J. Clark. *XSL Transformations (XSLT) 1.0*. W3C Recommendation, 1999. http://www.w3.org/TR/xslt

[Xu96] J. Xu, W.B. Croft. *Query Expansion Using Local and Global Document Analysis*. Proceedings of SIGIR-96, 19th ACM Conference on Research and Development in Information Retrieval, 1996. pp. 4-11.

[Yoshikawa01] M. Yoshikawa, T. Amagasa, T. Shimura, S. Uemura. *XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases*. ACM Transactions on Internet Technology, Vol. 1, No. 1, 2001. pp. 110-141.

[Zarić04] M. Zarić, D. Okanović, Z. Protić. *Searching the Digital Library of Theses and Dissertations*. Intl. Conf. on Distributed Library Information Systems, Ohrid, June 1-6, 2004. (in print).

[Zeremski02] М. Зеремски**.** *Моделирање UNIMARC формата у XML технологији.* Магистарска теза, Природно-математички факултет, Департман за математику и информатику, Нови Сад, 2002. http://diglib.ns.ac.yu/ndltd/docs/set1/ndltd64/ZeremskiMMagistarskaTeza.pdf

[Zhang95] J. Zhang. *Application of OODB and SGML Techniques in Text Database: An Electronic Dictionary System*. ACM SIGMOD Record, Vol. 24, No. 1, 1995. pp. 3-8.